FP6-IST-002020

COGNIRON

The Cognitive Robot Companion

Integrated Project

Information Society Technologies Priority

Joint RA4 Deliverable

D4.2005

Report on *Successful Decomposition of the Task into Uncorrelated Constraints and Extraction of the Goals*, on *Studies and Methods for Solving the Correspondence Problem in Context* and on *Building of and Reasoning over Complex Task Knowledge.*

Due date of deliverable: 31/12/2005
Actual submission date: 25/01/2006

Start date of project: January 1st, 2004 **Duration:** 48 months

Contributing partners:
EPFL, UH, UKA

Revision: Final
Dissemination Level: PU

Executive Summary

This deliverable reports on work conducted as part of research activity 4 (RA4) on *Task and Skill Learning* during month T12-T24. During that period RA4 was divided into three subworkpackages, namely:

WP4.1: Inferring the goals, representation and metric of the task

WP4.2: Social Context for the Correspondence Problem

WP4.3: Incremental Knowledge Acquisition of Complex Tasks

This deliverable is composed of three separate reports that describe work conducted in each of these subareas, by the three leading partners EPFL, UH and UKA (note that each subreport has its own list of references, see the table of content in page 4). Each work provides one piece of the large puzzle we aim to solve, see Figure 1. In particular, work done at EPFL primarily on extracting the important features of the task complements work done at UH on solving the correspondence problem. While those works take complementary bottom up approaches to the learning of skills, work conducted at UKA follows a rather top-down approach that complement work at EPFL and UH by addressing the problem of inserting prior knowledge to allow incremental learning of tasks (i.e. combination of known skills).

Role of RA4 - Learning Skills and Tasks' - in Cogniron

Learning Skills and Tasks is fundamental to the development of cognitive robot companion, and, thus, is crucial to the project COGNIRON. For the companion to show adaptive, long-life learning behavior, it must be capable of acquiring new skills when required (e.g. change of workplace or of habit on the user's part). It must be capable of reuse (in the sense of bootstrapping knowledge) and incremental acquisition of skills through the learning of complete tasks.

Relevance to COGNIRON Functions and Key Experiments

RA4 contributes directly to the KE3 experiment. The results of WP4.1 and WP4.2 will be combined and demonstrated as part of the KE3 experiment, following the script Learning Skills: Arranging and interacting with objects. The results of workpackage WP4.3 will be demonstrated in the script Learning Tasks: Serving a guest of the KE3 experiment. RA4 will provide the CF-LIF function, "Learning the important features of the task", the CF-RG function, "Recognizing gestures", needed for the completion of this first script, and the CF-LCT, "Learning complex tasks", needed for the completion of this second script. In addition, in the long run, it will contribute to and benefit from the development of the CF-GR function, "Gesture Recognition" (provided by RA2) and the function CF-LCT, "Reasoning about Task and its Own Capabilities CF-RET" (provided by RA6). A schematic of the links across the different research themes within RA4 and the above cogniron functions is given in Figure 1.

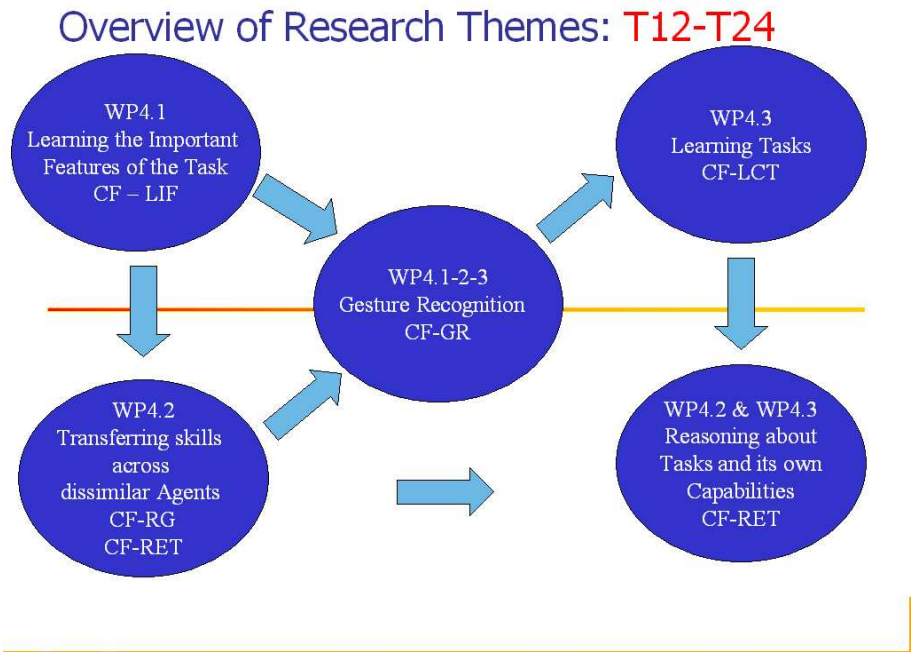


Figure 1: Schematic of the interaction across the various subworkpackages of RA4 and their relationships with the cogniron functions CF-LIF ("Learning the important features of the task"), CF-RG ("Recognizing gestures"), CF-LCT ("Learning complex tasks"), CF-GR function ("Gesture Recognition") and CF-LCT ("Reasoning about Task and its Own Capabilities CF-RET".)

CF-LIF: Learning Important Features of a Task

For the robotic companion to be able to selectively learn from observing humans performing a given task, it must be capable of extracting what is relevant and discarding what is irrelevant in a given demonstration. This is the purpose of CF-LIF.
Work conducted at EPFL and reported in Section 1 contributes to CF-LIF by developing an architecture for automatic extraction of the important features of the task [2, 5]. The features that are statistically invariant across the various demonstrations are deemed as the important features.
In addition, the work conducted at UH and reported in Section 2, toward a characterization of *space of effect, state and action metrics* [7, 3] informs the development of algorithms for automatic metric extraction (cf. WP4.1).

CF-RG: Learning to Reproduce Gestures

The robotic companion in COGNIRON will have to be able to replicate a demonstrated task by its user (imitate), although its embodiment and affordances will differ from those of a human (i.e. the robot needs to address the *correspondence problem*).
The work conducted at UH and reported in Section 2 contributes to CF-RG by the development of the JABBERWOCKY system [7], a system that uses captured motion data of a human demonstrating a task to produce action commands that if executed by an imitating agent result in achieving corresponding effects. The system is able to generalize across dissimilar initial object configurations given the task subgoal granularity and appropriate effect metrics, generating action commands targeted for

multiple imitator platforms, both simulated in software and in hardware. The system presented in [3] also contributes to CF-RG by presenting a novel generic approach to the correspondence problem, using body-mapping for cases of state and/or action matching formalized via correspondence induced metrics.
In addition, the work conducted at EPFL and reported in Section 1 provides methods for optimizing locally the path to be reproduced according to the task's metric (learned in CF-LIF), taking into account the features of the task, as well as the body of the robot and external constraints.

CF-LCT: Learning complex tasks

Learning and Reasoning are two of the most important capabilities of a Cognitive Robot Companion. The objective of the CF-LCT is to acquire the knowledge a robot system must have in order to accomplish a certain task from the household domain ("learning") and to gain insights from the consideration of the complete task knowledge a robot has as a whole ("reasoning"). Work conducted at UniKarl provided methods for the acquisition of raw data from dedicated sensors and several processing steps for denoising, segmentation into elementary operations and restructuration into the hierarchical task representation found in WP4.3 during the first project phase. During the second phase, the work reported in section 3 contributed to CF-LCT by providing simple reasoning methods on the learned task knowledge, like the learning of the sequential constraints of a task.

KE3: Learning Skills and Tasks

KE3 stresses the robot's ability to learn from implicit (imitation learning) and explicit (verbal interaction) teaching. As the embodiment of the human user and the robotic companion will be dissimilar, a *correspondence problem* must be solved, defined by the imitator's embodiment, the sub-goal granularity, and the metrics used to match aspects of behaviour. Since the demonstrations will necessarily encapsulate mistakes on the user's parts or, more generally, irrelevant data, these should be discarded. Finally, since the number of demonstrations must be kept to their minimum (5 to 10 per task) for the training of the robot to be bearable to the user, the robot must exploit advanced tools for statistical analysis of the data to speed up learning (Section 2), as well as use prior knowledge (addressed in Section 3).

Contents

1 Report on successful decomposition of the task into uncorrelated constraints and extraction of the goals
Lead Partner: EPFL

1.1 Introduction

Work conducted as part of WP4.1 during T12-T24 addressed the problems of *what to imitate* in simple uni- and bi-manual manipulatory tasks. To recall, the *what to imitate* issue relates to the generic problem of how to decompose a given task into a set of goals or sub-goals, that encapsulate the important features of the task to be imitated. The problem can hardly be decoupled from the secondary issue of *how to imitate*, tackled in WP4.2, as the demonstrator's subgoals must be meaningful and feasible for the imitator to be successful in its imitation. Thus, during this past year, we have developed a method for solving both issues in a single generic framework.
To start, we assume that the relevant features of a task consist of the features that presented spatio-temporal invariances across the various demonstrations. This is true if the demonstrations are conducted in a systematic fashion. Thus, to solve the *What to imitate* issue, we use a probabilistic method, based on Hidden Markov Models (HMM), for extracting the relative importance of reproducing either the gesture or the specific hand path in a given task (the encoding and decoding of gestures using PCA/ICA + HMM has been documented in details in [3, 4]). This, then, allows us to determine a metric of imitation performance. Then, based on this task decomposition, we can solve the *how to imitate* issue for various contexts, by computing the trajectory for each of the robot's limbs that optimizes the metric of the task, given a set of robot's body constraints.
Figure 2 shows the information flow in the complete model. The reader should refer to [2] for details.

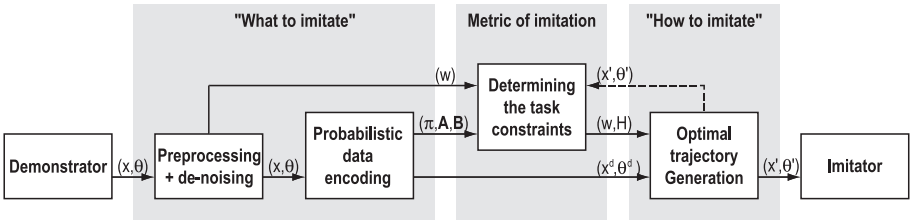


Figure 2: Information flow across the model. The signals from the different demonstrations (in our case, joint angle trajectories and position of the objects) are first segmented and analyzed in details to extract the spatio-temporal invariants, deemed as the *relevant features* of the task. This is done through, first a reduction of the dimensionality through Principal/Independent Component Analysis (PCA/ICA) and a statistical encoding in Hidden Markov Models (HMM). This information is then used to determine a metric of imitation performance, that combines linearly the relative importance of each feature extracted in the first stage of processing. Finally, the optimal reproduction is determined by minimizing the metric of the task, subjected to the robot's body constraints. In other words, the system finds a tradeoff between reproducing at best the important features of the task and not breaking the robot.

1.2 Results

We validated the method in a series of experiments where a human demonstrator taught through kinesthetics a humanoid robot how to manipulate simple objects [5, 6].

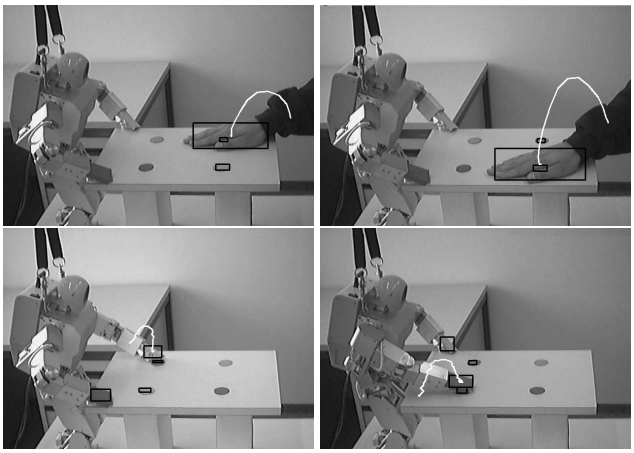


Figure 3: *Top:* Demonstration of an ipsi- (*top-left*) and contralateral (*top-right*) motion of the right arm to reach for a goal-target (color dot). The motion of the demonstrator as well as the location of the target are tracked in real time by motion captors and a stereo-vision pair of camera, respectively. *Bottom:* Reproduction by the humanoid robot of the motion candidate that satisfies best its hierarchy of goals. When the target is sufficiently close to the robot, the robot reproduces faithfully the whole demonstration, using the correct arm (in mirror fashion), following the same joint trajectories and reaching correctly for the target (bottom-left). However, when the target is too far for the robot to both reach for the target and reproduce the demonstrated gesture, the robot reaches for it with the arm closest to the goal, hence, performing only a partial reproduction of the demonstration (bottom-right). The trajectories displayed on the images are the ones tracked by the vision system. The squares show the objects detected during the demonstration and the reproduction.

In [5], we showed how such a controller for imitation learning can allow the robot to be selective in its imitation, depending on the context and on its hierarchy of goals to satisfy; a behavior similar to that displayed by young children [1]. Figure 3 illustrates the result of this study in a simple goal-directed imitation task. One can see the demonstrator performing a reaching motion for a target (colored-dot), using either an ipsi-lateral motion (arm closest to the target) or a contralateral motion (arm furthest to the target). The robot tracks the motions of the demonstrator as well as the location of the target continuously and analyzes those, so as to extract the important features of the task.
Once the demonstration terminated, the robot produces the motion candidate that satisfies best its hierarchy of goals. When the target is sufficiently close to the robot, the robot reproduces faithfully the whole demonstration, using the correct arm (in mirror fashion), following the same joint trajectories and reaching correctly for the target (bottom-left). However, when the target is too far for the robot to both reach for the target and reproduce the demonstrated gesture, the robot reaches for it with the arm closest to the goal, hence, performing only a partial reproduction of the demonstration (bottom-right).
In [2], we further validated this method to a wide range of tasks, showing that such controller can allow the robot to adapt continuously its reproduction when the context may change. We showed that, in each case, the robot managed to adapt its motions correctly, so as to reproduce the important qualitative features of each task, depending on the context.
In [6], we explored an alternative way of decomposing the task, using a combination of Gaussian Mixture Models and HMM. Further, we also extended the learning of the task metric by considering a time-dependent version of the original cost function H, presented in [5]. This generic cost function measures the variations of the constraints and of the dependencies across the variables over time. It

is continuous, positive, definite and can be estimated at any time along the trajectory, by interpolating between the finite set of Gaussians used to encode the trajectories.
Let $\{\vec{\theta}^d(t), \vec{x}^d(t)\}$ be the desired trajectories for the joints and hand path, generalized forms of the signals gathered during the demonstrations. Let $\{\vec{\theta}(t), \vec{x}(t)\}$ be the candidate trajectories for reproducing the motions. The metric of imitation performance (cost function for the task) H is then given by:

$$\begin{aligned} H &= H(\vec{\theta}^d(t), \vec{\theta}(t), \vec{x}^d(t), \vec{x}(t)) \\ &= \frac{1}{2}\left(\vec{\theta}(t) - \vec{\theta}^d(t)\right)^T \mathbf{W}^\theta(t)^{-1} \left(\vec{\theta}(t) - \vec{\theta}^d(t)\right) \\ &+ \frac{1}{2}\left(\vec{x}(t) - \vec{x}^d(t)\right)^T \mathbf{W}^x(t)^{-1} \left(\vec{x}(t) - \vec{x}^d(t)\right) \end{aligned}$$

H=0 corresponds to a perfect reproduction. The diagonal elements of the $\mathbf{W}^\theta(t)$ (4×4 matrix) and $\mathbf{W}^x(t)$ (3×3 matrix) matrices give a measure of the relative importance of each set of variables, while the other elements give a measure of the correlations across the joint angles and the 3D hand position of the demonstrator across the various presentations of the task (5 to 10 presentations at maximum).

1.3 Discussion and Future Work

This year, we developed a method to: 1) extract the important features, i.e. the spatio-temporal correlations across the multivariate dataset, of the task, 2) to determine a generic metric to evaluate the robot's imitation performance, and, finally, 3) to optimize the robot's reproduction of the task when placed in a new context according to the task metric.
The system we presented for solving the *what-to-imitate* and *how-to-imitate* issues is generic, in the sense that it makes no assumption on the robot's configuration (number of degrees of freedom and length of segments). Moreover, it produces smooth trajectories, even at the limits of the robot's workspace, which makes it a suitable robot controller.
In the majority of the experiments conducted this year, the robot was being taught through kinesthetic learning; i.e. by showing kinesthetically how to perform a task, the user "embodies" the robot's body. This simplified considerably the correspondence problem and overlooked the problem of having different embodiments. However, by testing the system in different situations than those taught, we tackled the second aspect of the correspondence problem, namely the variation in the context. In future work, we will move back to teaching the robot through external motion captors attached to the human body and combine methods developed by UH to tackle the first aspect of the correspondence problem.
Further, we will also consider explicit means of training the robot, in the form of head nods and pointing gestures to help refine the current purely statistical form of learning of the robot.

1.4 References

References

[1] H. Bekkering, A. Wohlschlger, and M. Gattis. Imitation of gestures in children is goal-directed. *Quarterly Journal of Experimental Psychology*, 53A (1):153–164, 2000.

[2] A. Billard, S. Calinon and F. Guenter (2006) Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks *Robotics & Autonomous Systems*, special issue on the social mechanisms of robot programming by demonstration (In press).

[3] Calinon, S. and Billard, A. (2005) Learning of Gestures by Imitation in a Humanoid Robot. *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions. Dautenhahn, K. and Nehaniv, C.L. (eds.)*. Cambridge University Press (In Press).

[4] S. Calinon and A. Billard (2005) Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM. In Proceedings of the 22nd International Conference on Machine Learning, 7-10 August, Bonn, Germany, 2005.

[5] Calinon, S., Guenter, F. and Billard, A. (2005) Goal-Directed Imitation in a Humanoid Robot. In Proceedings of the International Conference on Robotics and Automation, ICRA'05, Barcelona, Spain, 18-21 April 2005.

[6] Calinon, S., Guenter, F. and Billard, A. (2006) On Learning the Statistical Representation of a Task and Generalizing it to Various Contexts In Proceedings of the International Conference on Robotics and Automation, ICRA'06, Orlando, USA, May 15-18 2006.

1.5 Annexes

In Annexes, we include 2 journal papers [2, 3] and 3 conference papers [4, 5, 6] that report on work conducted as part of WP4.1 and WP4.2 by EPFL during this past year. All of those have been published or have been accepted for publication this year.
In addition, we have submitted two journal papers for the *International Journal of Humanoid Robotics* and the *IEEE Transactions on Man, Systems and Cybernetics*, which we do not include here.

2 Report on studies and methods for solving the correspondence problem in context
Lead Partner UH

This part of the report describes the work on WP4.2 "Social Context for the Correspondence Problem", as part of the University of Hertfordshire's second phase of the program of research for the COGNIRON project.

Role of *Social Context for the Correspondence Problem* in COGNIRON

The work presented in this report for WP4.2 (*"Social Context for the Correspondence Problem"*) helps to provide rigorous scientific foundations for COGNIRON functions CF-RG:*"Learning to reproduce gestures"* and CF-LIF:*"Learning important features of a task"*, and develops in synergy with workpackage WP4.1 (*"Inferring the goals, representation and metric of the task"*) by

- continuing the development of a broad theoretical and practical systematic scientific framework from which to select what aspects of behaviour to imitate (*goal and sub-goal metrics for actions, states, effects*) going beyond the current state-of-the-art,
- by extending the implementation of a multi-targetable architecture which allows the specific features extracted from human (or robot) demonstrations to be imitated on different platforms, and
- investigating the role of different types of observation in the social relationship between demonstrator/teacher and imitator in order to achieve efficient learning.

Relation to the Key-Experiments

The work presented in this part of the report is relevant to KE3 script: *Learning Skills "Arranging and interacting with objects"*. The script stresses the robot's ability to learn from implicit (imitation learning) and explicit (verbal interaction) teaching. As the embodiment of the human user and the robotic companion will be dissimilar, a *correspondence problem* must be solved, defined by the imitator's embodiment, the sub-goal granularity, and the metrics used to match aspects of behaviour. The work in WP4.2 addresses the correspondence problem in a social context. The relationship between demonstrator/teacher and imitator in spatial and interactive terms may vary over time, therefore it is necessary to understand the effect of different aspects of this relationship and their exploitation in order to achieve efficient robot programming by demonstration. Aspects include the dynamic versus static spatial relationship as well as feedback via moulding and scaffolding by the teacher. Extensions of the JABBERWOCKY system could be used to generate action commands that a robotic companion could use to imitate the tasks demonstrated by a human user. The corresponding solutions produced by the system can be targeted to multiple robotic platforms and adapt to arbitrary starting conditions.

2.1 Introduction

Building upon the results of the previous project phase, we carried out further experiments on robotic imitation with more sophisticated target imitator platforms (see section 2.2.1), while developing action, state and effect metrics to guide imitative behaviour (see section 2.2.1). Furthermore, we addressed the social context of imitation in robotic experiments systematically characterizing the spec-

trum from static to dynamic observation learning (see section 2.2.2). Initial studies into social scaffolding and moulding have also been carried out, demonstrating proof-of-concept for a novel hierarchical extensible architecture useful in the teaching of robots by human demonstration (see section 2.2.3). Moving toward more complex cultural transmission in populations, simulation and robotic studies have demonstrated the capacity of social learning to serve as a primary mechanism for the beginnings of culture in interactive robots with dissimilar embodiments (see section 2.2.4). A series of experimental runs and a small pilot user study were conducted in order to evaluate the performance of a system designed for robot imitation, with results suggesting that there is good alignment between a quantitative system-centered assessment and a more qualitative human-centered assessment of imitative performance (see section 2.2.5).
For detailed discussions of the relationship of COGNIRON work for WP4.2 in the context of other scientific work, see the attached publications (in the Appendices).
The relevance to the COGNIRON Functions and Key Experiments is discussed above.
Section 2.3 summarizes and discusses the studies carried out and associated methods for solving the correspondence problem for robotic programming by demonstration in a social context, and gives directions for future work.
Section 2.5 gives a selected list of publications included in the Appendices describing in detail the work presented in section 2.2.

2.2 Results

2.2.1 Correspondence Mappings

One of the fundamental problems in imitation is the *correspondence problem*, how to map between the actions, states and effects of the model and imitator agents, when the embodiment of the agents is dissimilar. In our approach, the matching is according to different metrics and granularity.

JABERWOCKY and Related Systems Based on a single demonstration of an object manipulation task by a human and using a combination of *effect metrics*, the JABBERWOCKY system is shown to produce correspondence solutions that are then performed by an imitating agent, generalizing with respect to different initial object positions and orientations in the imitator's workspace. The system is able to target multiple imitator platforms. Depending on the particular metrics and granularity used, the corresponding effects will differ (shown in examples), making the appropriate choice of metrics and granularity depend on the task and context.
This work was presented in [5, 6, 7]. The JABBERWOCKY system has been released to other partners for use as a service in the COGNIRON architecture. This release includes documentation and tutorials for a new version ported to MATLAB.
Another related system (also implemented in MATLAB), for solving the correspondence problem via body-mapping, has been developed during this second phase and provides state and action metrics (induced by correspondence mappings) - see section 2.2.1 below; details are in [3] (see Appendix).

Characterization of Space of Action, State and Effect Metrics In [3] we have addressed the problem of body mapping in robotic imitation where the demonstrator and imitator may not share the same embodiment (degrees of freedom (DOFs), body morphology, constraints, affordances and so on). Body mappings are formalized using a unified (linear) approach via correspondence matrices, which allow one to capture partial, mirror symmetric, one-to-one, one-to-many, many-to-one and many-to-many associations between various DOFs across dissimilar embodiments. We show how

metrics for matching state and action aspects of behaviour can be mathematically determined by such correspondence mappings, which may serve to guide a robotic imitator. The approach is illustrated and validated in a number of simulated 3D and robotic examples, using agents described by simple kinematic models and different types of correspondence mappings.
In [7] we provide a characterization of *space of effect metrics* (relative and absolute position, displacement and rotation of manipulated objects, as well as mirror symmetrical and related variants) that compliments the characterization of *space of action and state metrics* presented in [3].

2.2.2 Dynamic versus Static Observation

Research into robotic social learning, especially that concerned with imitation, often focuses at differing ends of a spectrum from *observational learning* at one end, to where a closer shared context is considered, for example, *following* or *matched-dependent behaviour* at the other. We study the implications and differences that arise when carrying out experiments both at the extremes and within this spectrum.
As a minimal physical test-bed, Khepera robots with minimal sensory capabilities were used, and after training, experiments were carried out where an imitating robot perceives the dynamic movement behaviours of another model robot carrying a light source. The robot learns the movement behaviour of the model by either statically observing the model, dynamically observing the model or by following the model. It finally re-enacts the learnt behaviour.
We compare the results of these re-enactments and illustrated the differences and trade-offs that arise between static observational and reactive following learning methods. We also considered circumstances where, for this robotic embodiment, dynamic observation has both advantages and disadvantages when compared to static observation, and discuss the implications that arise from using and combining these types of social learning.
This work is presented in [11, 12].

2.2.3 Moulding and Scaffolding: Initial Study

Programming robots to carry out useful tasks is both a complex and non-trivial exercise. A simple and intuitive method to allow humans to train and shape robot behaviour is clearly a key goal in making this task easier.
We developed an approach to this problem based on studies of social animals where two teaching strategies are applied to allow a human teacher to train a robot by *moulding* its actions within a carefully *scaffolded* environment. Within these environments, sets of competences can be built by building state/action memory maps of the robot's interaction within that environment. These memory maps are then polled using a k-nearest neighbor based algorithm to provide a generalized competence. We take a novel approach in building the memory models by allowing the human teacher to construct them in a hierarchical manner. This mechanism allows a human trainer to build and extend an action-selection mechanism into which new skills can be added to the robot's repertoire of existing competencies. These techniques are implemented on physical Khepera miniature robots and validated on a variety of tasks, also forming a basis of human-robot cultural transmission [13] (see also section 2.2.4 below).

2.2.4 Cultural Transmission

Social robotics opens up the possibility of individualized social intelligence in member robots of a community, and allows us to harness not only individual learning by the individual robot, but also the acquisition of new skills by observing other members of the community (robot, human, or virtual).

We demonstrate the use of social learning mechanisms like the ALICE generic imitation framework [1, 2] for transmission of skills between robots, and present some examples of transmission of a skill through a chain of simulated robots, despite differences in embodiment of agents involved. These simple examples demonstrate that by using social learning and imitation, *cultural transmission* is possible among robots, even heterogeneous groups of robots [4]. In addition, the emergence of shared categories at population level via imitation and self-organization is shown in COGNIRON work carried out at VUB, using a physical robotic arm platform [9].

2.2.5 Evaluation of Performance of Systems Designed for Robot Imitation

A series of experimental runs and a small pilot user study were conducted to evaluate the performance of a system designed for robot imitation. Performance assessments of similarity of imitative behaviours were carried out by machines and by humans: the system was evaluated quantitatively (from machine-centric perspective) and qualitatively (from a human perspective) in order to study the reconciliation of these views. The experimental results presented illustrated how the number of *exceptions* can be used as a performance measure by a robotic or software imitator of an object manipulation behaviour. (In this context, exceptions are events when the optimal displacement and/or rotation that minimize the dissimiliarity metrics used to generate a corresponding imitative behaviour cannot be directly achieved in the particular context.) Results of the user study giving similarity judgments on imitative behaviours were to examine how the quantitative measure of the number of exceptions (*from a robot's perspective*) used in the experiments corresponds to the qualitative evaluation of similarity (*from a human's perspective*) for the imitative behaviours generated by the JABBERWOCKY system. Results suggest that there is a good alignment between this quantitative system-centered assessment and the more qualitative human-centered assessment of imitative performance.
This work was presented in [8].

2.3 Summary and Discussion

Building upon the results of the previous project phase, we have carried out further experiments on robotic imitation with more sophisticated target imitator platforms, while developing action, state and effect metrics to guide imitative behaviour in social context. The characterization of space of metrics for imitation has been extended to encompass state and action metrics induced via different types of body-mappings, complementing work on effect metrics.
Social context has previously been largely ignored by other work on robot programming by demonstration. For the COGNIRON project, in which situated human-robot interaction is crucially important, it is necessary to address the social context of imitation. This has been studied here in robotic experiments systematically, characterizing the spectrum from static to dynamic observation learning.
Initial studies into social scaffolding and moulding have also been carried out, demonstrating proof-of-concept for a novel hierarchical extensible architecture useful in the teaching of robots by human demonstration.
Here, the roles of interaction and feedback become important for social learning by robots sharing an environment with humans.
Moving towards more complex cultural transmission in populations, simulation and robotic studies have demonstrated the capacity of social learning to serve as a primary mechanism for the beginnings of culture in interactive robots with dissimilar embodiments.
A series of experimental runs and a small pilot user study were conducted in order to evaluate the performance of a system designed for robot imitation, with results suggesting that there is good alignment

between a quantitative system-centered assessment and a more qualitative human-centered assessment of imitative performance.
Work described here addresses the correspondence problem in a social context. The relationship between demonstrator/teacher and imitator in spatial and interactive terms may vary over time, therefore it is necessary to understand the effect of different aspects of this relationship and their exploitation in order to achieve efficient robot programming by demonstration. Aspects include the dynamic versus static spatial relationship [10, 11, 12] as well as feedback via moulding and scaffolding by the teacher [13]. Extensions of the JABBERWOCKY system [5, 6, 7] could be used to generate action commands that a robotic companion could use to imitate the tasks demonstrated by a human user. The corresponding solutions produced by the system can be targeted to multiple robotic platforms and adapt to arbitrary starting conditions.

2.4 Future Work

The work will continue to advance scientific understanding by developing autonomous methods for how, what and who to imitate, that take into account the social context and the relationship between the robot learner and a teacher.
Social interaction plays a fundamental role in the development of cognition. Building on results from WP4 for T0-T24 on the correspondence problem (how to imitate) and related issues, we will focus on teaching robots using imitation, self-imitation and scaffolding with the robot's own activities and sensorimotor experiences at the center of the learning process. Different types of observational learning and imitation as well as self-imitation of teacher moulded actions will be explored with special attention to the situated development of the relationship of robot and teacher in time, space, and social context.

2.5 List of Publications Included in the Appendices

The following is a selection of 6 out of the 10 new COGNIRON publications for WP4.2 included as Appendices of this deliverable, reporting the detailed work summarized in section 2.2.

Correspondence Mappings

Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (2005d). Imitating using JABBERWOCKY to achieve corresponding effects in context. Computer Science Technical Report 441, University of Hertfordshire, Hatfield, Hertfordshire, UK. Submitted to *IEEE Trans. Syst., Man, Cybern. Part A: Systems and Humans*.
Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (2005a). Correspondence mapping induced state and action metrics for robotic imitation. Computer Science Technical Report 440, University of Hertfordshire, Hatfield, Hertfordshire, UK. Submitted to *IEEE Trans. Syst., Man, Cybern. Part B: Cybernetics*.

Dynamic versus Static Observation
Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2005b). Experimental comparisons of observation learning mechanisms for movement imitation in mobile robot. Computer Science Technical Report 442, University of Hertfordshire, Hatfield, Hertfordshire, UK. Submitted to *IEEE Trans. Syst., Man, Cybern. Part B: Cybernetics*.

Moulding and Scaffolding: Initial Study
Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (accepted for publication, 2006). Teaching robots by moulding behavior and scaffolding the environment. In *Proc. 1st ACM/IEEE Annual Conference on Human-Robot Interaction (HRI2006) – Salt Lake City, Utah, USA, March 2-4, 2006*.

Cultural Transmission
Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (in press, 2006a). Solving the correspondence problem in robotic imitation across embodiments: Synchrony, perception, and culture in artifacts. In Dautenhahn, K. and Nehaniv, C. L., editors, *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press.

Evaluation of Performance of Systems Designed for Robot Imitation
Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (accepted for publication, 2006b). Evaluation of robot imitation attempts: Comparison of the system's and the human's perspectives. In *Proc. 1st ACM/IEEE Annual Conference on Human-Robot Interaction (HRI2006) – Salt Lake City, Utah, USA, March 2-4, 2006*.

2.6 References

References

[1] Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (2002). Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Trans. Systems, Man & Cybernetics: Part A*, 32(4):482–496.

[2] Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (2004). Towards robot cultures? – Learning to imitate in a robotic arm test-bed with dissimilar embodied agents. *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, 5(1):3–44.

[3] Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (2005a). Correspondence mapping induced state and action metrics for robotic imitation. Computer Science Technical Report 440, University of Hertfordshire, Hatfield, Hertfordshire, UK. Submitted to *IEEE Trans. Syst., Man, Cybern. Part B: Cybernetics*.

[4] Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. (in press, 2006a). Solving the correspondence problem in robotic imitation across embodiments: Synchrony, perception, and culture in artifacts. In Dautenhahn, K. and Nehaniv, C. L., editors, *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press.

[5] Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (2005b). Achieving corresponding effects on multiple robotic platforms: Imitating using different effect metrics. In *Proc. Third International Symposium on Imitation in Animals and Artifacts – Hatfield, UK, 12-14 April 2005*, pages 10–19. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

[6] Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (2005c). An approach for programming robots by demonstration to manipulate objects: Considerations on metrics to achieve corresponding effects. In *Proc. 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '05)*, pages 61–66.

[7] Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (2005d). Imitating using JABBERWOCKY to achieve corresponding effects in context. Computer Science Technical Report 441, University of Hertfordshire, Hatfield, Hertfordshire, UK. Submitted to *IEEE Trans. Syst., Man, Cybern. Part A: Systems and Humans*.

[8] Alissandrakis, A., Nehaniv, C. L., Dautenhahn, K., and Saunders, J. (accepted, 2006b). Evaluation of robot imitation attempts: Comparison of the system's and the human's perspectives. In *Proc. 1st ACM/IEEE Annual Conference on Human-Robot Interaction (HRI2006) – Salt Lake City, Utah, USA, March 2-4, 2006*.

[9] Belpaeme, T., de Boer, B., and Jansen, B. (in press, 2006). The dynamic emergence of categories through imitation. In Dautenhahn, K. and Nehaniv, C. L., editors, *Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions*. Cambridge University Press.

[10] Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2004). An experimental comparison of imitation paradigms used in social robotics. In *Proc. IEEE RO-MAN 2004, 13th IEEE International Workshop on Robot and Human Interactive Communication, September 20-22, 2004 Kurashiki, Okayama Japan*, pages 691–696.

[11] Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2005a). An examination of the static to dynamic imitation spectrum. In *Proc. Third International Symposium on Imitation in Animals and Artifacts – Hatfield, UK, 12-14 April 2005*, pages 109–118. Society for the Study of Artificial Intelligence and Simulation of Behaviour.

[12] Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (2005b). Experimental comparisons of observation learning mechanisms for movement imitation in mobile robot. Computer Science Technical Report 442, University of Hertfordshire, Hatfield, Hertfordshire, UK. Submitted to *IEEE Trans. Syst., Man, Cybern. Part B: Cybernetics*.

[13] Saunders, J., Nehaniv, C. L., and Dautenhahn, K. (accepted, 2006). Teaching robots by moulding behavior and scaffolding the environment. In *Proc. 1st ACM/IEEE Annual Conference on Human-Robot Interaction (HRI2006) – Salt Lake City, Utah, USA, March 2-4, 2006*.

3 Report on building of and reasoning over complex task knowledge Lead Partner UKA

3.1 Introduction

One of the most intuitive ways for a robot companion to acquire new complex task knowledge is to learn it from the human user via demonstration and interaction. The aim of WP4.3 is to study methods involved in learning complex tasks within the household domain from as few demonstrations as possible and the processes of continuous refinement of this task knowledge when additional task demonstrations become available. Symbolic reasoning methods have been successfully applied for the latter transformation.

The system for the acquisition of task knowledge has been set up in the first phase of WP4.3 and was described in [1]. A hierarchical task representation, so-called macro-operators, was found. It represents the task knowledge, as well as preconditions and effects of the actions, on different semantical levels, from the elementary, robot-near level that comprises sensory-motor couplings to the manipulation segment level. The objective of WP4.3 in the second phase was to start the application of higher level reasoning methods to this task knowledge in order to exploit the presence of multiple demonstrations of the same task.

An important preliminary for the application of higher level reasoning methods is the ability of the system to identify similar subtasks across different demonstrations of the same task. WP4.3 defined explicit subtask similarity measures that allow a reidentification of common parts of two or more different demonstrations (see section 3.2).

Once corresponding subtasks are identified by the system, it can apply reasoning methods in order to refine the task knowledge. In Phase two, reasoning on sequential dependencies and independences between subtasks was implemented, applied and tested. This higher-level knowledge is represented in task precedence graphs (see section 3.3).

The observation, that the transfer of task knowledge between humans is a highly interactive process, led to the incorporation of a speech recognition module into the task learning system. This gives the user the ability to highlight particular effects of the task to be learned by adding vocal comments to the demonstration. Though this is still work in progress, first progress was made and documented in section 3.4.

3.2 Subtask Similarity Measures

In [3] a method for the determination of the similarity of tasks and parts of tasks is described. It descends the hierarchical task representation until it compares the effects associated with the leaves of a task tree. Each effect f is weighted with a weight $w(f)$ depending on it's relevance to the task to be learned. So the base operation of the assessment of effect conformance is

$$s_F(\mathcal{A}, \mathcal{B}) = \frac{\sum_{f \in \mathcal{A} \cap \mathcal{B}} w(f)}{\sum_{f \in \mathcal{A} \cup \mathcal{B}} w(f)}. \quad (1)$$

The choice of the weight function $w(f)$ is determined with respect to the information content of all effects. The information content of a certain effect, occurring during a task demonstration, is determined by two factors. The first is the occurrence rate of the effect over the whole task knowledge of the system. This allows to assess the similarity of subtasks even when only little knowledge on the specific class is known, e.g. when the first demonstration of a certain task is given. The second is the occurrence rate in all demonstrations that belong to the same task that is to be learned. This allows

to incrementally improve the similarity measure as soon as more knowledge in form of additional demonstrations of the same task becomes available.
[3] also discusses how these two forms of knowledge can be balanced in an incrementally learning system. In the beginning, when no or only little knowledge on the task at hand is available, the overall task knowledge gains much higher influence on the similarity measure than the task specific knowledge. At later stages, when the knowledge on the task increases and gets more reliable, the system favors it over the general task knowledge that may not apply too well on the certain task at hand.

3.3 Learning of Task Precedence Graphs

Task precedence graphs encode the hypotheses the system can state on the sequential structure of a task. The learning of task precedence graphs allows the system to schedule its operations most flexibly while still meeting the goals of the task.
Task precedence graphs are defined in [2]. They are directed, acyclic graphs that contain a temporal precedence relation $\rightarrow$. For two subtask o_1, o_2, $o_1 \rightarrow o_2$ means that the subtask o_1 has to be performed before the executing robot system can start to work on o_2.
Additionally, a method to learn the task precedence graph of a task incrementally is proposed in [2]. It relies on the gradual generalization of the hypothesis on the sequential relations, coded in task precedence graphs. When the system has learned the task precedence graph $P_m = (\mathcal{N}, \mathcal{R}_m)$ from the last m demonstrations and observes a new demonstration of the same task with the most restrictive set of task precedence relations $\mathcal{R}^{D_{m+1}}$ possible, the system generalizes its hypothesis to the task precedence graph with the relations contained in

$$\mathcal{R}_{m+1} = \mathcal{R}_m \cap \mathcal{R}^{D_{m+1}}.$$

This incremental approach allows the system to obtain an initial task precedence graph after seeing a single demonstration of a task and execute this task from the first demonstration on. At the same time it is enabled to generalize this sequential concept of the task in further learning steps, leading to a more general and flexible representation of the task knowledge.

3.4 Using Vocal Comments

In human learning, interaction and vocal comments accompanying demonstrations of a task play an important role. Within phase two, first work on interactive learning has been performed and vocal comments have been integrated into the learning process.
The user is given the possibility to tell the learning system the parts of the demonstration that he/she considers important. This speeds up the learning process and learning accuracy as the learning system is able to make guided generalizations.
[4] refines the subtask similarity measures mentioned in section 3.2 by taking into account speech inputs by the user. Each effect the user accomplishes during the demonstration can be highlighted with an appropriate vocal comment. Once this is done, the system can adapt the weighting function of this effect in order to let it gain much higher weights than effects that were not explicitly mentioned by the user.
In order to achieve this, the weight function $r(f)$ that assesses the relevance of every effect f is defined as

$$r(f) = I_{voc} \cdot w_{voc}(f) + I_{demo} \cdot w_{demo}(f). \quad (2)$$

with I_{demo} and I_{voc} being the information contents of the task demonstrations (as explained in section 3.2) and the vocal comments respectively and $w_{demo}(f)$, $w_{voc}(f)$ the heuristic weight functions resulting from the demonstrations and the vocal comments.

3.5 Future Work

WP 4.3 will investigate methods and algorithms for unsupervised incremental clustering of task knowledge, leading to an hierarchically structured representation of the overall task knowledge (task taxonomy).
This task taxonomy will be used as a data base for the recognition of similarities of new task demonstrations with already demonstrated and acquired tasks and to facilitate reasoning over all demonstrations of a task, in order to fuse it into higher order task knowledge and incorporate this into the task taxonomy again.
Higher level reasoning methods will be developed in the future that contain the learning of repetitive task patterns, the extraction of loops, their loop invariants as well as their termination conditions, the acquisition of task with alternatives and their selection criteria.

3.6 References

References

[1] D.4.3.1 report on selected categorizations of innate skills, skill representation and implementation. evaluation of machine learning techniques for enabling one shot learning. Technical report, University of Karlsruhe, 2004.

[2] M. Pardowitz, R. Zöllner, and R. Dillmann. Learning sequential constraints of tasks from user demonstrations. In *Proc. IEEE-RAS Intl. Conf. on Humanoid Robots (HUMANOIDS05)*, pages 424–429, Dec 2005.

[3] M. Pardowitz, R. Zöllner, and R. Dillmann. Incremental learning of task sequences with information-theoretic metrics. In *to be presented at the European Robotics Symposium (EUROS06)*, 2006.

[4] M. Pardowitz, R. Zöllner, S. Knoop, and R. Dillmann. Incremental learning of tasks from user demonstrations, past experiences and vocal comments. Submitted to *IEEE Trans. on Systems, Man, and Cybernetics Special Issue on Robot Learning by Observation, Demonstration and Imitation*, 2006.

3.7 Annexes

Accepted papers: [2, 3]
Submitted papers: [4]

Discriminative and Adaptive Imitation in Uni-Manual and Bi-Manual Tasks ⋆

Aude G. Billard, Sylvain Calinon, Florent Guenter

Autonomous Systems Laboratory 3 (ASL3)
School of Engineering, EPFL
Swiss Institute of Technology Lausanne EPFL
EPFL-STI-I2S-ASL3, Station 9, 1015 Lausanne, Switzerland
Corresponding Author: aude.billard@epfl.ch

Abstract

This paper addresses the problems of *what to imitate* and *how to imitate* in simple uni- and bi-manual manipulatory tasks. To solve the *what to imitate* issue, we use a probabilistic method, based on Hidden Markov Models, for extracting the relative importance of reproducing either the gesture or the specific hand path in a given task. This allows us to determine a metric of imitation performance. To solve the *how to imitate* issue, we compute the trajectory that optimizes the metric, given a set of robot's body constraints. We validate the methods in a series of experiments, where a human demonstrator teaches through kinesthetic a humanoid robot how to manipulate simple objects.

Key words: Discriminative Imitation, Learning, Humanoid Robots, Kinesthetic Teaching, HMM, Lagrange Optimization

1 Introduction

Recent advances in Robot Programming by Demonstration RbD, also referred to as *Learning by Imitation*, have identified a number of key issues that need to be solved for ensuring a generic approach to transferring skills across various agents and situations [17, 18]. These have been formulated as a set of generic questions, namely *what to imitate*, *how to imitate*, *when to imitate* and *who to imitate*. These questions were formulated in response to the large

⋆ Status: Accepted, To Appear in Robotics & Autonomous Systems Journal (Dec. 2005).

body of work in RbD that emphasized ad-hoc solutions to sequencing and decomposing complex tasks into *known* sets of actions performable by both the demonstrator and the imitator, see, e.g. [4, 12, 15, 23, 26, 27]. In contrast to these other works,the above four questions and their solutions aim at being generic in the sense of making no assumptions on the type of skills that may be transmitted. The drawback of such a generic approach is that it has yet to show how the methods will scale up, from acquiring basic skills to acquiring complex sequences of those.

In our prior work, we have addressed the *what to imitate* question, by developing a general architecture to extract the relevant features of a given task. The methods relied on computing the statistical variability of each element of the task, see [9, 6]. In this paper, we present an extension of this work to address the "how to imitate" problem for the control of uni-manual and bi-manual manipulation of objects, using a pair of 4 degrees of freedom robot arms. This leads us to tackling more generic issues of motor control, namely that of optimizing the arm controller given specific constraints. Specifically, we extend the pseudo-inverse optimization method for solving the inverse kinematics, so as to determine the optimal imitation strategy, i.e. the strategy that satisfies best all the constraints of a given task.

The issue of "how to imitate", also referred to as the *correspondence problem* [17], were first addressed very generically in a number of studies with simulated abstract agents acting in a Markov world [1, 5]. More recent work by [2, 14], also, considered the application of such a system for controlling a 2 degrees of freedom robotic arm. The solution to the correspondence problem in the latter works was, however, constrained to a particular arm and did not provide a general solution for robotic arms with an arbitrary number of degrees of freedom. Moreover, in each case, the metric for the task was given. Here, we present a method that, first, discovers the task metric and, second, extends the classical inverse kinematics solution to solve generically the correspondence problem for an arbitrary robotic arm.

Next, we illustrate the key issues at stake and present briefly the approach we take in this paper to solving those.

1.1 What-to-imitate and How-to-imitate

When learning a new task by imitation, the robot must first determine what are the relevant features of the task to imitate ("what to imitate"), and, second how to adapt its own motor program to produce an optimal imitation ("how to imitate"). Figures 1 and 2 illustrate these issues in a simple uni-manual task. Let us first consider the problem of determining the relevant features of

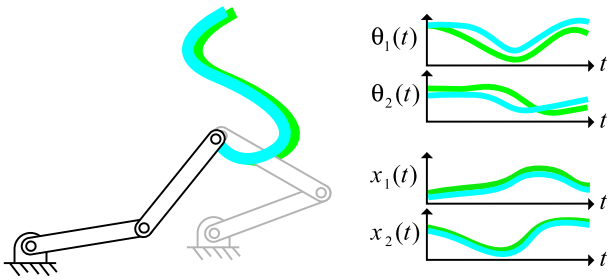


Fig. 1. Illustrative schema of the *what-to-imitate* issue. A 2 DOFs manipulator arm produces two demonstrations of a given task, namely drawing of an S figure, starting with different joint configurations. The path of the end effector given by $\mathbf{x} = \{x_1, x_2\}$ is invariant, while the joint trajectories $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ vary importantly.

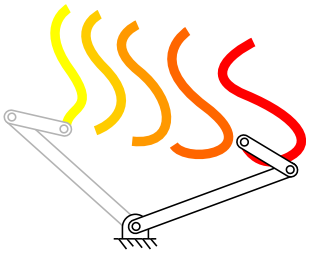


Fig. 2. Illustrative schema of the *how-to-imitate* issue. The manipulator of the imitator generates different alternatives to reproducing the task demonstrated in Figure 1 from satisfying purely the joint trajectories (left) to satisfying only the hand path (right).

a task and their relative importance.

What to imitate

Consider a two DOFs planar manipulator, performing two demonstrations of a given task and starting, in each case, with a different joint configuration, see Figure 1. If one was to record the trajectories of the joints $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ and that of the end-effector $\mathbf{x} = \{x_1, x_2\}$ of the manipulator, one would observe that the first set of variables, i.e. the joint angles, varies importantly, while the second set of variables remains quite constant across the demonstrations. Thus, the information conveyed by the hand path would appear more reliable than that conveyed by the joints. In other words, the task would appear to put stronger constraints on the hand path than on the joint trajectories. Thus, in order to reproduce the task, one would give more weight to reproducing the hand path than the joint trajectories.

In order to give a measure of the correctness of the reproduction, one needs a measure of imitation performance, i.e. a cost function. This cost function must encapsulate explicitly the task constraints as well as give a measure (metric) of the relative importance of the task constraints.

Let us define $H_1(\boldsymbol{\theta}, \boldsymbol{\theta'})$ and $H_2(\mathbf{x}, \mathbf{x'})$ as a measure of the discrepancy between

demonstrated and reproduced trajectories of the joints ($\boldsymbol{\theta}$, $\boldsymbol{\theta'}$) and hand path ($\mathbf{x}$, $\mathbf{x'}$) respectively, then, without loss of generality, we can define a global measure of the imitation performance by the weighted sum $H(\boldsymbol{\theta}, \boldsymbol{\theta'}, \mathbf{x}, \mathbf{x'}) = w_1 H_1(\boldsymbol{\theta}, \boldsymbol{\theta'} + w_2 H_2(\mathbf{x}, \mathbf{x'})$. The weights w_1 and w_2 give a measure of the relative importance of each signal. In the previous example, we would set $w_1 < w_2$ to give more importance to following the path of the end-effector than to reproducing the joint trajectories.

How to imitate

Once we have measured the relative importance of each task feature and have reported it in the cost function H, we must determine a trajectory for the joints and end-effector of the imitator's arm that is optimal with respect to the cost function. Note that the problem may not be as straightforward as it seems, since the demonstrator and the imitator may differ significantly in their embodiment (arm length, number of degrees of freedom for each arm). This is typically the case when transferring information from a human to a robot, even when the robot has a humanoid shape (as it is the case in our experiments). This idea is illustrated in Figure 2, where the 2 segments of the imitator's manipulator differ in length from those of the demonstrator. If the imitator's arm was let to replay directly the joint trajectories of the demonstrator's arm, we would end up with a very different path for the end-effector than that demonstrated. Conversely, replaying the hand path would result in major differences in the joint trajectories. Figure 2 illustrates the effect of generating different alternatives from satisfying purely the joint trajectories (left) to satisfying only the hand path (right).

Minimizing the cost function determines a trade-off between reproducing faithfully either the hand path or the joint trajectories. Note that the cost function may have several minima. In other words, there may be several solutions to the problem. Note, also, that the hand path and the joint trajectories are not independent variables, but are related to one another through a forward and inverse kinematics function, which we discuss next.

1.2 The Inverse Kinematics Problem

In classical control theory, the *inverse kinematics* usually refers to the inverse computation required to determine the position of each of the robot's joint for a given location of the robot's end-effector (usually its arm). If we define the coordinates of a manipulator as the n-dimensional vector of joint angles $\vec{\theta}$ and the position of the m-dimensional vector $\vec{x}$, the forward kinematics is given by: $\vec{x} = f(\vec{\theta})$, while the inverse kinematics is given by

$$\vec{\theta} = f^{-1}(\vec{x}) \tag{1}$$

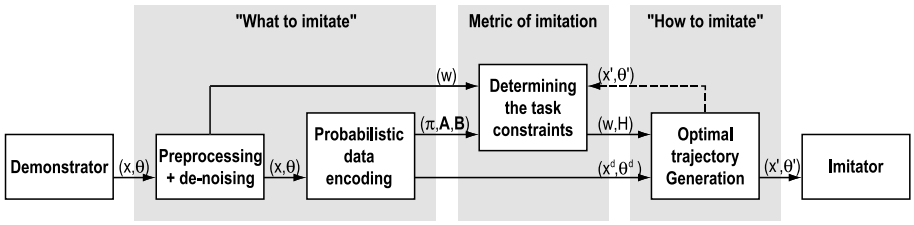


Fig. 3. Information flow across the model

where f is a continuous function $\in \mathbb{R}$.

When the problem is under constrained, i.e. when the number of degrees of freedom n exceeds the number of given variables m, i.e. $n > m$, e.g. when controlling a 4 degrees of freedom robot arm given the 3D position of the target, Equation 1 may have no solutions (degenerate case) or multiple solutions. Several solutions to this problem have been proposed, ranging from numerical solutions ([16], [24]) to geometrical solutions ([3], [25]). Moreover, in order to deal with the non-linearity of the transformation, various methods, based on multiple locally linear models, have also been explored, using either neural networks [19] or regression techniques [11]. Each solution has its advantages and drawbacks. In this paper, we redevelop the pseudo-inverse with optimization method proposed in [16] (a locally linear approximation) and adapt its form to optimize our global cost function H.

2 The Complete System

Figure 3 gives an overview of the input-output flow through the complete model. The model is composed of the following processes:

Preprocessing: We first discard the irrelevant signals (noise), by encoding the dataset $(\boldsymbol{\theta}, \mathbf{x})$ in left-right Hidden Markov Models (HMM) and by eliminating signals with too high variability.

Probabilistic Data Encoding: The remaining (relevant) signals are then encoded in a fully-connected HMM, in order to find an optimal probabilistic representation of the task.

Determining the task constraints: We then compute a measure of the relative importance of each variable of the data set, by using the probabilistic description of the task, and use this to determine the cost function (metric) of imitation performance

Optimal Trajectory Generation: We then compute (using Lagrange optimization) the trajectory that optimizes the cost function, given a set of robot's body constraints.

Next, we describe the computation carried out in each of these modules.

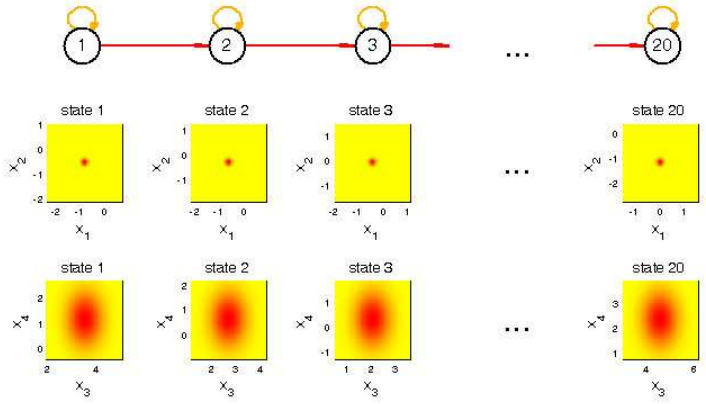


Fig. 4. Encoding of the dataset in a left-right HMM, with diagonal covariance output distribution matrix, used as a pre-processing step to extract the invariants. Here, the signals $\{x_3(t)\}$ and $\{x_4(t)\}$ present important variations, and will be discarded from the remaining processing steps.

2.1 Preprocessing-Denoising

Similarly to our prior work [7, 8], we encode the demonstrated gestures in Hidden Markov Models, so as to discard the intrinsic variability of each person's motion, and, so as to be able to regenerate a generalized form of the demonstrated gestures. However, encoding data using HMMs may lead to very poor performance, when part of the data is subjected to a noise with a large variance [1]. For this reason, we must first discard the data that may contribute to false generalization. In addition to reducing the noise, this procedure also enables us to reduce the dimensionality of the dataset, and, hence speed up learning, an advantage for applications that need to run in real time such as those we consider here.

[1] The *Expectation-Maximization* (EM) process used to train the HMMs aims at finding a local optimum to represent the data, by considering the transition probabilities and output distributions. When the variability of the output signals increases, the training process may not find a satisfying local solution. For example, when encoding two output signals, consisting of a random signal and an invariant signal, the EM training process will try to find a probabilistic representation for both signals, sometimes at the expense of the invariant signal, which is the relevant signal in our case. This problem is particularly evident when we try to model multiple signals with a fully-connected model and/or full covariance matrix, representing the output distribution, because a large set of parameters must be estimated with only a few examples. However, these HMM capabilities are still advantageous, to allow the encoding of signals presenting recurring patterns and/or correlations. Thus, a generic HMM model should nevertheless be used, at least at some point in the process, to encode general signals.

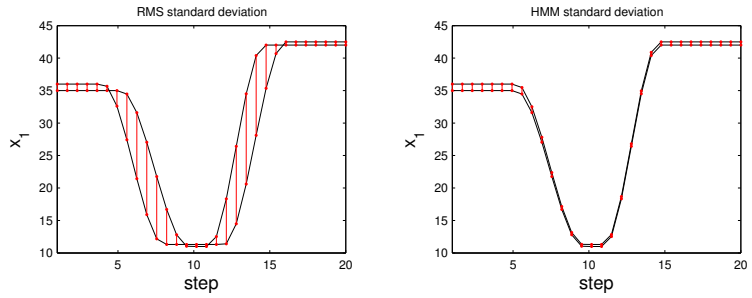


Fig. 5. Illustrative schema of the effect of applying different error measures on two instances of the signals x_1. *Left:* Euclidean distance error, *right:* error computed using the HMM representation. The vertical lines represent the error between the two signals.

Irrelevant (noisy) signals are signals that do not show any consistency across demonstrations, and, thus, may convey no information on a given task constraint. A naive approach would simply compute the Euclidean distance across all instances of a given signal $x_i(t)$, i.e. $d'_i = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(x_i(t) - \bar{x}_i(t))^2}$. Such a measure would not be adequate as it would be very sensitive to temporal distortions across the demonstrations, which are frequent in the experiments we consider (as two demonstrators may not perform the same task at the same speed). Thus, in order to compare two signals that may encapsulate non-homogeneous distortions in time, we choose to encode those in HMMs (to get rid of the time-distortion) and, then, to compare the variability of their distribution in the HMM representations (see Figure 4). The HMM encoding of the two signals may be viewed as trying to stretch temporally one signal, so as to match the other signal. Once encoded in the HMM, one may retrieve for each signal $\{x_i(t)\}_{i=1}^{K}$, a time-series of variables (the observables), whose distribution follow a set of Gaussians centered on $\{\mu_i(t)\}_{i=1}^{K}$ with variance $\{\sigma_i^2(t)\}_{i=1}^{K}$ (see Figure 5). One can then measure the variability of each signal using:

$$d_i = \sqrt{\frac{1}{T}\sum_{t=1}^{T}\sigma_i^2(t)} \tag{2}$$

For this pre-processing step, we used left-right HMM with up to 20 states and a diagonal covariance matrix. This is sufficient to find the relevant signals with only a few iterations (maximum number of EM iterations fixed to 2). Note that, in this pre-processing step, the role of the HMM is to extract robustly the spatial variations of the signals, but not to find an efficient encoding of the dataset.

The signals collected by the robot come from different modalities, with their own ranges of values, measurement units, and sensory resolution. In order to compare the intrinsic variability of different signals irrespective of their

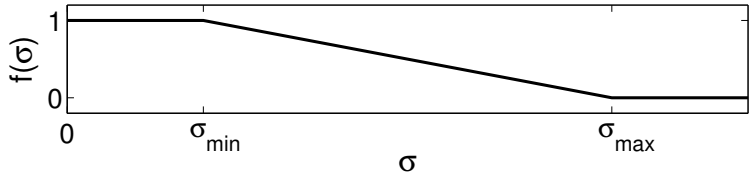


Fig. 6. Transfer function $f()$ used to transform the standard deviation of each signal to a value between 0 and 1, considering the sensibility and range of variation of the associated sensor.

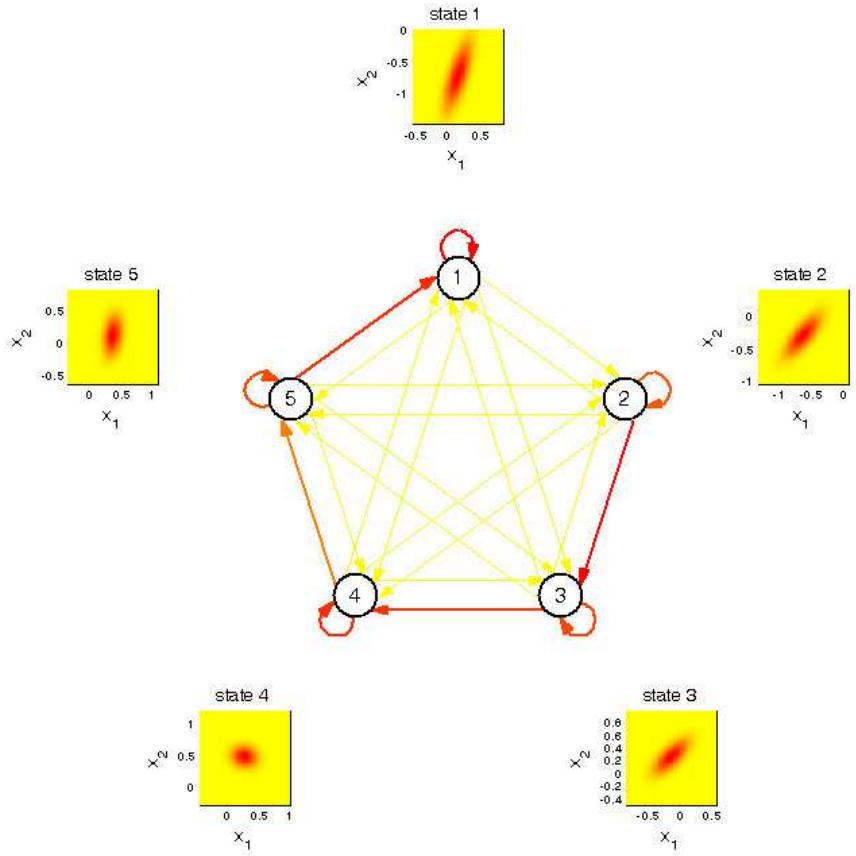


Fig. 7. Encoding of the dataset after preprocessing (see Figure 4) in a fully-connected HMM with 5 states. Each of the 5 states of the HMM outputs 2 distributions for the observable signals $\{x_1(t)\}$ and $\{x_2(t)\}$. The correlation across the signals is computed in the covariance matrices of the observable distributions attached to each state (tilted ellipses).

modality, we rescale those using a transfer function f,(see Figure 6). We then discard all the signals, such that $f(d_i, \sigma_i) < 0.1$

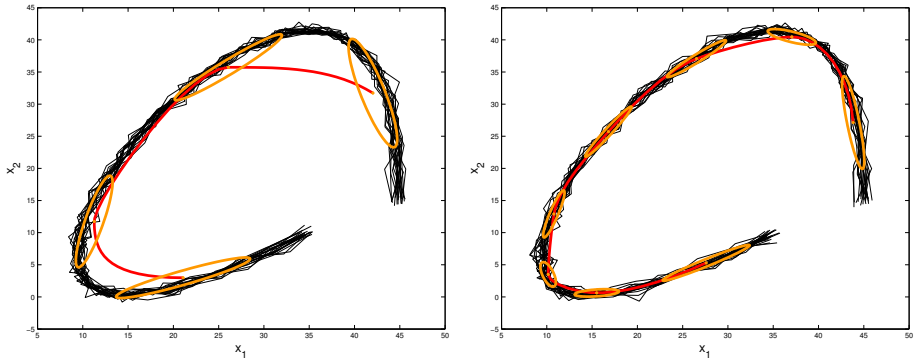


Fig. 8. Training data (thin lines) and generalized form of the data after retrieval (thick line), when encoding the data in a 4-states HMM (left) and in an 8-states HMM (right). The length of the axes of the ellipses (superimposed to the curves) correspond to the correlation factors learned by the HMM.

2.2 Probabilistic Data Encoding:

After preprocessing, the resulting signals are then encoded in a continuous fully-connected HMM with full covariance matrix. The HMM uses a set of parameters $\{\vec{\pi}, \mathbf{A}, \mathbf{B}\}$, representing, respectively, the initial states distribution, the states transition probabilities, and the observable distributions [2] . The distributions are continuous and modeled by a single Gaussian per state and output. That is, to each state $\{q(t)\}$ of the model is associated a distribution of observables with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, i.e. $\mathbf{B} = \{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}$.

Since the optimal number of states in the HMM may not be known beforehand, we use the *Bayesian Information Criterion* (BIC) criterion to select the optimal number of states for the model, by determining a trade-off between optimizing the model's likelihood (a measure of how well the model fits the data) and minimizing the parameters (i.e. the number of states used to encode the data).

$$BIC = -2\,\mathcal{L} + n_p\,log(T) \tag{3}$$

$\mathcal{L}$ is the log-likelihood of the model, given the observed dataset. n_p is the number of independent parameters in the HMM, and T the number of observation data used in fitting the model (in our case $T = K \cdot N$, for trajectories of size N). The first term of the equation measures how well the model fits the data, while the second term is a penalty factor that aims at keeping the total number of parameters low. In our experiments, we compute a set of candidate HMMs with up to 20 states and retain the model with the minimum score. Figure 8 shows the result of encoding data using a 4-state and a 8-state HMM (same data as in Figure 9, 4 and 7). The 8-state HMM fits better the data, but uses more parameters.

[2] People unfamiliar with HMM should refer to [21]

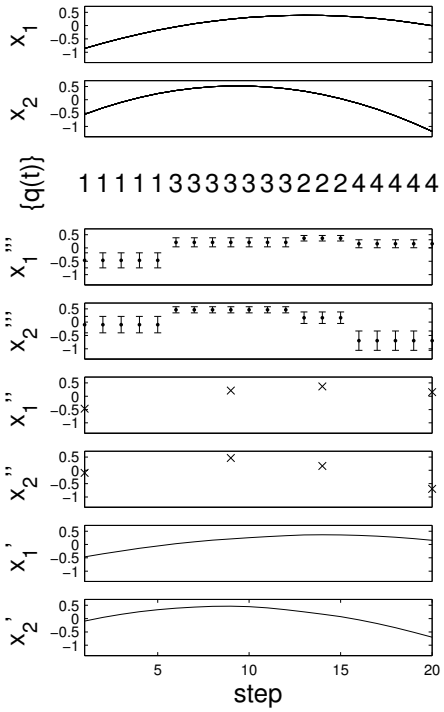


Fig. 9. Generalization and retrieval process, using the HMM representation of the data (see also Figure 8 and 7). Here a generated dataset of 2 signals is represented.

The transition probabilities $p(q(t)=j|q(t\text{-}1)=i)$ and the observation distributions $p(x(t)|q(t)=i)$ are estimated by the *Baum-Welch* algorithm, an *Expectation-Maximization* (EM) algorithm, that maximizes the likelihood that the training dataset can be generated by the HMM.

Once trained, the HMM can be use to recognize gestures (in our experiments, this is used to decide whether a new demonstration belongs or not to the same task). In order to measure the similarity between a new gesture and the ones encoded in the model, we run the *forward-algorithm*, an iterative procedure to estimate the likelihood that the observed data could have been generated by the model.

Once trained, the HMM can also be used to regenerate a generalized version of the gestures (in our experiments, this is used to generate a desired trajectory for the *Optimal Trajectory Generation* module). In order to regenerate a gesture from a given model, we reconstruct the optimal sequence of state transition using the *Viterbi algorithm*. Given this sequence of states, we retrieve a time-series of K variables $\{x_i'''(t)\}, i = 1, .., K$, by taking the mean values $\boldsymbol{\mu}$ of the Gaussian distributions for each observable (see Figure 9). We, then, extract a set of key-points $\{x_i''(t)\}$ from these time series. If there is a transition to a state n at time t_1 and if there is a transition to another state at time t_2, a key-point is created at the mean time $\frac{t_1+t_2}{2}$. By interpolating between these key-points and normalizing in time, we reconstruct the series of

observables $\{x'_i(t)\}$, using Piecewise cubic Hermite polynomial functions, see Figure 8). This gives us finally a set of desired trajectories $\vec{\theta}^d, \vec{x}^d\}$, which will then be used to compute the optimal trajectory for the imitator, as we will see next.

2.3 Determining the task constraints by a cost function

In [6], we proposed a general formalism for determining the cost function of an imitation task. The metric (or cost function) measures the quality of the reproduction, and, as such, drives the selection of an appropriate controller for the reproduction of the task.

Let $\{x_i(t)\}$ and $\{x'_i(t)\}$ with $i = 1, ...K$ be a set of demonstrated and a reproduced signals. Then, we define the global cost function H as:

$$H = \frac{1}{K} \sum_{i=1}^{K} w_i \cdot H_i(x_i, x'_i) \tag{4}$$

where $w_i \in [0, 1]$ are the weights measuring the relative importance of each signal. These are given by (see section 2.1):

$$w_i = f(d_i) \tag{5}$$

The cost function is bounded: $H \in [0, 1]$. H=0 corresponds to a perfect reproduction.

In the experiments reported here, we consider the particular case, where:

$$\begin{aligned} H(\vec{\theta}, \vec{\theta'}, \vec{x}, \vec{x'}) = & \frac{1}{2}(\vec{\theta} - \vec{\theta}^d)^T \mathbf{W}^\theta (\vec{\theta} - \vec{\theta}^d) \\ & + \frac{1}{2} (\vec{x} - \vec{x}^d)^T \mathbf{W}^x (\vec{x} - \vec{x}^d) \end{aligned} \tag{6}$$

Where $(\vec{\theta}, \vec{\theta}^d), (\vec{x}, \vec{x}^d)$ are, respectively, the current and desired positions of the joints and of the hand. $\mathbf{W}^\theta$ (4x4 matrix) and $\mathbf{W}^x$ (3x3 matrix) are the set of weights w_i associated, respectively, with the four joints trajectories and the 3D Cartesian path of the hand. By simplicity and for clarity, we will omit the vectorial notations for the rest of the developments.

Once the cost function and the relative influence of each constraint have been determined, we generate a trajectory that is optimal with respect to the cost function H. We must, however, take into account the body constraints of the robot, which are given by the inverse kinematics function $\theta = f^{-1}(x)$. Following [16], we consider an iterative, locally linear, solution to this equation, such that:

$$\dot{x} = \mathbf{J} \cdot \dot{\theta} \tag{7}$$

Where $\dot{\theta}(t) = \theta(t) - \theta(t-1)$ and $\dot{x}(t) = x(t) - x(t-1)$ are the velocity vectors of the joints and the hand path. $\mathbf{J}$ is the Jacobian, a $4x3$ matrix.

Similarly, by substituting $c(t) = \theta(t-1) - \theta^d(t)$ and $d(t) = x(t-1) - x^d(t)$ in (6), we have:

$$\begin{aligned} H(\dot{\theta}, \dot{x}) = & \frac{1}{2} \left(\dot{\theta} - c\right)^T \mathbf{W}^\theta \left(\dot{\theta} - c\right) \\ & + \frac{1}{2} \left(\dot{x} - d\right)^T \mathbf{W}^x \left(\dot{x} - d\right) \end{aligned} \tag{8}$$

The problem is now reduced to finding a minimum of (8) when subjected to (7). Since H is a quadratic function, the problem can be solved analytically by Lagrange optimization. We define the *Lagrangian* as:

$$\begin{aligned} L(\dot{\theta}, \lambda) = & \frac{1}{2} \left(\dot{\theta} - c\right)^T \mathbf{W}^\theta \left(\dot{\theta} - c\right) \\ & + \frac{1}{2} \left(\dot{x} - d\right)^T \mathbf{W}^x \left(\dot{x} - d\right) \\ & + \lambda^T (\dot{x} - \mathbf{J} \cdot \dot{\theta}) \end{aligned} \tag{9}$$

where λ is the vector of associated Lagrange multipliers and λ^T its transpose.

There are two necessary condition for $\dot{\theta}^*$ to be an optimum of H subjected to the condition given by (7):

$$\frac{\partial L(\dot{\theta}^*, \lambda)}{\partial \dot{\theta}} = 0; \tag{10}$$

$$\frac{\partial L(\dot{\theta}^*, \lambda)}{\partial \lambda} = 0; \tag{11}$$

When solving (11), we find again (7). When substituting $\dot{x}$ by $\mathbf{J} \cdot \dot{\theta}$ in (9), and, then, deriving along $\dot{\theta}$, we get:

$$W^{\theta}(\dot{\theta}^* + c) + \mathbf{J}^T(W^x(\mathbf{J}\dot{\theta}^* + d)) = 0 \tag{12}$$

where $\mathbf{J}^T$ is the transpose of $\mathbf{J}$.

Solving for $\dot{\theta}$:

$$\dot{\theta}^* = -(\mathbf{W}^{\theta} + \mathbf{J}^T\mathbf{W}^x\mathbf{J})^{-1}(\mathbf{J}^T\mathbf{W}^x d + \mathbf{W}^{\theta} c) \tag{13}$$

We can then recompute the joint trajectories, using $\theta(t) = \theta(t-1) + \dot{\theta}(t)$ and the hand path using $\dot{x}(t) = \mathbf{J}\dot{\theta}(t)$ and $x(t) = x(t-1) + \dot{x}(t)$.

3 Experiments

We conducted a set of three experiments to demonstrate the validity of our model for teaching a humanoid robot simple uni-manual and bi-manual manipulatory tasks. The tasks consisted in "stirring the fondue", "hammering a nail" and "grabbing a ball with two hands". In each case, the robot was shown the task five times. Small variations in the demonstrations were introduced to let the key features of the task (i.e. the invariants) be more salient. The model, then, extracted the relative importance of each variable (i.e. hand path, joint trajectories, position of the hand with respect to the objects) and constructed the weights of the cost function accordingly, see Section 2.3. Note that the weights for the position of the hand relative to the objects were used to determine which of the absolute or relative task description one should choose, by picking the representation with maximal weights.

Then, the robot was required to reproduce each task in different locations in space, by displacing the object to be manipulated. This procedure aimed at demonstrating the robustness of the system when the constraints were

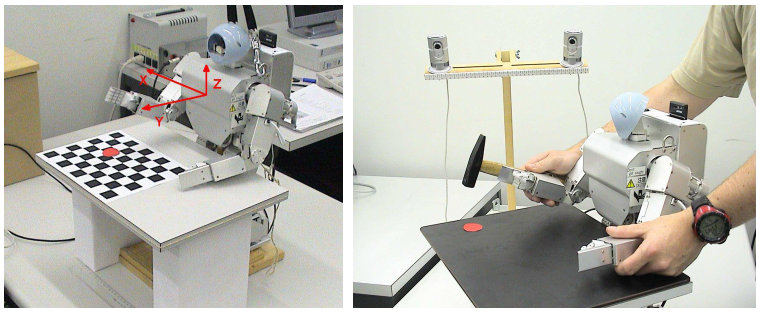


Fig. 10. Experimental Setup: **Left:** The robot stands upright in front of a table. A color-based stereoscopic vision system tracks the 3D-position of the different objects. **Right:** The robot is being taught how to hammer a nail through kinesthetics, i.e. by recording the kinematics of the joint trajectories while the demonstrator moves the robot's arms through each of the task's steps.

transposed to different locations within the robot's workspace. Would the optimal solution change? If yes, would the new trajectory remain stable and smooth?

4 Experimental setup

The experiments were conducted with a Fujitsu HOAP-2 humanoid robot with 25 degrees of freedom (DOF), of which only the 8 DOFs of the two arms were required in the experiments. The remaining DOFs of the torso and legs were set to a constant position, so as to support the robot in an upright posture, facing a table, see Figure 10.

A color-based stereoscopic vision system tracks the 3D-position of the different objects used in the experiments at a rate of 15Hz, with an accuracy of 10 mm. The system uses two Phillips web-cams with a resolution of 320x240 pixels. The tracking is based on color segmentation for detecting the skin color and the color of the different objects in the YCbCr color space [3].

In the experiments reported here, the robot was taught through kinesthetics, i.e. by the demonstrator moving its two arms through each of the task's steps, see Figure 10. To achieve this, the robot's motors were set in a passive mode, whereby each limb could be backdrived by the human demonstrator. The kinematics of each joint motions was recorded at a rate of 1000Hz during the demonstration and was, then, downrated to 15Hz to match the tracking rate of the objects.

[3] Only Cb and Cr are used, to be robust to changes in luminosity

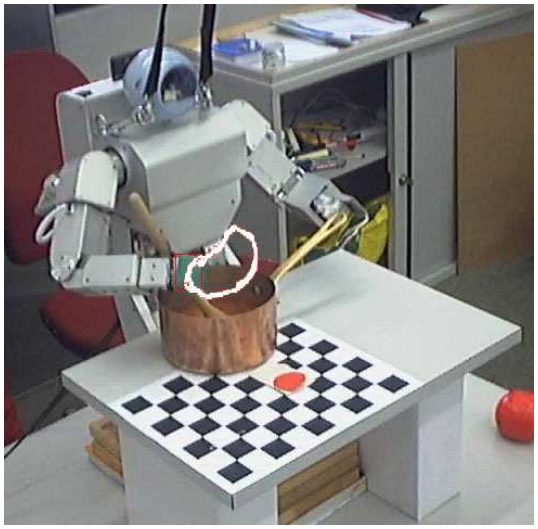


Fig. 11. The trajectory in white corresponds to the optimal reproduction of the gesture "stirring the Fondue", i.e. the solution leading to the lowest value of the cost function.

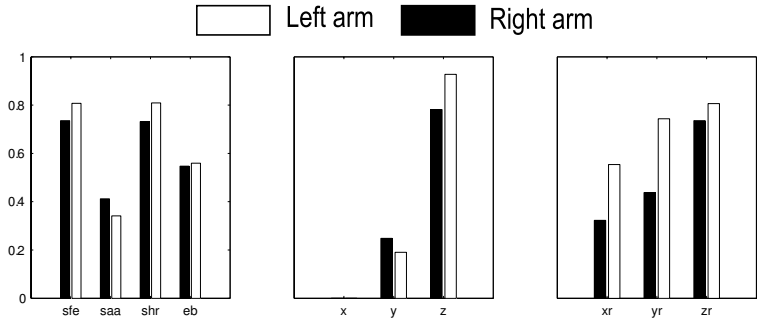


Fig. 12. Weights extracted after five demonstration of "stirring the Fondue". SFE: Shoulder Flexion-Extension, SAA: Shoulder Abduction-Adduction, SHR: Shoulder Humeral Rotation, EB: Elbow. x,y,z: Absolute position of the hand. xr,yr,zr: relative position of the hand with respect to the saucepan.

4.1 "Stirring the Fondue"

The first experiment consisted in teaching the robot how to "stir a fondue" (typical swiss meal), i.e. it had to learn to hold stiff the saucepan with the left arm and to perform cyclic rotational motions with its right arm, while keeping the stick inside the saucepan, see Figure 11. The robot was shown five demonstrations, during which the demonstrator made sure to keep the same rotational motions, while displacing the position of the saucepan on the table.

Figure 12 shows the values taken by the weights computed by the *determining the task constraints* module, see Section 2.3. As expected, the weights associated with the joints are very important (reflecting their relative invariance in the task), except for SAA (which varied more than the other DOFs, as an

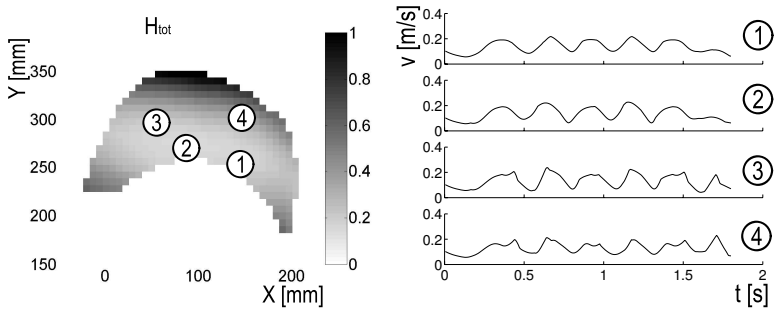


Fig. 13. **Left:** map of the values taken by the cost function H as a function of the position of the sauce pan on the table in the task "stirring the fondue". **Right:** Variation of the amplitude of the speed vector of the right hand along time.

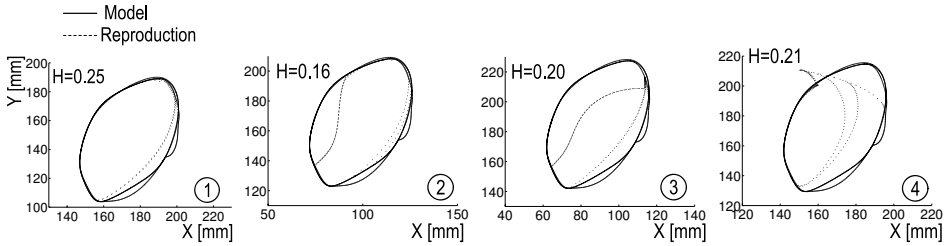


Fig. 14. (Straight Line:) Trajectory followed by the hand path when the saucepan is located at positions 1, 2, 3 and 4 in the workspace, see Figure 13, superimposed to the demonstrated trajectory (thin line). Since the hand path is the criterion with lowest importance, we observe large distortions, while the corresponding value of the cost function remain low, see Figure 13.

effect of displacing the saucepan across the table during the demonstrations). The absolute hand path is on the other hand much more variable, except along z (i.e. no vertical motion). Note that the saucepan was displaced principally along the x axis rather than along the y axis, to ensure that all locations would be reachable by the robot (thus, leading to a larger weight for y than for x). Similarly, the relative position of the hand with respect to the saucepan was pretty invariant, leading to very high weights for these features.

Figure 13 shows the values taken by the cost function H when estimating the optimal trajectory, by moving systematically the location of the saucepan on the table. The white areas correspond to the locations where there are no solutions (i.e. areas not reachable by the robot). As the saucepan is moved across the table, the robot must adapt its posture to satisfy best the different constraints, such as, e.g., remaining within its joint limits. This results in important distortions of the hand path (whose reproduction has lowest importance according to the cost function) at locations 2 and 3 (that reach out of the joint limits), see Figure 14, while the cost function at the same locations remain very low, see Figure 13.

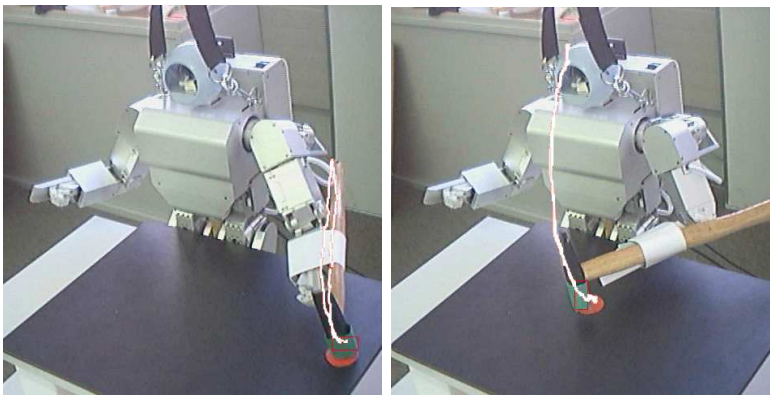


Fig. 15. Reproduction of the movement "hammering a nail" at 2 different locations in the task workspace.

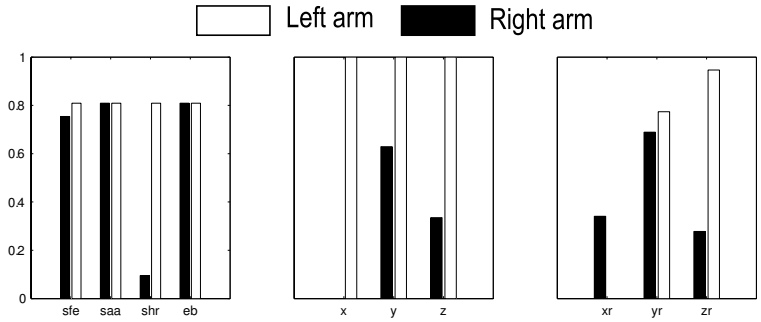


Fig. 16. Weights extracted after five demonstrations of "hammering a nail" while keeping the left arm in the same position.

4.2 Hammering a nail

In the second experiment, the task consisted in "hammering a nail", see Figure 10 right, i.e. to let the right arm slam down on a color dot placed on the table. In order to demonstrate the sensitivity of our system to the experimental conditions, we conducted two sets of experiments. In the first set, the robot was shown the gesture with the right arm only, leaving the left arm in a rest position. In the second set, the left arm was moved randomly, during each of the five demonstrations.

Figures 16 and 17 show the values of the weights extracted in each of the two conditions. In the first condition, see Fig. 16, the system has found a correlation along the y and z axes between the left arm (static) and the nail. That is due to the fact that, during the demonstration, the "nail" was moved mainly along the x axis. In contrast, in the second condition, no such correlation was found, since the left arm was moved randomly, see Fig. 17.

Once trained, the robot was, then, requested to reproduce the motion at different locations on the table.

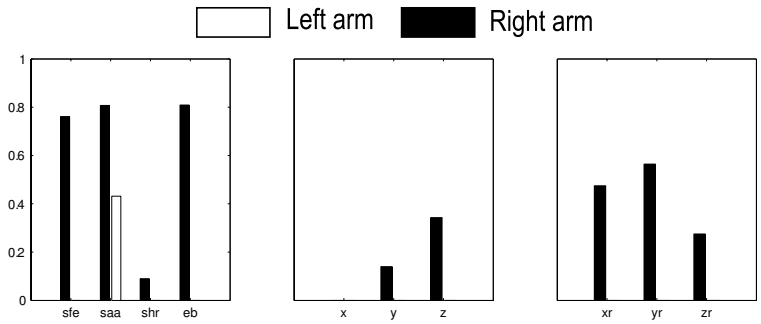


Fig. 17. Weights extracted after five demonstrations of "hammering a nail" while setting the left arm to a random position at the start of each demonstration.

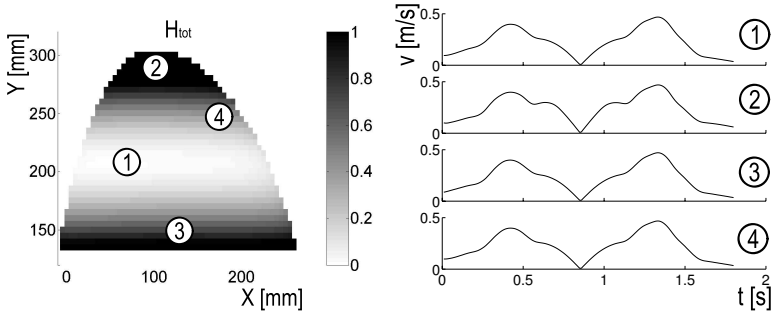


Fig. 18. **Left:** map of the values taken by the cost function H as a function of the position of the "nail" on the table in the task "hammering a nail" (first condition: left arm static). **Right:** Variation of the amplitude of the speed vector of the right hand along time.

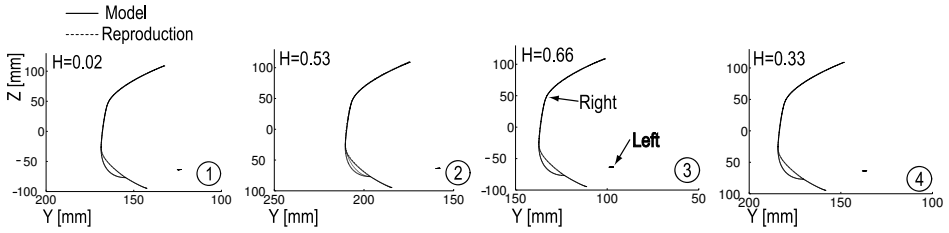


Fig. 19. Trajectory followed by the hand path when the "nail" is located at positions 1, 2, 3 and 4 in the workspace, see Figure 18, superimposed to the demonstrated trajectory in the task "hammering a nail".

As expected, the optimal value of the cost function H increases when the target approaches the limits of the workspace, i.e. when the constraints on the position of the hand and joints can no longer be fulfilled, see Fig.18 and Fig.20. The optimal trajectory, i.e. the optimum of the cost function, is reached in the middle of the workspace.

Figures 19 and 21 show the optimal trajectories, i.e. the trajectory corresponding to the minimum of the cost function, for each of the two conditions. We observe that, after training in the first condition, the left arm moves along y with the target during reproduction, in order to satisfy the erroneous relation

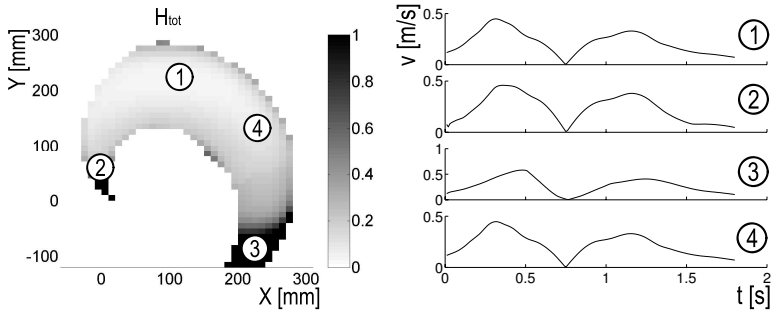


Fig. 20. **Left:** map of the values taken by the cost function H as a function of the position of the "nail" on the table in the task "hammering a nail" (second condition: left arm random). **Right:** Variation of the amplitude of the speed vector of the right hand along time.

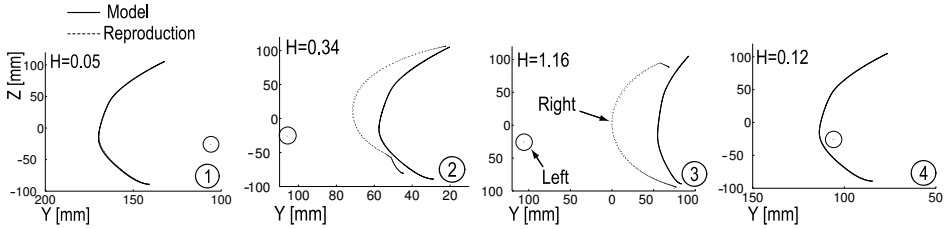


Fig. 21. Trajectory followed by the hand path when the nail is located at positions 1, 2, 3 and 4 in the workspace (see Fig 20), superimposed to the demonstrated trajectory.

between the target position and that of the left arm, see Figure 19. Thus, when the target comes too close to the front of the robot, the left arm is blocked and the system no longer finds a plausible solution. Such constraints are inexistent in the second training condition and, thus, ones does not observe the same limitation (compare the valid areas of the cost function for $y < 150$ in Figures 18 and 20).

We, also, note that in all four examples the reproduced hand path fits very closely the desired hand path, even when the "nail" is at the limit of the task workspace. This is due to the fact that the workspace of the task, in the first condition, is now at the intersection between the workspaces of the two arms. Thus, when the left arm is blocked at the limit of the task workspace, the right arm can still reproduce the movement because its workspace is not limited in the same way.

In the second condition(Fig. 21), the valid workspace depends only on the right arm (since the left arm is static). Thus, at the extreme parts of the workspace (trajectory 2 and 3), the reproduced hand path no longer matches the demonstrated one.

Fig. 22. HOAP2 is grabbing a ball with to hands

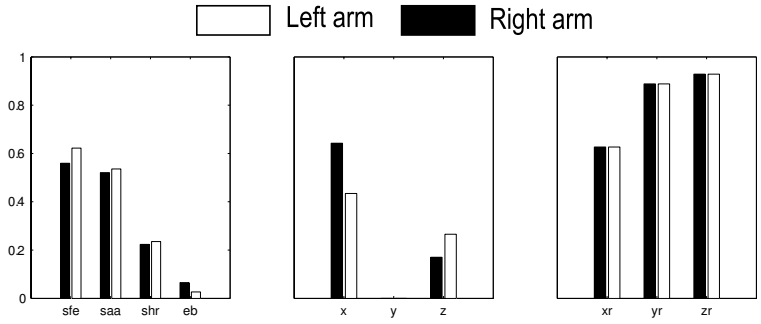


Fig. 23. Weights extracted after five demonstrations of the task "Grabbing a ball with 2 hands"

4.3 Grabbing a ball with two hands

This experiment consisted in grasping a ball with two hands and lifting it up over a short distance, see Figure 22. During training and testing, the location of the target (the ball) was varied along a vertical plan perpendicular to the HOAP2's torso. While, in the first two experiments, we had tested our system with static targets, we here consider a task, composed of two parts. In the first part, the object is static, while in the second part, it moves together with the two hands. This creates a strong correlation between the hand path and the object path, which is correctly extracted by the weights, see Figure 23. Note that, because we compute the weights by averaging over the whole trajectory, we loose the information that this correlation applies only to the second part of the task. In future work, we will extend the metric to make the weights time dependent, i.e. to replace $\mathbf{W}^\theta, \mathbf{W}^x$ in Equation 6 by $\mathbf{W}^\theta(t)$ and $\mathbf{W}^x(t)$.

However, this limitation does not prevent us in this particular task to obtain good performance during testing. Figure 24 shows that, once again, the best area to reproduce the movement is in front of the robot at about half the height (around $z = -100$) covered during testing. Figure 25 displays four trajectories reproduced in different parts of the task workspace. In each case, the reproduction is qualitatively good and we observe only very small distortions of the hand path at the edge of the workspace. In other words, in each case, the goals of the task, i.e. grabbing the ball and lifting it, are satisfied. When

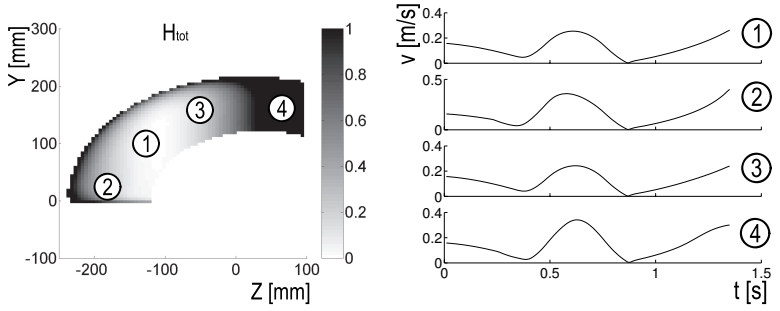


Fig. 24. **Left:** map of the values taken by the cost function H as a function of the position of the "ball" on the table in the task "Grabbing a ball with 2 hands" (first condition: left arm static). **Right:** Variation of the amplitude of the speed vector of the right hand along time.

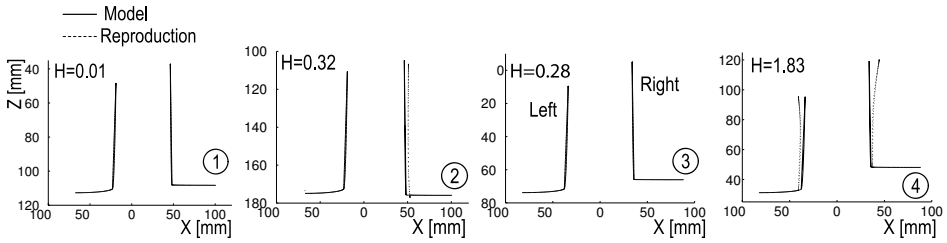


Fig. 25. Trajectory followed by the hand path when the ball is located at positions 1, 2, 3 and 4 in the workspace, see Figure (See Fig 24), superimposed to the demonstrated trajectory.

reaching the limits of its joints, the robot adapted the trajectory of its two arms simultaneously, resulting in a slight shift from the original trajectory, while still keeping the ball firm in its grasp.

5 Discussion and Conclusion

This paper presented a method to 1) extract the important features of a given task, 2) to determine a generic cost function to evaluate the robot's performance, and, finally, 3) to optimize the robot's reproduction of the task in a new context based on this cost function. The method was validated in three sets of experiments that set different constraints. The tasks consisted in either reproducing the gesture (stirring a stick in a saucepan), the path of the end effectors relative to the object (hammering a nail) or the path of the effectors relative to one another (grasping a ball with 2 hands).

We showed that, in each case, the robot managed to adapt its motions "cleverly", so as to reproduce correctly the important qualitative features of each task (i.e. keeping the stick in the saucepan, hitting the nail, keeping the ball firmly in its grasp). However, none of these high-level goals were explicitly

represented in the robot's control system, but, were nevertheless correctly extracted by our probabilistic system.

Note, however, that this statistical method has limitations. Because the relative importance of each task constraint is computed over the whole demonstration, it blurs out changes in the relative importance of the constraints that may appear over the course of the demonstrations (as in the example of grasping the ball and lifting it). These changes could, easily, be determined, by looking at the variability of each state of the HMM separately. Thus, in further work, we will extend the method to take into account time-varying constraints. Moreover, this will be used to determine transitions across subtasks, a subtask consisting of a region of the cost function with equal value for all constraints. We will also investigate ways of setting priors on the weights, so that the system could converge faster to a task description, when two tasks are very similar.

The system we presented for solving the "how to imitate" issue is generic, in the sense that it makes no assumption on the robot's configuration (number of degrees of freedom and length of segments). Moreover, it produces smooth trajectories, even at the limits of the robot's workspace, which makes it a suitable robot controller.

As first stressed out by Nehaniv and colleagues [18], there is a multitude of correspondence problems, when trying to transfer skills across various agents and situations. One may consider

- **Different embodiments:** the demonstrator does not share the same morphology and characteristics as the imitator.
- **Different situations:** the environment and constraints during the demonstration and the reproduction are different

In the experiments we presented in this paper, the robot was being taught through kinesthetics. By showing kinesthetically how to perform a task, the user "embodies" the robot's body. Thus, this way, we simplified the correspondence problem and overlooked the problem of having different embodiements. However, by testing the system in different situations than those taught, we tackled the second aspect of the correspondence problem.

Note that, in these experiments reported here, we assumed implicitly that the kinematics of joint angle trajectories and hand path is sufficient to describe the skill, and that dynamics is of less importance. This may not be true and taking into account the forces applied on the object may certainly be very important in certain task. However, it is very likely that these are not learned through imitation, but, through more generic motor learning processes.

Note that the correspondence problem or "how to imitate" question relates,

also, to a more generic issue of imitation, that of defining a common representation for sharing the information across various modalities [10, 20, 22, 13], which we did no address here. Indeed, in the present work, we assumed that both agents would have access to very low-level information, namely to the position of each joint and of the end-effector in real time. Such an assumption is certainly not valid in all scenarios and does not account for human capacity to imitate.

References

[1] A. Alissandrakis, C.L Nehaniv, and K. Dautenhahn. Imitating with alice: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 32:4:482–496, 2002.

[2] A. Alissandrakis, C.L Nehaniv, and K. Dautenhahn. Towards robot cultures? - learning to imitate in a robotic arm test-bed with dissimilar embodied agents. *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, 5:1:3–44, 2004.

[3] T. Asfour and R. Dillman. Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In *IEEE/RSJ Intl Conference on intelligent Robots and Systems*, 2003.

[4] P. Bakker and Y. Kuniyoshi. Robot see, robot do : An overview of robot imitation. In *AISB Workshop on Learning in Robots and Animals, Brighton, UK*, April 1996.

[5] T. Belpaeme, B. de Boer, and B. Jansen. The role of population dynamics in imitation. In *In Dautenhahn, K. and Nehaniv, C.L. (eds.) Imitation and Social Learning in Robots, Humans and Animals: Behavioural, Social and Communicative Dimensions. Cambridge University Press. Cambridge, UK.* MIT Press, 2004. To appear.

[6] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics & Autonomous Systems*, 47:2-3:69–77, 2004.

[7] S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[8] S. Calinon and A. Billard. Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm. In *Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany*, 2005.

[9] S. Calinon, F. Guenter, and A. Billard. Goal-directed imitation. In *Proceedings of the International Conference on Robotics and Automation, Barcelona.*, 2005.

[10] J. Demiris and G. Hayes. Imitation as a dual-route process featuring pre-

dictive and learning components: A biologically-plausible computational model. In C. Nehaniv and K. Dautenhahn, editors, *Imitation in Animals and Artifacs*. MIT Press, 2001 (In Press).

[11] A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *IEEE/RSJ Intl Conference on Intelligent Robots and Systems*, 2001.

[12] M. Ehrenmann, O. Rogalla, R. Zoellner, and R. Dillmann. Teaching service robots complex tasks: Programming by demonstation for workshop and household environments. In *Proc. of the IEEE Int. Conf. on Field and Service Robotics. (FRS), Finnland*, 2001.

[13] M. Ito and J. Tani. On-line imitative interaction with a humanoid robot using a dynamic neural network model of a mirror system. *Adaptive Behavior*, 12:2:93–115, 2004.

[14] B. Jansen and T. Belpaeme. Goal-directed imitation through repeated trial-and-error interactions between agents. *Robotics & Autonomous Systems*, 2005. Submitted.

[15] M. Kaiser and R. Dillmann. Building elementary robot skills from human demonstration. In *Proceedings. International Conference on Robotics and Automation*, volume 3, 1996.

[16] A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 7 (12), pp. 868-871*, 1977.

[17] C. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, *Learning Robots: An Interdisciplinary Approach*. World Scientific Press, 1999.

[18] C. L. Nehaniv. Nine billion correspondence problems and some methods for solving them. In *Proceedings of the Second International Symposium on Imitation in Animals & Artifacts, The Society for the Study of Artificial Intelligence and Simulation of Behaviour*, pages 93–95, 2003.

[19] E. Oyama, A. Agah, K. MacDorman, T. Maeda, and S. Tachi. A modular neural network architecture for inverse kinematics model learning. *Neurocomputing*, 2001.

[20] E. Oztop and M. A. Arbib. Schema design and implementation of the grasp-related mirror neuron system. *Biological Vybernetics*, 2002. In Press.

[21] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:2:257–285, February 1989.

[22] E. Sauser and A. Billard. Three dimensional frames of references transformations using recurrent populations of neurons. *Neurocomputing*, 64:5–24, 2004.

[23] M. Skubic and R.A. Volz. Acquiring robust, force-based assembly skills from human demonstration. In *IEEE Transactions on Robotics and Automation*, volume 16:6, pages 772 –781, 2000.

[24] G. Tevatia and S.Schaal. Inverse kinematics for humanoid robots. In

IEEE Intl Conference on Robotics and Automation, 2000.

[25] X. Wang and J.P. Verriest. A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation. *The Journal of Visualization and Computer Animation, vol. 9, pp. 33-47*, 1998.

[26] M. Yeasin and S. Chaudhuri. Toward automatic robot programming: learning human skill from visual data. In *IEEE Transactions on Systems, Man and Cybernetics, Part B*, volume 30:1, 2000.

[27] J Zhang and B Roessler. Self-valuing learning and generalization with application in visually guided grasping of complex objects. *Robotics & Autonomous Systems*, 47:2-3:772 –781, 2004.

To appear in: Dautenhahn and Nehaniv, editors,
Imitation and Social Learning in Robots, Humans and Animals:
Behavioural, Social and Communicative Dimensions.
Cambridge University Press, 2005. In Press.

Learning of Gestures by Imitation in a Humanoid Robot

Sylvain Calinon and Aude Billard

Autonomous Systems Lab
Swiss Federal Institute of Technology Lausanne - EPFL
CH-1015 Lausanne, Switzerland
{sylvain.calinon,aude.billard}@epfl.ch
http://asl.epfl.ch/

Keywords: Imitation Learning, PCA, HMM, Humanoid Robots, Programming by Demonstration

1 Introduction

Traditionally, robotics developed highly specific controllers for the robot to perform a specific set of tasks in highly constrained and deterministic environments. This required to embed the controller with an extensive knowledge of the robot's architecture and of its environment. It was soon clear that such an approach would not scale up for controlling robots with multiple degrees of freedom, working in highly variable environments, such as humanoid robots required to interact with humans in their daily environment.

The field has now moved to developing more flexible and adaptive control systems, so that the robot would no longer be dedicated to a single task, and could be re-programmed in a fast and efficient manner, to match the end-user needs.

Robot learning by imitation, also referred to as *robot programming by demonstration*, explores novel means of implicitly teaching a robot new motor skills [5, 10, 20]. This field of research takes inspiration in a large and interdisciplinary body of literature on imitation learning, drawing from studies in Psychology, Ethology and the Neurosciences [9, 4, 1]. To provide a robot with the ability to imitate is advantageous for at least two reasons: it provides a natural, user-friendly means of implicitly programming the robot; it constrains the search space of motor learning by showing possible and/or optimal solutions.

In this chapter, we explore the issue of recognizing, generalizing and reproducing arbitrary gesture [3]. In order to take a general stance toward gesture recognition and reproduction, we address one major and generic issue, namely *how to discover the essence of a gesture*, i.e. how to find a representation of the data that encapsulates only the key aspects of the gesture, and discards the intrinsic variability across people motion.

To illustrate the idea, consider the following examples: when asked to imitate someone writing letters of the alphabet on a board, you will find it sufficient to only track the trajectory followed by the demonstrator's hand on the board. In contrast, when learning to play tennis, you will find it more important to follow

the trajectories of the demonstrator's arm joint angles, rather than the position of the hand (the ball's position varying importantly over the trials). Choosing, in advance, the optimal representation of the data (whether hand-path or joint angles) would greatly simplify the analysis and speed up learning.

In the application presented in this chapter, the robot is endowed with numerous sensors enabling it to track faithfully the demonstrator's motions. Some of the data gathered by the sensors are redundant and correlated. The first stage of processing performed on our data consists in applying Principal Component Analysis (PCA) in order to determine a space in which the data are decorrelated, and, consequently, to reduce the dimensionality of the dataset, so as to make the analysis more tractable.

In order for the robot to learn new skills by imitation, it must be endowed with the ability to generalize over multiple demonstrations. To achieve this, the robot must encode multivariate time-dependent datasets in an efficient way. One of the major difficulty in learning, recognizing and reproducing sequential patterns of motion is to deal simultaneously with the variations in the data and with the variations in the sequential structure of these data. The second stage of processing of our model uses Hidden Markov Models (HMMs) to encode the sequential patterns of motion in stochastic finite state automata. The motion is then represented as a sequence of states, where each state has an underlying description of multi-dimensional data (see Figure 4). The system takes inspiration in a recent trend of research that aims at defining a formal mathematical framework for imitation learning [19, 15, 3]. We present an implementation of these approaches in a noisy real-world application.

The remaining of this chapter is divided as follows: section 2 presents the experimental set-up. Section 3 describes in details the model. Results are presented in Section 4, and discussed in Section 5, stressing out the parallels existing between our robotic model and theoretical models of imitation learning in animals.

2 Experimental Set-up

Data used for training the robot have been generated by eight healthy volunteers (students at the EPFL School of Engineering). Subjects have been asked to imitate a set of 6 motions performed by a human demonstrator in a video. The motions consist in:

- Knocking on a door
- Raising a glass, drinking, and putting it back on a table
- Waving goodbye
- Drawing the stylized alphabet letters A, B and C

The subject's gestures have been recorded by 3 x-sens motion sensors, attached to the torso and the right upper- and lower-arm. Each sensor provides the 3D absolute orientation of each segment, by integrating the 3D rate-of-turn, acceleration and earth-magnetic field, at a rate of 100Hz. The joint angle trajectories of the shoulder joint (3 degrees of freedom (DOFs)) and of the elbow

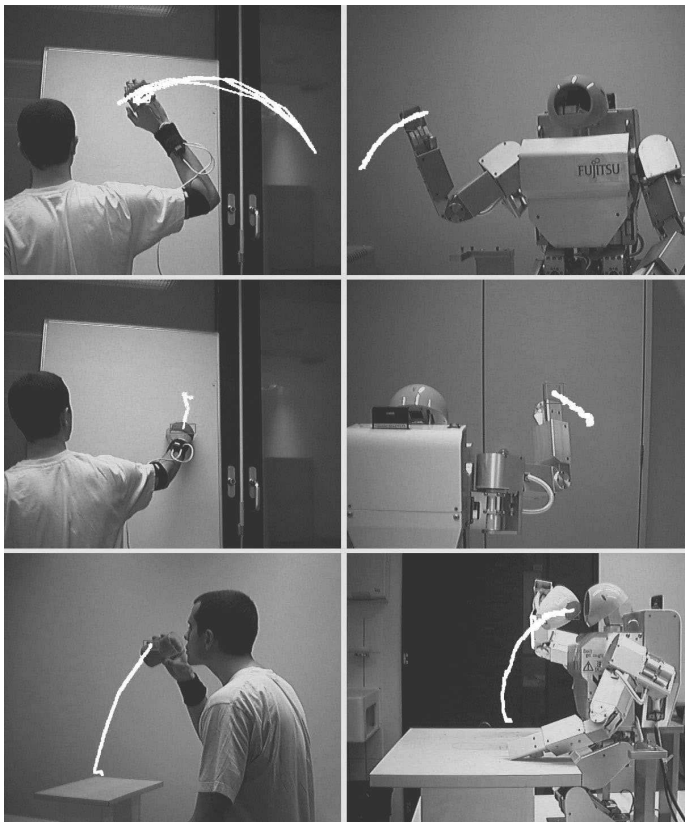


Fig. 1. Demonstration (*left column*) and reproduction (*right column*) of different tasks: waving goodbye (*1st line*), knocking on a door (*2nd line*), and drinking (*3rd line*). Three gyroscopic motion tracking sensors are attached on the upper arm, lower arm and torso of the demonstrator. The trajectories of the demonstrator's hand, reconstructed by the stereoscopic vision system, are superimposed to the image.

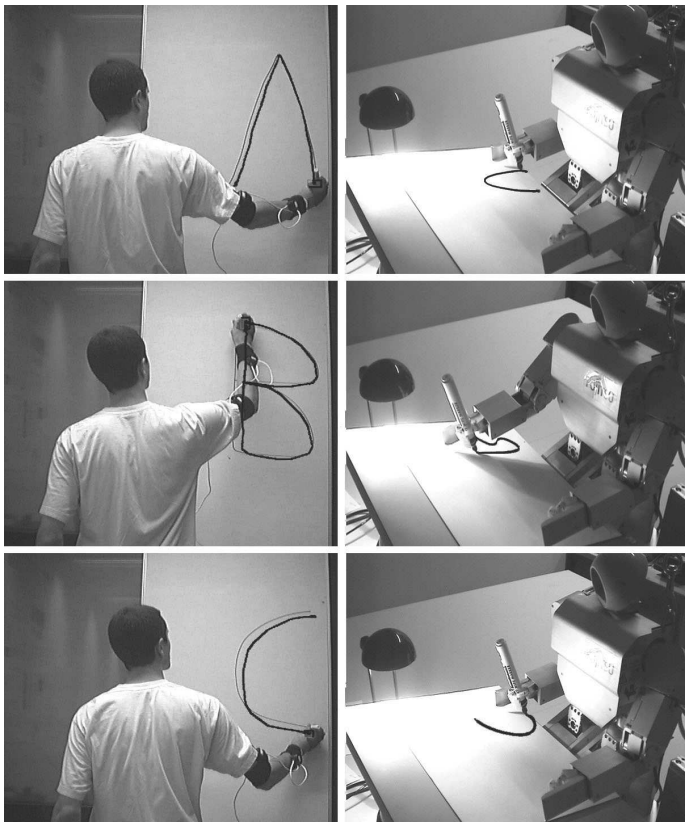


Fig. 2. Demonstration (*left column*) and reproduction (*right column*) of drawing the three stylized alphabet letters A, B and C. The motion reproduced by the robot follows a trajectory generalized across the different demonstrations. As the trajectory is projected by PCA in 2 dimensions, the letters can be written on a different plane.

(1 DOF) are reconstructed with a precision of 1.5 degrees, taking the torso as reference. These sensors provide a motor representation of the gesture, that can be used without major modification to control the robot.

A color-based stereoscopic vision system tracks the 3D-position of a marker placed on the demonstrator's hand, at a rate of 15Hz, with a precision of 10 mm. The system uses 2 Phillips webcams with a resolution of 320x240 pixels. The tracking is based on color segmentation in the YCbCr color space (Y is dismissed to be robust to changes in luminosity).

The robot is a Fujitsu humanoid robot HOAP-2 with 25 DOFs. In the experiments reported here, only the robot's right arm (4 DOFs) is used for the task. The torso and legs are set to a constant and stable position, in order to support the robot's standing-up.

3 Data processing

The complete dataset consists of the trajectories of 4 joint angles and the 3-dimensional trajectory of the hand in Cartesian space. Figure 3 shows a schematic of the sensory-motor flow. The data are first projected onto an uncorrelated, low-dimensional subspace, using PCA. The resulting signals are, then, encoded in a set of Hidden Markov Models. A generalized form of the signals is then reconstructed by interpolating across the time series output by the HMMs and reprojecting onto the original space of the data using the PCA eigenvectors.

For each experiment, the dataset is split in two. The first half is used to train the model, and the second half to test the model.

Let $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$ be the hand path in Cartesian space, and $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3, \boldsymbol{\Theta}_4\}$ the joint angle trajectories of the right arm, after interpolation, normalization in time (same number of data for each time series), and shifting such as the first data points coincide.

3.1 Preprocessing by Principal Components Analysis (PCA)

PCA is a technique used extensively to discover and reduce the dimensionality of a dataset. In this work, we use it to find a suitable representation of our multivariate dataset [14]. PCA consists in determining the directions (*eigenvectors*) along which the variability of the data is maximal. It assumes that the data are linear and normally distributed.

By projecting the data onto the referential defined by the eigenvectors of the correlation matrix, one obtains a representation of the dataset that minimizes the statistical dependence across the data. Consecutively, dimensionality reduction can be achieved by discarding the dimensions along which the variance of the data is smaller than a criterion. This provides a way to compress the data without losing much information and simplifying the representation.

PCA is applied separately to $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$ and $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3, \boldsymbol{\Theta}_4\}$ to determine if a better representation in each dataset can be used. The means $\{\boldsymbol{m}_1^X, \boldsymbol{m}_2^X, \boldsymbol{m}_3^X\}$ and $\{\boldsymbol{m}_1^\Theta, \boldsymbol{m}_2^\Theta, \boldsymbol{m}_3^\Theta, \boldsymbol{m}_4^\Theta\}$ are subtracted for each dimension.

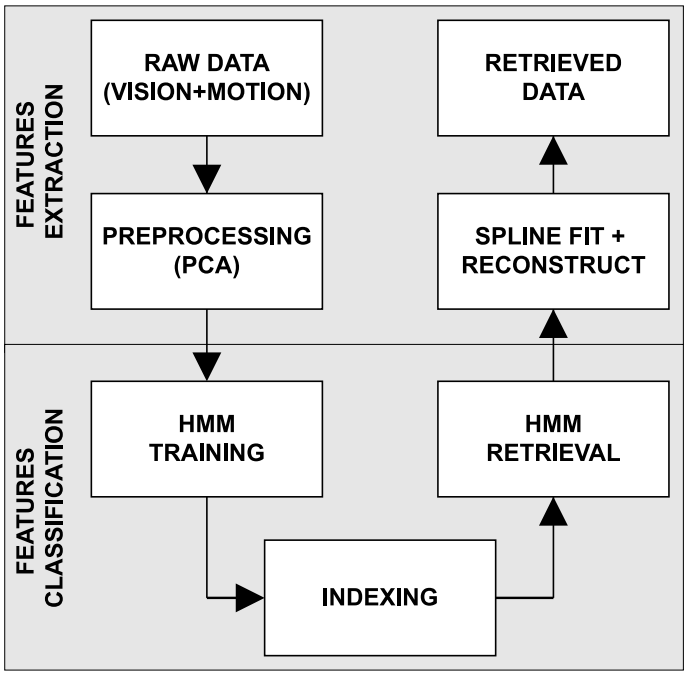


Fig. 3. Schematic of the sensory-motor flow: the data are first projected onto an uncorrelated, low-dimensional subspace, using PCA. The resulting signals are, then, encoded in a set of HMMs. A generalized form of the signals is then reconstructed by interpolating across the time series output by the HMMs and reprojecting onto the original space of the data using the PCA eigenvectors.

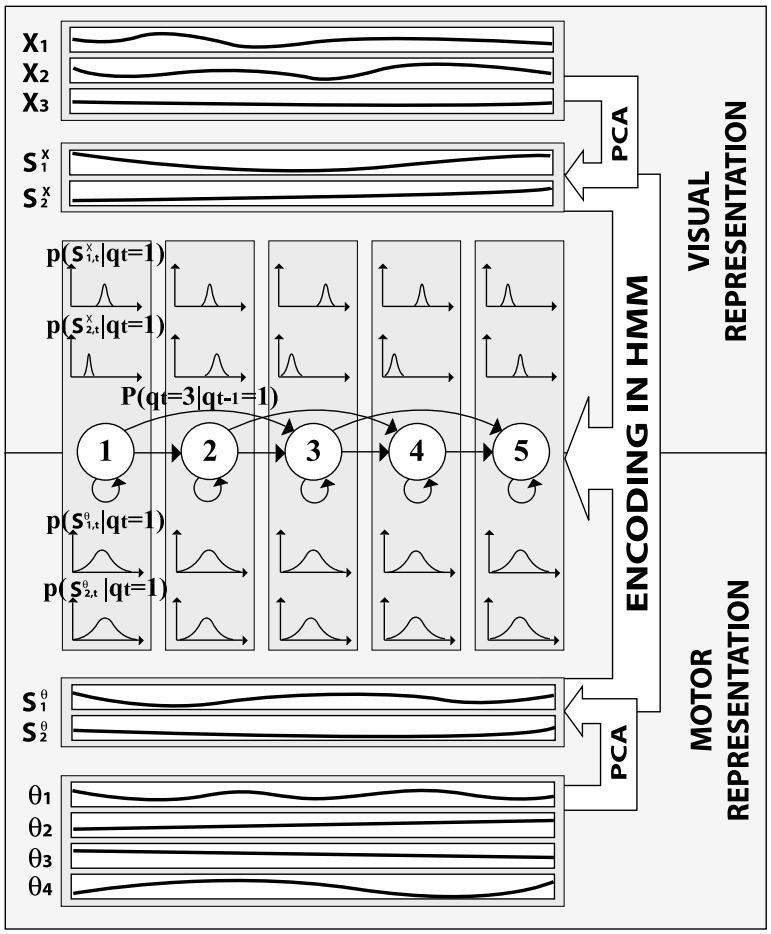


Fig. 4. Encoding of the hand path in Cartesian space $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$ and joint angles trajectories $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3, \boldsymbol{\Theta}_4\}$ in a HMM. The data are pre-processed by PCA, and the resulting signals $\{\mathbf{S}_1^X, \mathbf{S}_2^X, \ldots, \mathbf{S}_I^X\}$ and $\{\mathbf{S}_1^\Theta, \mathbf{S}_2^\Theta, \ldots, \mathbf{S}_J^\Theta\}$ are learned by the HMM. The data are represented as sequences of states, with transition probabilities between the states (not all the transitions are depicted). Each state in the HMM outputs multivariate data, represented by Gaussian functions.

Then, the 3 eigenvectors $\{\boldsymbol{v}_1^X, \boldsymbol{v}_2^X, \boldsymbol{v}_3^X\}$ and associated eigenvalues $\{e_1^X, e_2^X, e_3^X\}$ are calculated for the hand path. The 4 eigenvectors $\{\boldsymbol{v}_1^\Theta, \boldsymbol{v}_2^\Theta, \boldsymbol{v}_3^\Theta, \boldsymbol{v}_4^\Theta\}$ and associated eigenvalues $\{e_1^\Theta, e_2^\Theta, e_3^\Theta, e_4^\Theta\}$ are calculated for the joint angle dataset. An indication of the relative importance of each direction is given by its eigenvalue. Let I and J be the number of eigenvectors required to obtain a satisfying representation of $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$ and $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3, \boldsymbol{\Theta}_4\}$, such that the information lost by projecting the data onto these eigenvector is small. By expressing these datasets in their new basis, given respectively by I and J eigenvectors, the time series become $\{\mathbf{S}_1^X, \mathbf{S}_2^X, \ldots, \mathbf{S}_I^X\}$ with $I \leq 3$ to represent the hand path, and $\{\mathbf{S}_1^\Theta, \mathbf{S}_2^\Theta, \ldots, \mathbf{S}_J^\Theta\}$ with $J \leq 4$ to represent the joint angle trajectories. The selection criterion is to retain the first K components that cover over 80% of data's spread, i.e. to have $\sum_{i=1}^{K} e_i > 0.8$.

Applying PCA before encoding the data in a HMM has the following advantages:

- It helps reducing noise, as the noise is now encapsulated in the lower dimensions.
- It reduces the dimensionality of the dataset, which reduces the number of parameters in the Hidden Markov Models, and makes the training process faster.
- It produces a representation of the dataset that is easier to handle, and that can be used under different conditions, for the reproduction of the task (e.g. learning to write an alphabet letter on a plane, and writing it on another plane).

For example, while drawing an alphabet letter, the dimensionality of the 3D Cartesian path can be reduced to a 2D trajectory. By projecting the dataset on the drawing plane defined by two eigenvectors, the trajectory is then described by 2 signals (the 3rd eigenvectors is not used to reconstruct the dataset). Similarly, when reaching for an object, the joint angle trajectory of the shoulder is correlated with the joint angle trajectory of the elbow, and, thus, the shoulder and elbow trajectories could be expressed by only one signal.

3.2 Encoding in Hidden Markov Models (HMM)

For each gesture, a set of time series $\{\mathbf{S}_1^X, \mathbf{S}_2^X, \ldots, \mathbf{S}_I^X, \mathbf{S}_1^\Theta, \mathbf{S}_2^\Theta, \ldots, \mathbf{S}_J^\Theta\}$ is used to train a Hidden Markov Model with $I + J$ output variables. The parameters are expressed as a set of parameters $\{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$, representing respectively the initial states distribution, the states transition probabilities, the means of the output variables, and the standard deviations of the output variables [1].

Continuous HMMs are used to encode the data with a parametric description of the distributions. A single Gaussian is assumed to approximate sufficiently each output variable (see Figure 4). A *mixture of Gaussian* could approximate any shape of distribution. However, it is not useful in our system, since the

[1] People unfamiliar with HMM should refer to [18]

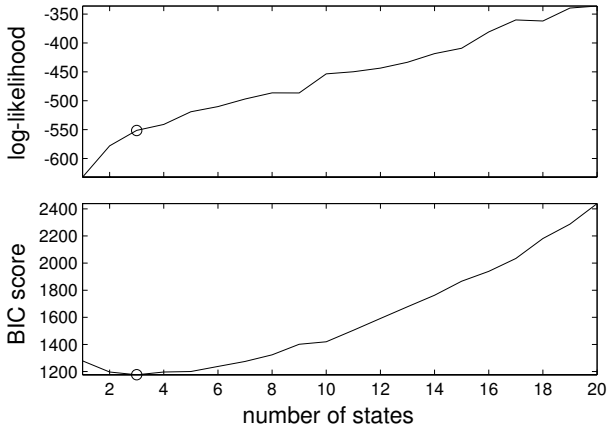


Fig. 5. A BIC criterion is used to determine the optimal number of states of the HMM, required to encode the data. *Top*: log-likelihood of the HMM according to the number of states used to represent the data. *Bottom*: the minimum BIC score gives a criterion to select the minimal number of states required to represent the data. It finds a trade-off between maximizing the likelihood of the model and minimizing the number of parameters used to model the data. Here, the gesture *waving goodbye* is modeled optimally with only 3 states.

training is performed with too few training data to generate an accurate model of distribution with more than one Gaussian.

The transition probabilities $P(q_t{=}j|q_{t-1}{=}i)$ and the observation distributions $p(S_t|q_t{=}i)$ are estimated by Baum-Welch, an *Expectation-Maximization* algorithm, that maximizes the likelihood that the training dataset can be generated by the corresponding model. The optimal number of states in the HMM may not be known beforehand. The number of states can be selected by using a criterion that weights the model fit (i.e. how well the model fits the data) with the economy of parameters (i.e the number of states used to encode the data). In our system, the Bayesian Information Criterion (BIC) [21] is used to select an optimal number of states for the model:

$$BIC = -2\,log(L) + n_p\,log(T) \tag{1}$$

The first term is used for the model fit, with L the likelihood of the fitted model. The second term is a penalty term, with n_p the number of independent parameters in the HMM, and T the number of observation data used in fitting the model. Data are encoded in different HMMs from one state to 20 states, and the model with the minimum score is retained (see Figure 5).

3.3 Recognition

Once trained, the HMM can be used to recognize whether a new gesture is similar to the ones encoded in the model. For each of the HMM, we run the

forward-algorithm to estimate the likelihood that the new signals could have been generated by one of the models. A measure of distance across two model's predictions is compared to a model-dependent threshold, to guarantee that the gesture is close to a model, but far enough from the others to be considered as recognized (see [7] for details).

3.4 Data retrieval

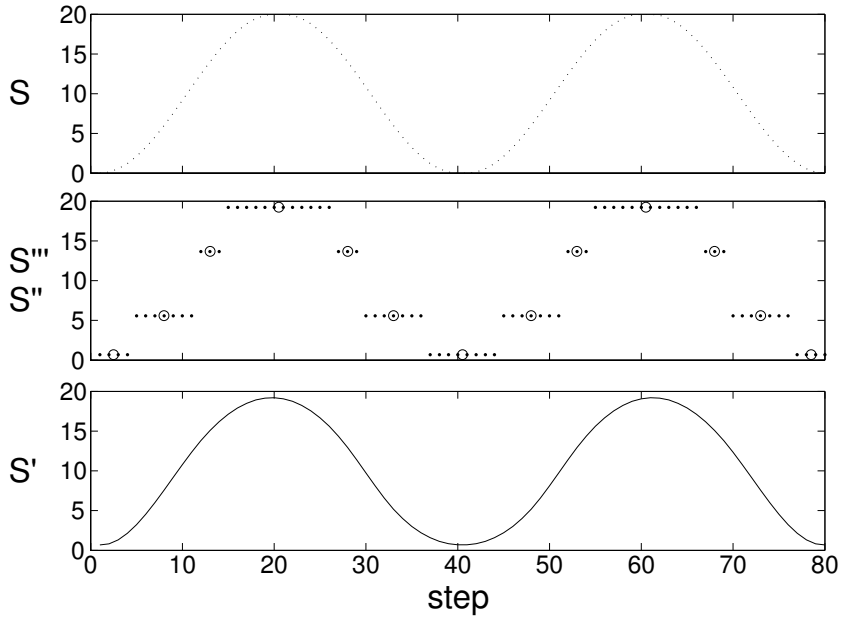


Fig. 6. Example of the retrieval process. The original signal S (*dotted-line*) is encoded in a HMM with 4 states. A sequence of states and corresponding output variables S''' are retrieved by the Viterbi algorithm (*points*). Keypoints S'' are defined from this sequence of output variables (*circles*). The retrieved signal S' (*straight-line*) is then computed by interpolating between the keypoints and normalizing in time.

When a gesture is recognized by a HMM, a generalization of the gesture is reproduced. Given the observation of the gesture and the parameters $\{\boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$ of the HMM, a sequence of states is reconstructed by the Viterbi algorithm. Given this sequence of states, the output variables $\{\mathbf{S}_1^{'''X}, \mathbf{S}_2^{'''X}, \ldots, \mathbf{S}_I^{'''X}, \mathbf{S}_1^{'''\Theta}, \mathbf{S}_2^{'''\Theta}, \ldots, \mathbf{S}_J^{'''\Theta}\}$ are retrieved, by taking the mean value $\boldsymbol{\mu}$ of the Gaussian distribution for each output variable.

Keypoints $\{\mathbf{S}_1^{''X}, \mathbf{S}_2^{''X}, \ldots, \mathbf{S}_I^{''X}, \mathbf{S}_1^{''\Theta}, \mathbf{S}_2^{''\Theta}, \ldots, \mathbf{S}_J^{''\Theta}\}$ are then extracted from these time series. If there is a transition to a state n at time t_1 and if there is a transition to another state at time t_2, a keypoint is created at the mean

time $\frac{t_1+t_2}{2}$. By interpolating between these key-points and normalizing in time, the output variables $\{\mathbf{S}_1'^X, \mathbf{S}_2'^X, \ldots, \mathbf{S}_I'^X, \mathbf{S}_1'^\Theta, \mathbf{S}_2'^\Theta, \ldots, \mathbf{S}_J'^\Theta\}$ are reconstructed (see Figure 6). Finally, by using the eigenvectors found by PCA, the whole hand path $\{\mathbf{X}_1', \mathbf{X}_2', \mathbf{X}_3'\}$ and joint angle trajectories $\{\boldsymbol{\Theta}_1', \boldsymbol{\Theta}_2', \boldsymbol{\Theta}_3', \boldsymbol{\Theta}_4'\}$ are reconstructed.

3.5 Imitation metrics

In [3], [7] and [8], we have developed a general formalism for determining the metric of imitation performance. The metric measures the quality of the reproduction, and, as such, drives the selection of an appropriate controller for the reproduction of the task.

One way to compare the relative importance of each set of variables (i.e. joint angles, hand path) in our experiment is to look at their variability. Here, we take the perspective that the relevant features of the movement, i.e. those to imitate, are the features that appear most frequently, i.e. the invariants in time, and apply the metric to determine the relevance of the Cartesian and joint angle representation to reproduce a gesture.

Following this framework, we model the task's cost function as a weighted linear combination of metrics applied to the joint angle trajectories and the hand path.

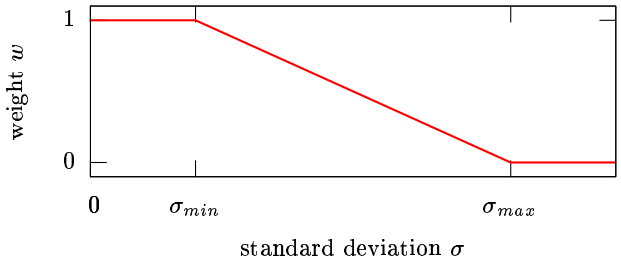


Fig. 7. Function used to transform a standard deviation σ to a weight factor $w \in [0, 1]$. σ_{min} corresponds to the accuracy of the sensors. σ_{max} represents the maximal standard deviation measured during a set of demonstrations generated by moving randomly the arms during one minute.

Unidimensional case: Let $D = \{x_1, x_2, \ldots, x_T\}$ and $D' = \{x_1', x_2', \ldots, x_T'\}$ be the demonstration and the reproduction datasets of a variable x. The cost function J is defined by:

$$J(D, D') = 1 - f(e) \tag{2}$$

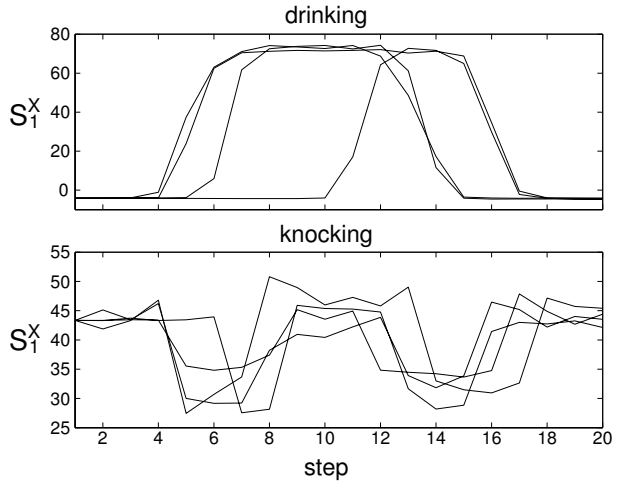


Fig. 8. 4 demonstrations of the *drinking* gesture and the *knocking* gesture (only one variable from the visual information is represented). Even if the trajectories are rescaled in time, data do not overlap, because they present non-homogeneous distortions in time. By encoding the data in HMM, it is still possible to distinguish very well the two datasets.

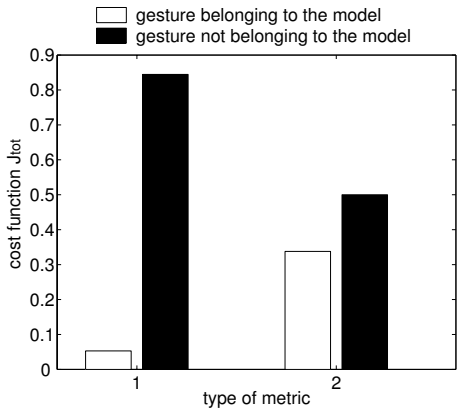


Fig. 9. Comparison of two metrics to evaluate the quality of a reproduced trajectory. *Left:* using the error measure e based on HMM encoding of the dataset. *Right:* using the RMS error e' with the trajectories rescaled in time. The white and black bar corresponds respectively to the data belonging to the model, and not belonging to the model. The error based on HMM encoding discriminates better the 2 datasets.

$J \in [0,1]$ calculates an estimation of the quality of the reproduction, using two different metrics. Optimizing the imitation consists of minimizing J (J=0 corresponds to a perfect reproduction). e is a measure of distance across the observed data D' and the training data D. A transformation function $f()$ normalizes and bounds each variable within minimal and maximal values (see Figure 7). This results in the elimination of the effect of the noise, intrinsic to each variable, so that the relative importance of each variable can be compared.

The metric uses the HMM representation of the data to compute the error value e, robust to distortion in time. The *Viterbi algorithm* is first used to retrieve the best sequence of states $\{q_1, q_2, \ldots, q_T\}$, given the observation data $D' = \{x'_1, x'_2, \ldots, x'_T\}$ of length T. If $\{\mu_1, \mu_2, \ldots, \mu_T\}$ is the sequence of means associated with the sequence of states, we define:

$$e = \frac{1}{T} \sum_{t=1}^{T} |x'_t - \mu_t| \tag{3}$$

We have compared this error measure to the most commonly used *root mean square* (RMS) error, calculated with signals rescaled in time, using the dataset shown in Figure 8. The RMS error is computed as:

$$e' = \frac{1}{T} \sum_{t=1}^{T} |x'_t - x_t| \tag{4}$$

The results of the metrics calculated using e or e' are presented in Figure 8. Each data has been tested with the two models, and should produce respectively a low value of J if they belong to the corresponding model, and a high value if they do not. The metric using the HMM representation of the time-series gives better results than the one using the static error computed on rescaled signals (see Figure 9). Indeed, HMM can deal with the distortions in time in the 2 datasets.

Multidimensional case: When data have K dimensions, the metric J_{tot} is expressed as

$$J_{tot} = \frac{1}{K} \sum_{i=1}^{K} w_i \, J(D_i, D'_i) \tag{5}$$

$w_i \in [0,1]$ weight the importance of each set of variables. These factors are extracted from the demonstration and reflect the variance of the data during the demonstration. To evaluate this variability, we also use the statistical representation provided by the HMM. The *Viterbi algorithm* is used to retrieve the best sequence of states $\{q_1, q_2, \ldots, q_T\}$, given the observation data D'. If

$\{\sigma_1^i, \sigma_2^i, \ldots, \sigma_T^i\}$ is the sequence of standard deviations of variable i, associated with the sequence of states, we define:

$$w_i = f(\frac{1}{T}\sum_{t=1}^{T}\sigma_t^i) \tag{6}$$

If the variance of a given variable is high, i.e. showing no consistency across demonstrations, then, satisfying some particular instance of this variable will have little bearing on the task. The factors w_i in the cost function equation reflect this assumption: if the standard deviation of a given variable is low, the value taken by the corresponding w_i are close to 1. This way, the corresponding variable will have a strong influence in the reproduction of the task.

A mean standard deviation is thus calculated over the whole path, and is transformed by a function (see Figure 7) to give a weight $w_i \in [0, 1]$ to estimate the relevance of dataset i.

w_i can then motivate the use of either a direct joint angles controller or an inverse kinematics controller. In order to use both controllers simultaneously, one can extend the inverse kinematics solution to encapsulate constraints on the joint angles, as in [8].

Since the demonstrator and the robot do not share the same embodiment (they differ in the length of their arms and in the range of motion of each DOF), there is a *correspondence problem* [15]. Here, this problem is solved by hands. The joint angle trajectories of the demonstrator are automatically shifted and scaled, when required, to ensure that these fit within the range of motion of the robot.

4 Results and performance of the system

The training set consists of the joint angle trajectories and hand path of 4 subjects performing the 6 different motions. The test set consists of 4 other subjects performing the same 6 motions, with only the hand path trajectories, to demonstrate the recognition ability of the model even in the face of missing data. Once a gesture has been recognized by the model based on the hand path only, the complete joint angle trajectories can be retrieved.

23 motions have been recognized correctly (recognition rate of 96%). The only error has happened for one instance of the *knocking on a door* motion, that has been confused with the *waving goodbye* motion. This is not surprising. Indeed, when projecting these two motions on the first principal component of their respective model, one observes that the resulting trajectories are quite similar (both motions involve a principal oscillatory component). Thus, it can be difficult to classify them correctly using exclusively the time series after projection. The robustness of the system could be improved by comparing the principal components extracted from the test set with the ones extracted from the training set, in combination with the HMM classification. The drawback is that the

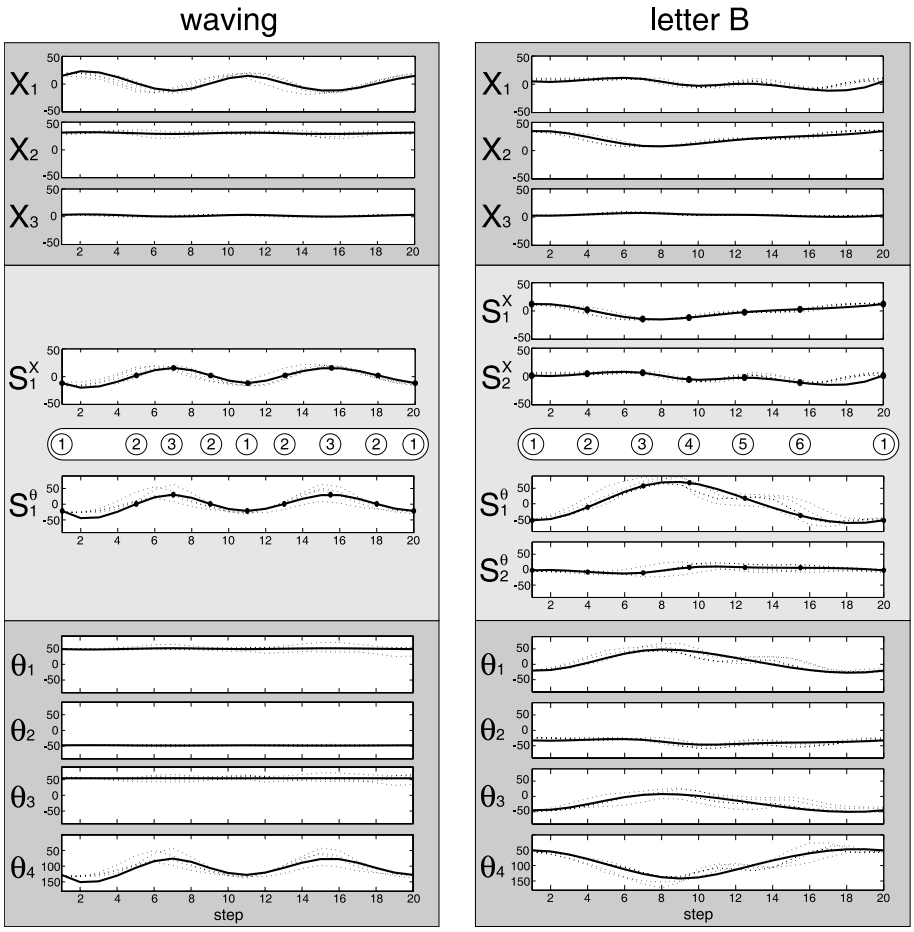


Fig. 10. Demonstration, transformation by PCA, encoding in HMM and retrieval of the 2 different motions *waving goodbye* and *drawing letter B*. The 5 demonstrations in visual coordinates $\{\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3\}$ and motor coordinates $\{\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2, \boldsymbol{\Theta}_3, \boldsymbol{\Theta}_4\}$ are represented by dotted lines. The retrieved generalized trajectory is in bold line.

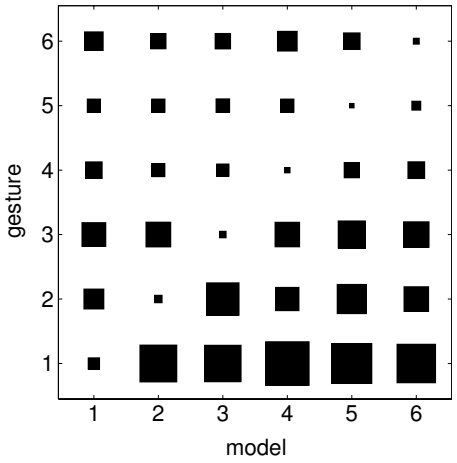


Fig. 11. Cost function J_{tot}, when testing the gestures of the test set with the different HMMs (the size of the square is proportional to J_{tot}). Each row corresponds to a specific gesture: *1)* drawing letter A, *2)* drawing letter B, *3)* drawing letter C, *4)* waving goodbye, *5)* knocking on a door, *6)* drinking. These gestures are tested with the 6 corresponding models. For each row, the column with lowest value indicates what is the best model corresponding to the gesture.

system would not be able to recognize a similar gesture performed in a different situation. For example, the principal components of an alphabet letter are not the same if the user writes it on a table or on a blackboard. After projection, the resulting signals are however similar, and can be recognized by HMM.

Figure 1, 2 and 10 shows the encoding and decoding of the 6 motions. As expected, 2 principal components are sufficient to represent the hand path when drawing each of the 3 letters, as well as when performing the *knocking* gesture. For *waving* and *drinking*, a single component is sufficient. Consistently, 2 principal components are sufficient to represent the joint trajectories when drawing the 3 letters of the alphabet, while only a single component is required to represent the gestures of *waving*, *knocking* and *drinking*.

The resulting signals for *letter A*, *waving*, *knocking* and *drinking* are modeled by a HMM of 3 states. *letter B* is modeled with 6 states, and *letter C* with 4 states. The keypoints in the trajectories correspond roughly to inflexion points or relevant points describing the motion. The number of states found by the BIC criterion grows with the complexity of the signals to model.

Figure 11 represents the values of the cost function J, when testing the gestures of the test set with the different HMM models. The weights w_i found by the system are quite always similar for the motor and visual representations, which means that both representations could be used to reproduce the motion, with a slight preference to the visual representation. Indeed, we see on Figure 10 that there is not so much difference between the variations of the signals in both representations. It can be due to the experimental setup, where the motion of

the users are recorded only in one situation. In the next experiments, different situations or environments should be used to provide more variations in one or the other dataset.

5 Discussion on the model

The combination of PCA and HMM is used successfully in our application to reduce the dimensionality of a dataset and to extract the primitives of a motion. Preprocessing of the data using PCA removes the noise and reduces the dimensionality of the dataset, making the HMM encoding more robust. The parameters of the whole model are then $\{\boldsymbol{v}^X, \boldsymbol{v}^\Theta, \boldsymbol{m}^X, \boldsymbol{m}^\Theta, \boldsymbol{\pi}, \mathbf{A}, \boldsymbol{\mu}, \boldsymbol{\sigma}\}$. This requires less parameters than HMM encoding of the raw data (see [13, 7]), as the number of output values and number of states are optimally reduced.

The advantage of encoding the signals in HMMs, instead of using a static clustering technique to recognize the signals retrieved by PCA, is that it provides a better generalization of the data, with an efficient representation, robust to distortion in time. As an example, let us consider a situation, where two demonstrators A and B raise a glass at the same speed, drink with different speed, and put the glass back on a table simultaneously. In HMM, the difference in amplitude are fitted by a Gaussian, for each state and each variable. The distortions in time are handled by using a probabilistic description of the transitions between the states, while a simple normalization in time would not have generalized correctly over the 2 demonstrations.

The model is general in the sense that no information concerning the data is encapsulated in the PCA preprocessing or in the HMM classification, which makes no assumption on the form of the dataset. However, extracting the statistical regularities is not the only mean of identifying the relevant features in a task, and it would probably not allow learning of a more complicated task. In further work, we will exploit the use of other machine learning techniques to extract the optimal representation. In addition, we will consider the use of explicit pointers (e.g. speech) in combination to statistics, in order to extract the key-features more robustly and more efficiently.

Finally, it would be interesting to extend the model to using *asynchronous HMM*. Such models have been exploited successfully in speech processing to model the joint probability of pairs of asynchronous sequences describing the same sequence of events (e.g. visual lip reading and audio signals) [2]. It could be used in our application to learn and retrieve the best alignment between two sequences in visual and motor representations. It can be useful since the datasets are not always synchronized, but still needs to have correspondence between the different representations.

5.1 Similarity with works in Psychology and Ethology

Some of the features of our model bear similarities with those of theoretical models of animal imitation. These models often assume that data are correctly

discretized, segmented, and classified. By using PCA and HMM to encode the information, it is still possible to keep the elements of these frameworks that are relevant to a robotic system, offering at the same time a probabilistic description of the data, more suitable for a real-world application.

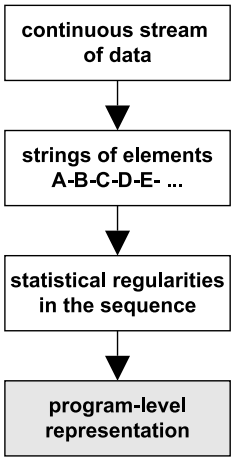


Fig. 12. R.W. Byrne's string parsing imitation model (schema inspired from [6])

Imitation using String Parsing R.W. Byrne has suggested to study the cognitive processes underlying animal's imitation by using *String Parsing* to segment a continuous task into basic elements [6]. Imitating a task with a sequential or hierarchical organization of basic actions have been observed in species as diverse as rats and apes, to learn complex feeding skills. The process requires an effective segmentation of the elements, so that imitation learning becomes a practical method to acquire more complex skills from basic elements. Such a segmentation allows to reuse known features, and extract the underlying structure in the observed behavior. If the whole task is perceived as a discrete sequence of items, the statistical regularities can be extracted, and the hierarchical structure is discovered by observing several times the same task.

Similarly, in a robotic system using PCA and HMM, the structure that underlies a sequence of elements can be acquired statistically by observing regularities across multiple demonstrations. Moreover, in HMM learning algorithm, a discrete set of key-features is extracted from a continuous flow of motion data, and the sequential organization of the key-features is learned by the model. The structure of the observed behavior is described probabilistically by transition probabilities between the key-features. By using a fully-connected model, it is thus possible to extract statistically the sequential structure of a task, with recurring elements and optional parts that are not always needed.

In our implementation, HMMs are used to find the key-features in the trajectories, and learn gestures by extracting the regularities in the sequences. Two concurrent stochastic processes are involved, one modeling the sequential structure of the data, and one modeling the local properties of the data.

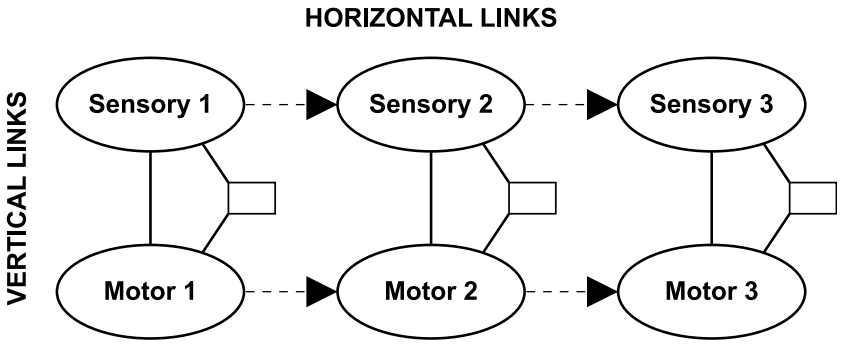


Fig. 13. C.M. Heyes and E.D. Ray's Associative Sequence Learning (ASL) model (schema inspired from [11])

Associative Sequence Learning (ASL) C.M. Heyes and E.D. Ray's *Associative Sequence Learning* (ASL) mechanism [12, 11] suggests that imitation requires a *vertical association* between a model's action, as viewed from the imitator's point of view, and the corresponding imitator's action. The vertical links between the sensory representation of the observed task and the motor representation are part of a repertoire, where elements can be added or refined. ASL suggests that the links are created essentially by experience, with a concurrent activation of sensory and motor representations. A few of them can also be innate, as biological data seem to indicate. The model pictures that the mapping between the sensory and motor representation can be associated with a higher level representation (boxes depicted in Figure 13).

The horizontal links model the successive activation of sensory inputs to learn the new task, that activates at the same time the corresponding motor representation, to copy the observed behavior. Repetitive activation of the same sequence strengthens the links to help motor learning. Depending on the complexity of task organization, numerous demonstrations may be needed to provide sufficient data to extract the regularities. The more data are available, the more evident is the underlying structure of the task, to clarify which elements are essential, which are optionals, and which are variations in response to changing circumstances.

This stresses the need of a probabilistic framework in a robotic application that can extract invariants across multiple demonstrations. Such a model is in agreement with a HMM decomposition of the task, where the underlying

structure is learned by using multiple demonstrations. If a given pattern appears frequently, its corresponding transition links are strengthen. If each action perceived by the imitator is also coded as an action that it can execute, the reproduction of a task can be considered. Any action that the imitator is able to perform can then also be recognized by observation of a model demonstrating the task.

Each hidden state in a HMM can output multimodal data. It can thus model multiple variables in different frames of reference. Its role is to make a link between the different datasets, and can be considered as a label or as a higher level representations common to the visual and motor data (see Figure 4). Indeed, in HMM, the sequence of states is not observed directly, and generates the visual or motor representation.

Thus, the architecture of a multivariate HMM has a horizontal process to associate the elements in a sequential order, by learning transition probabilities between the hidden states, and a vertical process that associates each sensory representation to appropriate motor representation, which is done through the hidden state. If data are missing from one part or the other (visual or motor representation), it is still possible to recognize a task, and retrieve a generalization of the task in the other representation, if required.

By using a similar representation of the ASL model in Figure 13, our system focus on learning the *horizontal links*. The *vertical associations* represented through the hidden states, are still hard-coded, specifying prior knowledge on the robot architecture. It involves a simple rescaling of the joint angles to fit the range of motion of the robot.

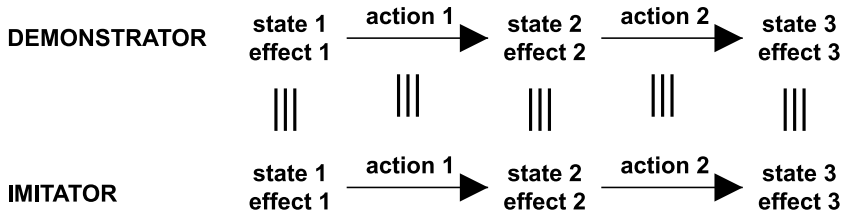


Fig. 14. C.L. Nehaniv and K. Dautenhahn's algebraic framework to map states, effects and/or actions of the demonstrator and the imitator (schema inspired from [17])

Algebraic framework for the correspondence problem The *correspondence problem* refers to the problem of creating an appropriate mapping between what is performed by the demonstrator and what is reproduced by the imitator. The two agents may not share the same embodiments (e.g. difference in limb lengths, sensors or actuators). Such correspondences can be constructed at various levels of granularity, reflecting the choice of a sequence of subgoals. C.L. Nehaniv and K. Dautenhahn have suggested a general algebraic framework [17,

16], to address the matching problem in both natural and artificial systems, and interpret the ASL model in this framework. It consists of representing the behavior performed by the demonstrator and imitator as two automata structures, with states and transitions. The *states* are the basic elements segmented from the whole task, that can produce *effects*, i.e. responses in the environment (e.g. object displaced). An *action* is the transition from one state to another state.

An imitation process is defined as a partial mapping process (relational homomorphism) between the demonstrator and imitator *states*, *effects*, and *actions*. An observer (e.g. external observer, demonstrator or imitator) decides which of the *states*, *actions* or *effects* are the most important ones to imitate, by fixing an imitation metric. Different metrics are used to yield solutions to different correspondence problems. These metrics also allow to formally quantify the success of an imitation.

This notation is closely related to the one used in our system. A HMM is an extension of the automata depicted in this algebraic framework. The difference is that these automata are described stochastically, and are thus more suitable to be used with noisy data, in a real-world application. Each hidden state outputs probabilistically distributed values that can be seen as *effects*, and the equivalent of *actions* are the transitions between hidden states, also probabilistically defined. Note that encoding the data in HMM does not resolve the correspondence problem, but gives a suitable framework for its representation, to treat the *what-to-imitate* and the correspondence problem in a common framework.

In further research, our work will address the correspondence problem paradigm [1, 16, 17, 15]. The contribution in our system would be to ensure a robust mapping between the two agents. In Alissandrakis et al work, an external observer decides which mappings are relevant to the imitation, i.e decides which of the states, the actions or the effects should be mapped. The similarity is measured by observer-dependent metrics. The contribution of our work would be to extract the relevant features of a task from the statistical regularities across the multiple demonstrations, instead of specifying them in advance.

6 Conclusion

This chapter has presented an implementation of a HMM-based system to encode, generalize, recognize and reproduce gestures, with representation of the data in visual and motor coordinates. The model has been tested and validated in a humanoid robot, using kinematics data of human motion.

The framework offers a stochastic method to model the process underlying gesture imitation. It makes a link between theoretical concepts and practical applications. In particular, it stresses the fact that the observed elements of a demonstration, and the organization of these elements should be stochastically described to have a robust robot application, that takes account of the high variability and the discrepancies across demonstrator and imitator points of view.

Acknowledgments

Thanks to the undergraduate students C. Dubach and V. Hentsch who helped to develop the motion tracking with x-sens and vision. This work was supported in part by the Swiss National Science Foundation, through grant 620-066127 of the SNF Professorships program, and by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020, Integrated Project COGNIRON.

References

1. A. Alissandrakis, C.L. Nehaniv, and K. Dautenhahn. Imitating with alice: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, 32(4):482–496, 2002.
2. S. Bengio. Multimodal speech processing using asynchronous hidden markov models. *Information Fusion*, 5(2):81–89, 2004.
3. A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.
4. A. Billard and G. Hayes. Drama, a connectionist architecture for control and learning in autonomous robots. *Adaptive Behavior*, 7(1):35–64, 1999.
5. A. Billard and R. Siegwart. Robot learning from demonstration. *Robotics and Autonomous Systems*, 47(2-3):65–67, 2004.
6. R.W. Byrne. Imitation without intentionality. using string parsing to copy the organization of behaviour. *Animal Cognition*, 2:63–72, 1999.
7. S. Calinon and A. Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Proceedings of the IEEE/RSJ Intl Conference on Intelligent Robots and Systems (IROS)*, pages 2769–2774, Sendai, Japan, September 28 - October 2 2004.
8. S. Calinon, F. Guenter, and A. Billard. Goal-directed imitation in a humanoid robot. In *Proceedings of the IEEE Intl Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, April 18-22 2005.
9. J. Demiris and G. Hayes. *Imitation as a Dual-Route Process Featuring Predictive and Learning Components: A Biologically-Plausible Computational Model*, chapter 13, pages 327–361. MIT Press, c. nehaniv and k. dautenhahn edition, 2001.
10. R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47(2-3):109–116, 2004.
11. C.M. Heyes. Causes and consequences of imitation. *Trends in Cognitive Sciences*, 5:253–261, 2001.
12. C.M. Heyes and E.D. Ray. What is the significance of imitation in animals? *Advances in the Study of Behavior*, 29:215–245, 2000.
13. T. Inamura, I. Toshima, and Y. Nakamura. *Acquiring Motion Elements for Bidirectional Computation of Motion Recognition and Generation*, volume 5, pages 372–381. Springer-Verlag, b. siciliano and p. dario edition, 2003.
14. I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.
15. C. Nehaniv and K. Dautenhahn. *Of Hummingbirds and Helicopters: An Algebraic Framework for Interdisciplinary Studies of Imitation and Its Applications*, volume 24, pages 136–161. World Scientific Press, j. demiris and a. birk edition, 2000.

16. C.L. Nehaniv and K. Dautenhahn. Like me? - measures of correspondence and imitation. *Cybernetics & Systems: An International Journal*, 32(1-2):11–51, 2001.
17. C.L. Nehaniv and K. Dautenhahn. *The correspondence problem*, pages 41–61. MIT Press, 2002.
18. L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, February 1989.
19. S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.
20. S. Schaal, A. Ijspeert, and A Billard. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, 358(1431):537–547, 2003.
21. G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

Recognition and Reproduction of Gestures using a Probabilistic Framework combining PCA, ICA and HMM

Sylvain Calinon SYLVAIN.CALINON@EPFL.CH
Aude Billard AUDE.BILLARD@EPFL.CH
Autonomous Systems Lab, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

Abstract

This paper explores the issue of recognizing, generalizing and reproducing arbitrary gestures. We aim at extracting a representation that encapsulates only the key aspects of the gesture and discards the variability intrinsic to each person's motion. We compare a decomposition into principal components (PCA) and independent components (ICA) as a first step of preprocessing in order to decorrelate and denoise the data, as well as to reduce the dimensionality of the dataset to make this one tractable. In a second stage of processing, we explore the use of a probabilistic encoding through continuous Hidden Markov Models (HMMs), as a way to encapsulate the sequential nature and intrinsic variability of the motions in stochastic finite state automata. Finally, the method is validated in a humanoid robot to reproduce a variety of gestures performed by a human demonstrator.

1. Introduction

Robot learning by imitation, also referred to as *robot programming by demonstration* (RbD) explores novel means of implicitly teaching a robot new motor skills. Such approach to "learning by apprenticeship" has proved to be advantageous for learning multidimensional and non-linear functions. The demonstrations constrain the search space by showing possible and/or optimal solutions (Isaac & Sammut, 2003; Abbeel & Ng, 2004; Billard et al., 2004). A core assumption of such approach is that the demonstration set is sufficiently complete and that it shows solutions that are also optimal, or at least possible, for the imitator. The latter condition is not necessarily met when the demonstrator and the imitator differ importantly in their perception and action spaces, as it is the case when transferring skills from a human to a robot.

In the work presented here, we go beyond pure gesture recognition and explore the issues of recognizing, generalizing and reproducing arbitrary gestures. We aim at extracting a representation of the data that encapsulates only the key aspects of the gesture and discards the variability intrinsic to each person's motion. This representation makes it, then, possible for the gesture to be reproduced by an imitator agent (in our case, a robot) whose space of motion differs significantly in its geometry and natural dynamics to that of the demonstrator.

Recognition and generalization must span from a very small dataset. Indeed, because one cannot ask the demonstrator to produce more than 5 to 10 demonstrations, one must use algorithms that manage to discard the high variability of the human motions, while not setting up priors on the representation of the dataset (that is highly context- and task-dependent).

In our experiments, the robot is endowed with numerous sensors enabling it to track faithfully the kinematics of the demonstrator's motions. The data gathered by the different sensors are redundant and correlated, as well as subjected to various forms of noise (sensor dependent). Thus, prior to applying any form of encoding of the gesture, we perform a decomposition of the data into either principal components (PCA) or independent components (ICA), in order to decorrelate and denoise the data, as well as to reduce the dimensionality of the dataset to make this one tractable.

In order to generalize across multiple demonstrations, the robot must encode multivariate time-dependent data in an efficient manner. One major difficulty in learning, recognizing and reproducing sequential patterns of motion is to deal simultaneously with the spa-

Appearing in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

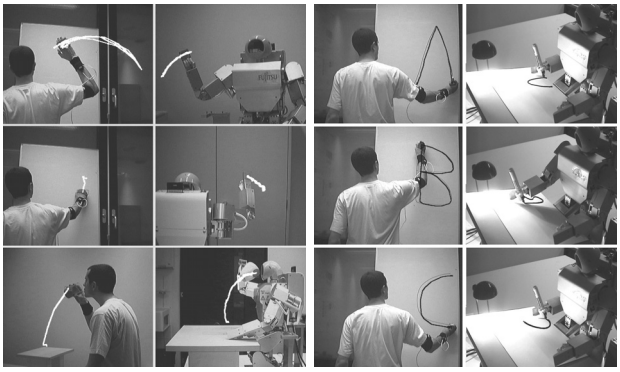


Figure 1. 1st and 3rd columns: Demonstration of different gestures. *2nd and 4th columns:* Reproduction of a generalized version of the gestures. The trajectories of the demonstrator's hand, reconstructed by the stereoscopic vision system, are superimposed to the image.

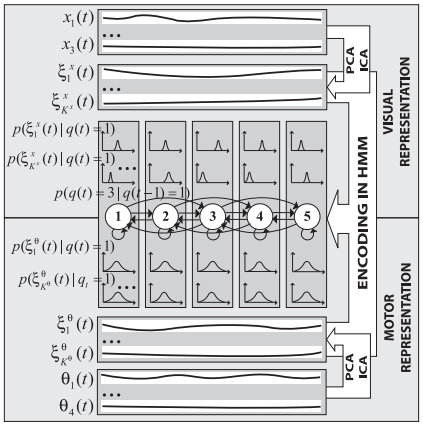


Figure 2. Encoding of the hand path $\vec{x}(t)$ and of the joint angles trajectories $\vec{\theta}(t)$ in a HMM. The data are preprocessed by PCA or ICA, and the resulting signals $\{\vec{\xi^x}(t), \vec{\xi^\theta}(t)\}$ are learned by the HMM. The model is fully connected (for clarity of the picture, some of the transitions have been omitted).

tial and temporal variations of the data, see e.g. (Chudova et al., 2003). Thus, in a second stage of processing, we explore the use of a probabilistic encoding through continuous Hidden Markov Models (HMMs), as a way to encapsulate the sequential nature and intrinsic variability of the motions in stochastic finite state automata. Each gesture is, then, represented as a sequence of states, where each state has an underlying probabilistic description of the multi-dimensional data (see Figure 2).

Similar approaches to extracting primitives of motion have been followed, e.g., by (Kadous & Sammut, 2004; Ijspeert et al., 2002). Our approach complements (Kadous & Sammut, 2004) by investigating how these primitives can be used to reconstruct a generalized and parameterizable form of the motion, so that these can be successfully transferred into a different dataspace (that of the robot). Moreover, in contrast to (Ijspeert et al., 2002), who take sets of Gaussians as the basis of the system, we avoid predefining the form of the primitives and let the system discover those through an analysis of variance.

Closest in spirit to our approach is the work of (Abbeel & Ng, 2004), who use a finite-state Markov decision process to encode the underlying constraints of an apprenticeship driving task. While this approach lies in a discrete space, in our work, we must draw from continuous distributions to encapsulate the continuity in time and space of the gestures.

2. Experimental set-up

Data consist of human motions performed by eight healthy volunteers. Subjects were asked to imitate a set of 6 gestures demonstrated in a video recording. The motions consist in: 1) Knocking on a door, 2) Bringing a cup to one's mouth and putting it back on the table, 3) Waving goodbye and 4-6) Drawing the stylized alphabet letters A, B and C (see Figure 1).

Three *x-sens* motion sensors attached to the torso and the right upper- and lower-arm of the demonstrator recorded the kinematics of motion of the shoulder joint (3 degrees of freedom (DOFs)) and of the elbow joint (1 DOF) with a precision of 1.5 degrees and at a rate of 100Hz. A color-based stereoscopic vision system tracks the 3D-position of a marker placed on the demonstrator's hand, at a rate of 15Hz, with a precision of 10 mm.

The experiments are performed on a Fujitsu humanoid robot HOAP-2 with 25 DOFs. Note that only the robot's right arm (4 DOFs) is used for reproducing the gestures. The torso and legs are set to a constant and stable position, in order to support the robot's standing-up.

3. Data processing

Let $\vec{x}(t) = \{x_1(t), x_2(t), x_3(t)\}$ be the hand path, and $\vec{\theta}(t) = \{\theta_1(t), \theta_2(t), \theta_3(t), \theta_4(t)\}$ the joint angle trajectories of the right arm after interpolation and normalization in time. The data are first projected onto a low-dimensional subspace, using either PCA or ICA. The resulting signals are, then, encoded in a set of HMMs (see Figure 2). A generalized form of the signals is, then, reconstructed by interpolating between

the key-points retrieved by the HMMs. The complete signals are then recovered by projecting the data onto the robot's workspace.

3.1. Principal Component Analysis (PCA)

PCA determines the directions along which the variability of the data is maximal (Jolliffe, 1986). We apply PCA separately to the set of variables $\vec{\theta}(t)$ and $\vec{x}(t)$ in order to identify an underlying uncorrelated representation in each dataset. After subtracting the means from each dimension, we compute the covariance matrices $C^x = E(\vec{x}\vec{x}^T)$ and $C^\theta = E(\vec{\theta}\vec{\theta}^T)$.

The 3 eigenvectors $\vec{v}_i^x$ and associated eigenvalues λ_i^x of the hand path are given by $C^x\vec{v}_i^x = \lambda_i^x\vec{v}_i^x$, $\forall i \in \{1,\ldots,3\}$. The 4 eigenvectors $\vec{v}_i^\theta$ and associated eigenvalues λ_i^θ of the joint angle trajectories are given by $C^\theta\vec{v}_i^\theta = \lambda_i^\theta\vec{v}_i^\theta$, $\forall i \in \{1,\ldots,4\}$. We project the two datasets onto their respective basis of eigenvectors and obtain the time series $\vec{\xi^x}(t) = \{\xi_1^x(t), \xi_2^x(t), \ldots, \xi_{K^x}^x(t)\}$ for the hand path, and $\vec{\xi^\theta}(t) = \{\xi_1^\theta(t), \xi_2^\theta(t), \ldots, \xi_{K^\theta}^\theta(t)\}$ for the joint angle trajectories. K^x and K^θ form, respectively, the minimal number of eigenvectors to obtain a *satisfying* representation of each original dataset, i.e. such that the projection of the data onto the reduced set of eigenvectors covers at least 98% of the data's spread: $\sum_{i=1}^{K}\lambda_i > 0.98$.

Applying PCA before encoding the data in a HMM has the following advantages: 1) It helps reducing noise, as the noise is now encapsulated in the lower dimensions (but it also discards the high-frequency information). 2) It reduces the dimensionality of the dataset, which reduces the number of parameters in the Hidden Markov Models, and speeds up the training process. 3) It produces a parameterizable representation of the dataset that offers the required flexibility to generalize to different constraints. For example, the 3D path followed by the demonstrator's hand, when drawing a letter of the alphabet, can be reduced to a 2D signal.

3.2. Independent Component Analysis (ICA)

Similarly to PCA, ICA is a linear transformation that projects the dataset onto a basis that best represents the statistical distribution of the data. ICA searches the *directions along which statistical dependence of the data is minimal* (Hyvärinen, 1999).

Let $\vec{x}$ be a multi-dimensional dataset resulting from a linear composition of the independent signals $\vec{s}$, given by: $\vec{x} = \mathbf{A}\vec{s}$. ICA consists of estimating both the "sources", i.e. $\vec{s}$, and the mixing matrix $\mathbf{A}$ by maximizing the non-gaussianity of the independent components. Non-gaussianity can be estimated using, among others, a measure of negentropy.

Here, we use the fixed-point iteration algorithm developed by (Hyvärinen, 1999). Prior to applying ICA, we reduce the dimensionality of the dataset following the PCA decomposition described above and, consequently, apply PCA on the K optimal components. While with PCA, the components are ordered with respect to their eigenvalues λ_i, which allows us to easily map the resulting signals to their corresponding HMM output variables, ordering the ICA components is unfortunately not as straightforward. Indeed, the order of the components is somewhat random. In order to resolve this problem, we order the ICA signals according to their negentropy[1].

3.3. Hidden Markov Model (HMM)

For each gesture, a set of time series $\{\vec{\xi^x}(t), \vec{\xi^\theta}(t)\}$ is used to train a fully connected continuous Hidden Markov Model with K^x+K^θ output variables. The model takes as parameters the set $M=\{\vec{\pi}, \mathbf{A}, \mu, \sigma\}$, representing, respectively, the initial states distribution, the states transition probabilities, the means of the output variables, and the standard deviations of the output variables. For each state, the output variables are described by multivariate Gaussians, i.e. $p(\xi_i^\theta) \sim \mathcal{N}(\mu_i^\theta, \sigma_i^\theta)\ \forall i \in \{1,\ldots,K^\theta\}$ and $p(\xi_i^x) \sim \mathcal{N}(\mu_i^x, \sigma_i^x)\ \forall i \in \{1,\ldots,K^x\}$ A single Gaussian is assumed to approximate sufficiently each output variable[2] (see Figure 2).

The transition probabilities $p(q(t)=j|q(t\text{-}1)=i)$ and the observation distributions $p(\xi(t)|q(t)=i)$ are estimated by the *Baum-Welch* algorithm, an *Expectation-Maximization* algorithm, that maximizes the likelihood that the training dataset can be generated by the corresponding model. The optimal number of states in the HMM may not be known beforehand. The number of states can be selected by using a criterion that weights the model likelihood (i.e. how well the model fits the data) with the economy of parameters (i.e the number of states used to encode the data). In our system, the *Bayesian Information Criterion* (BIC) (Schwarz, 1978) is used to select an optimal number of states for the model:

$$BIC = -2\,log(L) + n_p\,log(T) \qquad (1)$$

[1]Note that this does not completely ensure that the ordering is conserved and a manual checkup is sometimes required.

[2]There is no advantage to use a *mixture of Gaussians* for our system, since the training is performed with too few training data to generate an accurate model of distribution with more than one Gaussian.

where $L = P(D|M)$ is the likelihood of the model M, given the observed dataset D, n_p is the number of independent parameters in the HMM, and T the number of observation data used in fitting the model (in our case $T = (K^x + K^\theta) \cdot N$, for trajectories of size N). The first term of the equation is a measure of how well the model fits the data, while the second term is a penalty factor that aims at keeping the total number of parameters low. In our experiments, we compute a set of candidate HMMs with up to 20 states and retain the model with the minimum score.

3.4. Recognition Criteria

For each experiment, the dataset is split equally into a training and a testing set. Once trained, the HMM can be used to recognize whether a new gesture is similar to the ones encoded in the model. For each HMM, we run the *forward-algorithm* (Rabiner, 1989), an iterative procedure to estimate the likelihood L that the observed data D could have been generated by the model M, i.e. $L = P(D|M)$. In the remaining of the paper, we will refer to the log-likelihood value $LL = log(L)$, a usual means of computing the likelihood. A gesture is said to belong to a given model when the associated LL is strictly greater than a given fixed threshold ($LL > -100$ in our experiments). In order to compare the predictions of two concurrent models, we set a minimal threshold for the difference across log-likelihoods of the two models ($\Delta LL > 100$ in our experiments). Thus, for a gesture to be recognized by a given model, the voting model must be very confident (i.e. generating a high LL), while other models predictions must be sufficiently low in comparison.

3.5. Data Reconstruction

Once a gesture has been recognized, the robot imitates the gesture, by producing a similar (generalized form of) the gesture. The generalized form of the gesture is reconstructed in 5 steps (see Figure 3): 1) We first extract the best sequence of states (according to the model's parameters $\{\vec{\pi}, \mathbf{A}, \mu, \sigma\}$), using the *Viterbi algorithm* (Rabiner, 1989). 2) We, then, generate a time-series of $K^x + K^\theta$ variables $\{\vec{\xi'''^x}(t), \vec{\xi'''^\theta}(t)\}$ by computing the mean values μ of the Gaussian distribution of each output variable at each state. 3) We then reduce this time series to a set of key-points $\{\vec{\xi'^x}(t), \vec{\xi'^\theta}(t)\}$, in-between each state transitions. 4) By interpolating between these key-points and normalizing in time, we construct the set of output variables $\{\vec{\xi'^x}(t), \vec{\xi'^\theta}(t)\}$, using Piecewise cubic Hermite polynomial functions (the benefits of this transformation on the stability of the system are discussed in Section 5.2).

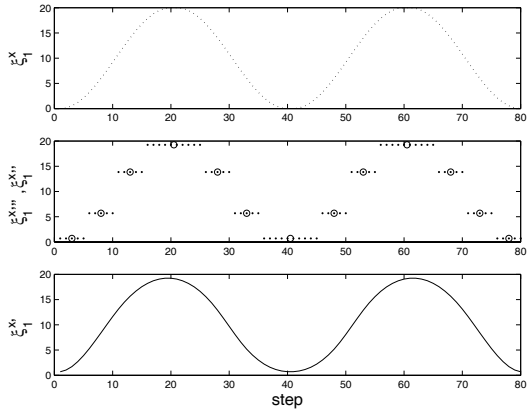


Figure 3. Schematic of the retrieval process on a generic sine curve. The original signal $\{\xi_1^x(t)\}$ (*dotted-line*) is encoded in a HMM with 4 states. A sequence of states and corresponding output variables $\{\xi_1^{x'''}(t)\}$ are retrieved by the *Viterbi* algorithm (*points*). Key-points $\{\xi_1^{x''}(t)\}$ are defined from this sequence of output variables (*circles*). The retrieved signal $\{\xi_1^{x'}(t)\}$ (*straight-line*) is then computed by interpolating between the key-points and normalizing in time.

5) Finally, by reprojecting the time series onto the robot's workspace (using a rescaling transformation on the linear map extracted by PCA/ICA), we recompute the complete hand path $\vec{x}'(t)$ and joint angle trajectories $\vec{\theta}'(t)$, which is, then, fed to the robot controller.

4. Selection of a controller

In (Billard et al., 2004), we determined a cost function according to which we can measure the quality of the robot's reproduction and drive the selection of a controller. The controller combines direct and inverse-kinematics, so as to optimize the cost function. In other words, the controller balances reproducing either the demonstrated hand path or the demonstrated joint angle trajectories (note that these two constraints may be mutually exclusive in the robot's workspace), according to their relative importance.

The relative importance of each set of variables is inversely proportional to its variability. The rational is that, if the variance of a given variable is high, i.e. showing no consistency across demonstrations, this suggests that satisfying some particular constraints on this variable will have little bearing on the task.

The variance of each set of variables is estimated using the probability distributions computed by the HMMs during training. If $\{q(t)\}$ is the best sequence of states retrieved by a given model, and $\{\sigma(t)\}$ the associated

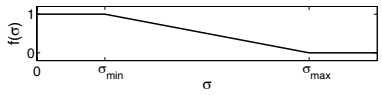


Figure 4. In order to relate the variability of the different signals collected by the robot (given that these come from modalities that differ in their measurement units and resolution), we define a transfer function $f(\sigma)$ to normalize and bound each variable, such that $f(\sigma) \in [0; 1]$.

sequence of standard deviations, we define:

$$\alpha = \begin{cases} 0 \text{ if } f(\bar{\sigma}^x) > f(\bar{\sigma}^\theta) \\ 1 \text{ if } f(\bar{\sigma}^x) \leq f(\bar{\sigma}^\theta) \end{cases} \quad (2)$$

where $\bar{\sigma}^x$ is the mean variation for the hand path and $\bar{\sigma}^\theta$ the mean variation for the joint angles over the different states (see also Figure 4). α determines the controller to reproduce the task. $\alpha = 0$ corresponds to a direct controller, while $\alpha = 1$ corresponds to an inverse-kinematics controller. In the experiments reported here, we determined that $\alpha = 0$ for the *waving* and *knocking* gesture (i.e. direct controller), and $\alpha = 1$ for the other gestures (i.e. inverse kinematics controller).

5. Stability issues

In this section, we briefly discuss the stability of our learning system and of our controller. This is, however, not a formal proof.

5.1. Stability of the learning criterion

Once recognized, a gesture can be used to re-adjust the model's parameters, assuming that the gestures used to train the model are still available. However, a given model will be readjusted to fit a given gesture iff the likelihood that the model has produced the gesture is sufficiently large, i.e. larger than a fixed threshold[3], see Section 3.4. In practice, we found that such criteria insure that a good model will not depreciate over time. We trained an uncorrupted model continuously, starting with 0% noise and adding up to 40% noise to the dataset (after which the recognition performance would depreciate radically, as shown in Table 1, and the new gestures would not be used for training). The results are reported in Figure 8, showing that the original model remains little disturbed by the process.

The above constraints, however, do not satisfy completely the algorithm proposed by (Ng & Kim, 2005)

[3]The thresholds were set by hand and proved to ensure sufficiently strong constraint on the stability, while allowing some generalization over the natural variability of the data.

Table 1. Recognition rates as a function of the spatial noise: *'test data'* refers to a testing set comprising original human data corrupted with spacial and temporal noise, see Figure 5. *'retrieved data'* refers to a testing set comprising synthetic data, generated by corrupted models.

	TEST DATA		RETRIEVED DATA	
	PCA	ICA	PCA	ICA
HUMAN DATA (HD)	100%	100%	-	-
HD + r^s=10%	72.0%	75.3%	80.3%	86.0%
HD + r^s=20%	65.0%	73.0%	79.3%	81.0%
HD + r^s=30%	54.0%	66.7%	73.7%	82.3%
HD + r^s=40%	35.0%	34.7%	73.3%	84.3%
HD + r^s=50%	15.3%	13.0%	74.0%	81.3%

to ensure stability of an online learning system. Since LL is a measure of the variability of the data of order N, where N is the number of states in our system, the above two conditions ensure that the complete variability of the sequence is within bound. However, it does not ensure that each state's variability is bounded.

5.2. Stability of the controller

The issue of the stability of the controller is beyond the scope of this paper. However, in practice and to fulfill some basic engineering requirements, we have used methods that ensure that the system will be bounded within the robot's workspace.

The piecewise cubic Hermite polynomial functions, also referred to as "clamped" cubic spline, used to interpolate the trajectories across the model's keypoints, ensures BIBO stability, i.e., under bounded disturbances, the original signal remains bounded and does not diverge (Sun, 1999).

The robot's motion are controlled by a built-in PID controller, whose gains have been set so as to provide a stable controller for a given range of motions. In order to insure the stability of the Fujitsu controller, the trajectories are automatically rescaled, shifted or cut off, if they are out-of-range during control.

6. Results and performance of the system

We trained the model with a dataset of 4 subjects performing the 6 different motions shown in Figure 1. After training, the model's recognition performance were measured against a test set of 4 other individuals performing the same 6 motions. Subsequently, once a gesture had been recognized, we tested the model's

capacity to regenerate the encoded gesture, by retrieving the corresponding complete joint angle trajectories and/or hand path, depending on the decision factor α, see Section 4. These trajectories were then run on the robot, as shown in Figure 1.

All motions of the test set were recognized correctly[4]. The signals for the *letter A*, *waving*, *knocking* and *drinking* gestures were modelled by HMMs with 3 states, while the *letter B* was modelled with 6 states and the *letter C* with 4 states. The key-points for each gesture corresponded roughly to inflexion points on the trajectories (i.e. relevant points describing the motion). The number of states found by the BIC criterion grows with the complexity of the signals we modelled.

We found that 2 PCA or ICA components were sufficient to represent the hand path as well as the joint trajectories for most gestures. We observed that the signals extracted by PCA and ICA presented many similarities. Moreover, as expected, we observed that the principal and independent components for both joint angle trajectories and hand paths bear the same qualitative characteristics, highlighting the correlations between the two datasets. Figure 6 shows an example of resulting trajectories when applying ICA preprocessing.

7. Robustness to noise

In order to evaluate systematically the robustness of our system to recognizing and regenerating gestures against temporal and spatial noise, we generated two new datasets based on the human dataset. The first dataset, aimed at testing the recognition capabilities of the system, consisted of the original human data corrupted with either spatial and temporal noise. The second dataset, aimed at measuring the reconstruction capabilities of the system, consisted of synthetic data, generated by a corrupted model, i.e. a model trained with the first training of corrupted human data. We report the results of each set of measures in Table 1.

7.1. Noise generation

Temporal noise was created by generating non-homogeneous deformations in time on the original signal (see Figure 5). The signal is discretized into N

[4]Note that, when reducing the number of components with PCA, i.e. $\sum_{i=1}^{K} \lambda_i > 0.8$ instead of $\sum_{i=1}^{K} \lambda_i > 0.98$, an error happened for one instance of the *knocking on a door* motion, that was confused with the *waving goodbye* motion. This is not surprising, since both motions involve the same type of oscillatory component.

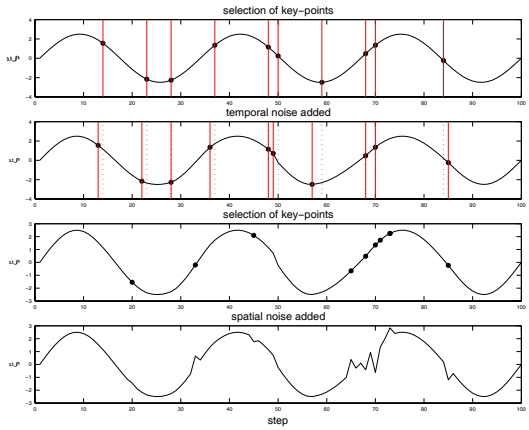


Figure 5. Noise generation process on a generic sine curve, with parameters $\{n^t, r^t, n^s, r^s\}$=$\{10\%, 50\%, 10\%, 50\%\}$. *1st row:* Random selection of 10 key-points. *2nd row:* Addition of temporal noise. *3rd row:* Random selection of 10 key-points. *4th row:* Addition of spatial noise.

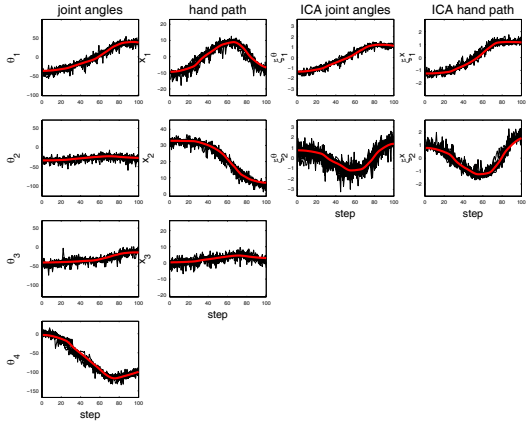


Figure 6. Decomposition and reconstruction of the trajectories resulting from drawing the alphabet letter C. Training data, added with synthetic noise ($\{n^t, r^t, n^s, r^s\}$=$\{10\%, 20\%, 10\%, 20\%\}$), are represented in thin lines. Superimposed to those, we show, in lighter bold lines, the reconstructed trajectories.

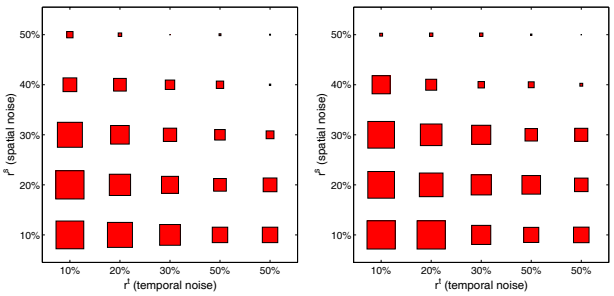


Figure 7. Recognition rates as a function of spatial and temporal noise, using either PCA (*left*) or ICA (*right*) decomposition. The squares size is proportional to the recognition rate, between 0% (smallest) and 100% (largest).

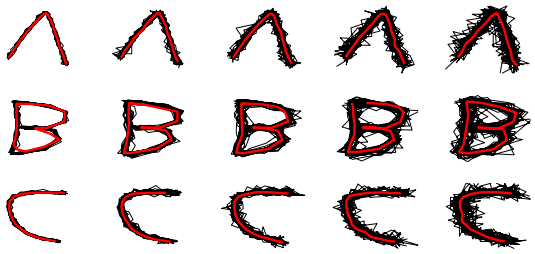


Figure 8. In bold, trajectory retrieved by a model trained successively with a dataset containing (from left to right) r^s=$\{10\%, 20\%, 30\%, 40\%, 50\%\}$ of noise, using PCA decomposition.

points. The algorithm goes as follows: 1) Select randomly $n^t \cdot N$ key-points in the trajectory, with a uniform distribution of size N. 2) Displace each key-point randomly in time following a Gaussian distribution centered on the key-point with a standard deviation $r^t\bar{\sigma}^t$. $\bar{\sigma}^t$ is the mean standard deviation of the distribution of key-points, i.e. $\bar{\sigma}^t(N) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(i-\frac{N}{2})^2}$. 3) Reconstruct the noisy signal by interpolating between the key-points.

Then, *spatial noise* was created by adding white noise to the original signal (see also Figure 5). The algorithm goes as follows: 1) Select $n^s \cdot N$ key-points in the trajectory, with a uniform random distribution of size N. 2) Displace each key-point randomly in space, with a Gaussian distribution centered on the key-point, with standard deviation $r^s\bar{\sigma}_s$. $\bar{\sigma}^s$ is the mean standard deviation in space, i.e. $\bar{\sigma}^s$=$\bar{\sigma}^\theta$ for the joint angles and $\bar{\sigma}^s$=$\bar{\sigma}^x$ for the hand path.

7.2. Recognition performance

In order to measure the recognition performance of our model, we trained a model with an uncorrupted dataset of human gesture (note that the dataset still encapsulated the natural variability of human motion), and tested its recognition rate against a corrupted dataset. The corrupted dataset was created by adding spatial and temporal noise to the original human dataset with n^t=10%, r^t=$\{10\%, 20\%, 30\%, 40\%, 50\%\}$ and n^s=100%, r^s=$\{10\%, 20\%, 30\%, 40\%, 50\%\}$. Comparative results for PCA and ICA preprocessing are presented in Figure 7 and in Table 1. We observed that the recognition rate clearly decreases with an increase in spatial noise. It also decreases with an increase in temporal noise, but less so than for the spatial noise, in agreement with the known robustness of HMM encoding of time series in the face of time distortions.

7.3. Reconstruction performance

The same process was carried out to evaluate the reconstruction performance of the system. We, first, trained a set of "corrupted models" with a set of human data corrupted with temporal and spatial noise. We, then, regenerated a set of signals from each of the corrupted models (see Figure 8). Finally, we measured the recognition rate of the good model (trained with uncorrupted human data) against this set of reconstructed signals, see Table 1. The recognition performance are better with the regenerated dataset than with the original corrupted dataset. This is not surprising, since the signals regenerated from corrupted models are by construction (through the Gaussian estimation of the observations distribution) less noisy than the ones used for training (since they are more likely to show a variability close to the mean of the noise distribution).

8. Discussion on the model

Results showed that the combinations PCA-HMM and ICA-HMM were both very successful at reducing the dimensionality of the dataset and extracting the primitives of each gesture. For both methods, the recognition rates and reconstruction performances were very high. As expected, preprocessing of the data using PCA and ICA removes well the noise, making the HMM encoding more robust. A second advantage of PCA/ICA encoding is that it reduces importantly the amount of parameters required for encoding the gestures in the HMM in contrast to using raw data as in (Inamura et al., 2003; Calinon et al., 2005).

The average performance using ICA decomposition is slightly better to that using PCA. However, ICA preprocessing is less deterministic than PCA preprocessing. Indeed, ICA components are computed iteratively, starting from a random distribution. Thus, the algorithm does not ensure to find the same components at each run. PCA directly orders the components with respect to their eigenvalues, while ICA components are ordered with respect to their negentropy value, which can induce errors. To achieve optimal encoding requires, thus, a manual check-up.

The advantage of encoding the signals in HMMs, instead of using a static clustering technique to recognize the signals retrieved by PCA/ICA, is that it provides a better generalization of the data, with an efficient representation, robust to distortion in time. An HMM encoding accounts for the difference in amplitude across the signals in the Gaussian distributions associated to each state. The distortions in time are handled by

using a probabilistic description of the transitions between the states, while a simple normalization in time would not have generalized correctly over demonstrations performed with time distortions.

Finally, a strength of the model lies in that it is general, in the sense that no information concerning the data is encapsulated in the preprocessing or in the HMM classification, which makes no assumption on the form of the dataset. However, extracting the statistical regularities is not the only mean of identifying the relevant features in a task. Moreover, such an approach would not scale up to learning complex tasks, consisting of sequential presentations of multiple gestures. In further work, we will exploit the use of priors in the form of either explicit segmentation points (e.g. generated by an external modalities such as speech), or in the form of a kernel composed of generic signals extracted by our present work to learn tasks involving sequential and hierarchical presentations of gestures.

9. Conclusion

This paper presented an implementation of a PCA/ICA/HMM-based system to encode, generalize, recognize and reproduce gestures. The model's robustness to noise was tested systematically and validated in a real world set-up using a humanoid robot and kinematics data of human motion. This work is part of a general framework that aims at improving the robustness of current methods in robot programming by demonstration, so as to make those suitable to a wide range of robotic applications. The present work demonstrates the usefulness of using a stochastic method to encode the characteristic elements of a gesture and the organization of these elements. Moreover, such a method generates a representation that accounts for the variability and the discrepancies across demonstrator and imitator sensory-motor spaces.

Acknowledgments

We would like to thank the reviewers for their useful comments and advices. The work described in this paper was partially conducted within the EU Integrated Project COGNIRON and was supported in part by the Swiss National Science Foundation, through grant 620-066127 of the SNF Professorships program.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *Proceedings of the International Conference on Machine Learning (ICML)*.

Billard, A., Epars, Y., Calinon, S., Cheng, G., & Schaal, S. (2004). Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, *47:2-3*, 69–77.

Calinon, S., Guenter, F., & Billard, A. (2005). Goal-directed imitation in a humanoid robot. *Proceedings of the IEEE Intl Conference on Robotics and Automation (ICRA)*. Barcelona, Spain.

Chudova, D., Gaffney, S., Mjolsness, E., & Smyth, P. (2003). Translation-invariant mixture models for curve clustering. *Proceedings of the international conference on Knowledge discovery and data mining* (pp. 79–88). New York, NY, USA.

Hyvärinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, *10*, 626–634.

Ijspeert, A., Nakanishi, J., & Schaal, S. (2002). Learning attractor landscapes for learning motor primitives. *Advances in Neural Information Processing Systems (NIPS)* (pp. 1547–1554).

Inamura, T., Toshima, I., & Nakamura, Y. (2003). Acquiring motion elements for bidirectional computation of motion recognition and generation. In B. Siciliano and P. Dario (Eds.), *Experimental robotics viii*, vol. 5, 372–381. Springer-Verlag.

Isaac, A., & Sammut, C. (2003). Goal-directed learning to fly. *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 258–265). Washington, D.C.

Jolliffe, I. (1986). *Principal component analysis*. Springer-Verlag.

Kadous, M., & Sammut, C. (2004). Constructive induction for classifying multivariate time series. *European Conference on Machine Learning*.

Ng, A. Y., & Kim, H. J. (2005). Stable adaptive control with online learning. *Proceedings of the Neural Information Processing Systems Conference, NIPS 17*.

Rabiner, L. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77:2*, 257–285.

Schwarz, G. (1978). Estimating the dimension of a model. *Annals of Statistics*, *6*, 461–464.

Sun, W. (1999). Spectral analysis of hermite cubic spline collocation systems. *SIAM Journal of Numerical Analysis*, *36:6*.

Proceedings of the 2005 IEEE
International Conference on Robotics and Automation
Barcelona, Spain, April 2005

Goal-Directed Imitation in a Humanoid Robot

Sylvain Calinon, Florent Guenter and Aude Billard
Autonomous Systems Lab (ASL3)
Swiss Federal Institute of Technology Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{*sylvain.calinon, florent.guenter, aude.billard*}*@epfl.ch*

***Abstract*— Our work aims at developing a robust discriminant controller for robot programming by demonstration. It addresses two core issues of imitation learning, namely *"what to imitate"* and *"how to imitate"*. This paper presents a method by which a robot extracts the goals of a demonstrated task and determines the imitation strategy that satisfies best these goals. The method is validated in a humanoid platform, taking inspiration of an influential experiment from developmental psychology.**

I. Introduction

Robots programming by demonstration has become a key topic of research in robotics (see [1] for a recent overview of core approaches in the domain). Work in that area tackles the development of robust algorithms for motor control, motor learning, gestures recognition and visuo-motor integration. While the field exists for more than twenty years, recent developments, taking inspiration in biological mechanisms of imitation, have brought a new perspective to the domain [2]. Programming by demonstration, now, encompasses more of the learning components of traditional approaches and is often referred to as *learning by imitation*.

Two core issues of imitation learning are known as *"what to imitate"* and *"how to imitate"* [3]. *What to imitate* refers to the problem of determining which of the features of the demonstration are relevant for the achievement of the task [4]. *How to imitate*, also referred to as the *correspondence problem* [5], is the problem of transferring an observed motion into one's own capabilities. Works tackling this issue have followed either an approach in which the correspondence is unique and the imitation must produce an exact, but parameterizable, reproduction of the trajectories [6]–[8], or an approach in which only a subset of predefined goals must be reproduced (e.g. [9]–[12]).

While prior work has concentrated on either of these issues separately, we propose a system that combines a method for solving the *what to imitate* problem by extracting the task constraints, with a method for solving the *how to imitate* problem given a set of task constraints. This paper extends our theoretical framework for solving the *what to imitate* problem [4], in incorporating the notion of goal preference and including a method for optimizing the reproduction (*how to imitate*). the complete system is validated in a humanoid platform, reproducing an influential experiment from developmental psychology.

The experimental set-up and the methods for data pre-processing are detailed in Section II. In Section III, we present a statistical method for extracting the constraints and inferring their relative importance. In Section IV, we describe a method to optimize the trajectory, using the task constraints extracted in the first phase of the analysis. Results and discussion are presented in Section V and VI.

II. Experimental set-up

A. Experimental scenario

The experiment starts with the (human) demonstrator and the (robot) imitator standing in front of a table, facing each other (see Figure 4). On both sides of the table, two colored dots (red and green) have been stamped at equal distance to the demonstrator and imitator's starting positions. In a first set of demonstrations, the demonstrator reaches for each dot alternatively with left and right arm. If the demonstrator reaches for the dot on the left handside of the table with his left arm, it is said to perform an ipsilateral motion. If conversely the demonstrator reaches the dot on the right handside of the table with his left arm, it is said to perform a contralateral motion. Then the demonstrator produces the same ipsilateral and contralateral motions, but without the presence of dots.

Each of these motions are demonstrated five times consecutively. In each case, the demonstrator starts from the same starting position. While observing the demonstration, the robot tries to make sense of the experiment by extracting the demonstrator's intention underlying the task. I.e. it determines a set of constraints for the task, by extracting relevant features in a statistical manner. When the demonstration ends, the robot computes the trajectory that satisfies best the constraints extracted during the demonstration and generates a motion that follows this trajectory.

The scenario of our experiment is a replication of a set of psychological experiments conducted with young children and adults [13]. In these experiments, Bekkering and colleagues have shown that children have a tendency to substitute ipsilateral for contralateral gestures, when the dots are present. In contrast, when the dots are absent from the demonstration, the number of substitutions drop significantly. Thus, despite the fact that the gesture is the same in both conditions, the presence or absence of a physical object (the dot) affects importantly the reproduction. When the object is present, object selection takes the highest

0-7803-8914-X/05/$20.00 ©2005 IEEE.

priority. Children, then, nearly always direct their imitation to the appropriate target object, at the cost of selecting the "wrong" hand. When removing the dots, the complexity of the task (i.e. the number of constraints to satisfy) is decreased, and, hence, constraints of lower importance can be fulfilled (such as producing the same gesture or using the same hand). Similar experiments conducted with adults have corroborated these results, by showing that the presence of a physical object affects the reproduction[1].

These experiments are informative to robotics, in helping us determine how to prioritize constraints (that we will also name goals throughout this paper) in a given task (and as such help us solve the *"correspondence problem"*). For instance, in the particular scenario, knowing the trajectory of the demonstrator's arm and hand path might not allow us to determine unequivocally the angular trajectories of the robot's arm. Indeed, depending on where the target is located, several constraints (goals) might compete and satisfying all of those would no always lead to a solution. For instance, in the case of contralateral motions, the robot's arm is too small to both reach the target and perform the same gesture. In that case, it must find a trade-off between satisfying each of the constraints. This amounts to determining the importance of each constraint with respect to one another.

B. Hardware

The demonstrator's motions are recorded by 5 x-sens motion sensors, attached to the torso and the upper- and lower-arms. Each sensor provides the 3D absolute orientation of each segment, by integrating the 3D rate-of-turn, acceleration and earth-magnetic field, at a rate of 100Hz. The angular trajectories of the shoulder joint (3 degrees of freedom) and the elbow (1 degree of freedom) are reconstructed by taking the torso as referential, with an accuracy of 1.5 degrees.

A color-based stereoscopic vision system tracks the 3D-position of the dots, the demonstrator's hands, and the robot's hands at a rate of 15Hz, with an accuracy of 10 mm. The system uses two Phillips webcams with a resolution of 320x240 pixels. The tracking is based on color segmentation of the skin and the objects in the YCbCr color space (only Cb and Cr are used, to be robust to changes in luminosity).

The humanoid robot is a Fujitsu HOAP-2. In this experiment, trajectory control affects only the two arms (4 DOFs each). The torso and legs are set to a constant position to support the robot's standing-up posture.

C. Encoding of the data into Hidden Markov Models

In order to reduce the dimensionality of the dataset to a subset of critical features, we pre-segment the joint angle trajectories and the hand path into a set of keypoints, corresponding to the inflexion points (see Figure V).

[1]In that case, the response latency is used instead of the proportion of errors

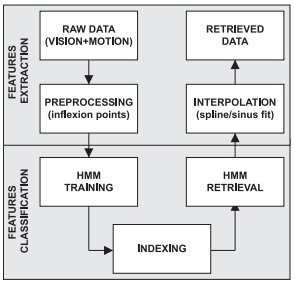


Fig. 1. Data processing loop. The trajectories are segmented into a set of keypoints. The keypoints sequences are classified using HMMs. The trajectories for the reproduction are generated by interpolation through the keypoints sequence regenerated by the HMMs.

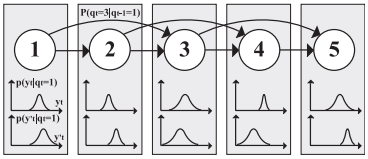


Fig. 2. Example of a left-right continuous HMM with 5 hidden states and 2 output variables y_t and y'_t.

The trajectories are subsequently encoded into Hidden Markov Models (HMMs), following our earlier work [14]. We here only briefly summarizes the method.

Each of the 4 joint angle trajectories is encoded in a left-right continuous HMM [15] (see Figure 1 and 2). The number of states is determined by the sequence with the highest number of keypoints in the training set. Each hidden state represents a keypoint j in the trajectory, and is associated with a stochastic representation, encoding two variables $\{y_j, y'_j\}$, namely the time lag between two keypoints and the absolute angle. The hand path is encoded in the same way, with 3 output distributions to encode the Cartesian components.

The transition probabilities $P(q_t{=}j|q_{t-1}{=}i)$ and the emission distribution $p(y_t|q_t{=}i)$ are estimated by the *Baum-Welch* iterative method. The *forward-algorithm* is used to estimate a log-likelihood value that an observed sequence could have been generated by one of the model. The *Viterbi algorithm* is used to generate a generalization of a trajectory over the demonstrations, by retrieving the best sequence of hidden states and the associated keypoint components. The corresponding trajectory is then reconstructed by applying a 3rd-order spline fit when using the Cartesian trajectory, and by applying a cosine fit when using the joint angle trajectory (see Figure V). The cosine fit corresponds to a cycloidal velocity profile, and keeps the keypoints as local maxima or minima during the reproduction.

III. Determining the task constraints

In [4] and [14], we have developed a general formalism for determining the metric of imitation performance. The

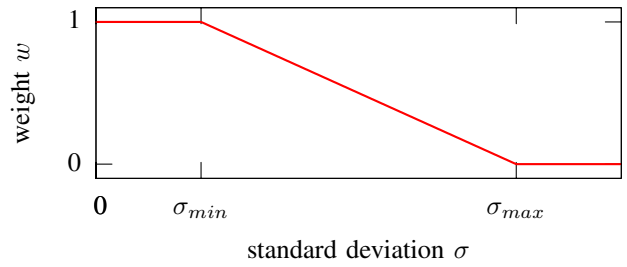


Fig. 3. Function used to transform a standard deviation σ to a weight factor $w \in [0,1]$. σ_{min} corresponds to the accuracy of the sensors. σ_{max} represents the maximal standard deviation measured during a set of demonstrations generated by moving randomly the arms around the setup during one minute.

metric (or cost function) measures the quality of the reproduction, and, as such, drives the selection of an appropriate controller for the reproduction of the task.

One way to compare the relative importance of each set of variables is to look at their variability. We take the perspective that the relevant features of the movement, i.e. those to imitate, are the features that appear most frequently, i.e. the invariants in time. We apply a cost function to determine the relevance of the different goals (or constraints) to reproduce a gesture. Following this framework, we model the task's cost function as a weighted linear combination of cost functions applied to the collected data.

A. Unidimensional case

Let $D = \{x_1, x_2, \ldots, x_T\}$ and $D' = \{x'_1, x'_2, \ldots, x'_T\}$ be the demonstration and the reproduction datasets of a variable x. The cost function J is defined by:

$$\begin{aligned} J(D, D') &= 1 - f(e) \\ e &= \frac{1}{T}\sum_{t=1}^{T} |x'_t - \mu_t| \end{aligned} \tag{1}$$

$J \in [0,1]$ gives an estimate of the quality of the reproduction. Optimizing the imitation consists of minimizing J. J=0 corresponds to a perfect reproduction. e is a measure of deviation between the observed data D' and the training data D, using the HMM representation of the data. The *Viterbi algorithm* is first used to retrieve the best sequence of states $\{q_1, q_2, \ldots, q_T\}$, given the observation data $D' = \{x'_1, x'_2, \ldots, x'_T\}$ of length T, where $\{\mu_1, \mu_2, \ldots, \mu_T\}$ are the sequence of means associated with the sequence of states. A transformation function $f()$ normalizes and bounds each variable within minimal and maximal values (see Figure 3). This eliminates the effect of the noise, intrinsic to each variable, so that the relative importance of each variable can be compared.

B. Multidimensional case

Let us consider a dataset of K variables. The complete cost function J_{tot} is given by:

$$J_{tot} = \frac{1}{K}\sum_{i=1}^{K} w_i \, J(D_i, D'_i) \tag{2}$$

where $w_i \in [0,1]$ is a measure for the relative importance of each cost function. These weights reflect the variance of the data during the demonstration. To evaluate this variability, we use the statistical representation provided by the HMM. If $\{q_1, q_2, \ldots, q_T\}$ is the best sequence of states retrieved by the model, and if $\{\sigma^i_1, \sigma^i_2, \ldots, \sigma^i_T\}$ is the associated sequence of standard deviations of variable i, we define:

$$w_i = f(\frac{1}{T}\sum_{t=1}^{T} \sigma^i_t) \tag{3}$$

If the variance of a given variable is high, i.e. showing no consistency across demonstrations, this suggests that satisfying some particular constraints on this variable will have little bearing on the task. The factors w_i in the cost function equation reflect this assumption: if the standard deviation of a given variable is low, the value taken by the corresponding w_i are close to 1. This way, the corresponding variable will have a strong influence in the reproduction of the task.

C. Goal-directed case

Extracting the relative importance of each set of variables statistically is sometimes too slow, and requires to have observed enough data to estimate their distribution. In order to address this problem, we have added a set of *goal-preference parameters* α_i, extending our cost function to express explicitly how a constraint can be prioritized over another.

The cost function is thus applied to 4 sets of variables, namely the joint angle trajectories, the hand path, the location of the objects at which actions are directed (the dots), and the laterality of the motion (which hand is used).

Let $D = \{\Theta, X, d, h\}$ and $D' = \{\Theta', X', d', h'\}$ be the datasets generated by the demonstrator and imitator respectively. $\{\vec{\theta}_1, \vec{\theta}_2, \vec{\theta}_3, \vec{\theta}_4\}$ are the generalized joint angle trajectories over the demonstrations, $\{\vec{x}_1, \vec{x}_2, \vec{x}_3\}$ the generalized Cartesian trajectory of the hand, $\{o_{11}, o_{12}, o_{13}\}$ and $\{o_{21}, o_{22}, o_{23}\}$ the 3D location of the first and second dot respectively. We compute $d_{kj} = x_j - o_{kj}$ a difference measure for component j between the hand and a dot k, at the end of a trajectory. $h \in \{1,2\}$ corresponds to the usage of the left and right arm respectively.

With $N = 4$ joint angles for each arm and $O = 2$ objects, and given the position of the hand and the objects defined by $P = 3$ variables in Cartesian space (we make the assumption that only one hand is used at the same time), we define the general cost function J_{tot} as:

$$
\begin{aligned}
J_{tot} &= \alpha_1 \frac{1}{N} \sum_{i=1}^{N} w_1^i \, J_1(\vec{\theta}_i, \vec{\theta}'_i) \\
&+ \alpha_2 \frac{1}{P} \sum_{j=1}^{P} w_2^j \, J_2(\vec{x}_j, \vec{x}'_j) \\
&+ \alpha_3 \frac{1}{OP} \sum_{k=1}^{O} \sum_{j=1}^{P} w_3^{kj} \, J_3(d_{kj}, d'_{kj}) \\
&+ \alpha_4 \, w_4 \, J_4(h, h') \qquad (4)
\end{aligned}
$$

The new factors α_i, with $\sum_i \alpha_i = 1$ and $0 \leq \alpha_i \leq 1$, are fixed by the experimenter, and determine the relative importance of each subset of data, i.e. the importance of each constraint (or goal) in the overall task (α_1 for the joint angle trajectories, α_2 for the hand path, α_3 for the hand-object distance, and α_4 for the laterality[2]).

The cost function $J_{1,2,3}$ are given by Equation 2[3]. J_4 is given by $J_4(h, h') = |p(h{=}1) - p(h'{=}2)|$, where $p(h{=}1)$ is the probability of using the left arm during the demonstrations.

$w^j_{1,2,3}$ follow Equation 3, and are set to 0 if the corresponding component is missing (e.g. if the dot is not detected by the vision system).

w_4 is given by $w_4 = 2 \; |p(h{=}1) - 0.5|$. It represents the importance of using either the left or right hand (laterality of the imitation) and is based on a measure of the probability with which either hand has been used over the whole set of demonstrations (w_4=0 if there is no preference).

IV. Determining an optimal imitation

Once the cost function and the relative influence of each constraint have been determined, we generate an optimal (with respect to the cost function J_{tot}) trajectory. In order to do this, we first generate a set of candidate trajectories for the hand path, using the HMMs. Because of the difference in size between the demonstrator and the robot, we rescale the trajectories so that the starting and final positions of the demonstrator's hand correspond to the rest position of the robot's hand and corresponding location in the robot's space, respectively. In order to allow the robot to do by default a mirror-imitation, we transform the demonstrated trajectory following a vertical symmetry. The trajectories are then interpolated from the set of keypoints.

The new trajectories are considered as candidates for the hand path. To generate the corresponding joint angle trajectories, we have to solve the inverse kinematics equation given by $\vec{\dot{x}} = \mathbf{T}\vec{\dot{\theta}}$, where $\mathbf{T}$ is the Jacobian. A solution to this equation can be found using the *pseudo-inverse with optimization* numerical solution [16]:

[2]We make the assumption that a mirror imitation is used in preference by the robot

[3]J_3 is similar to J, with a HMM of one state

$w_1^1 = 0.96$ $w_1^2 = 0.98$ $w_1^3 = 0.88$ $w_1^4 = 0.73$
$w_2^1 = 0.89$ $w_2^2 = 0.93$ $w_2^3 = 0.82$
$w_3^{11} = 1.00$ $w_3^{12} = 1.00$ $w_3^{13} = 1.00$
$w_3^{21} = 1.00$ $w_3^{22} = 1.00$ $w_3^{23} = 1.00$
$w_4 = 1.00$

TABLE I

Mean weight values found by the system, with the presence of dots. By removing the dots, the values stay the same, except that the w_3^j values are null.

	$\alpha_1{=}\alpha_2{=}\alpha_3{=}\alpha_4$		$\alpha_1{=}\frac{1}{2}\alpha_2{=}\frac{1}{4}\alpha_3$, $\alpha_4{=}0$	
	Dots	No dots	Dots	No dots
Left contralateral	**0.16**	**0.14**	0.22	**0.11**
Right ipsilateral	0.36	0.47	**0.08**	0.16

TABLE II

Values of the cost function J_{tot}. In bold: optimal reproduction, with a demonstration of a contralateral motion with right hand. With no goal-preference (left column), the robot imitates in mirror-fashion. With a goal-directed cost function (right column), the robot uses its closest hand when the dots are present.

$$
\begin{aligned}
\vec{\dot{\theta}}_c &= \mathbf{T}^+ \vec{\dot{x}} + \alpha(\mathbf{I} - \mathbf{T}^+\mathbf{T}) g(\vec{\theta}) \\
\mathbf{T}^+ &= \mathbf{T}^T (\mathbf{T}\mathbf{T}^T)^{-1} \qquad (5) \\
g(\vec{\theta}) &= \vec{\theta}_r - \vec{\theta}
\end{aligned}
$$

$g(\vec{\theta})$ is an optimization term which tends to minimize the distance between the arm position and a rest position given by the middle range of each joint angle. To avoid the singularities, we give some bounds to the joint angles. $\vec{\dot{\theta}}_c$ gives a solution for the Cartesian trajectories. In order to account for the influence of the demonstrator's joint angle trajectories, we add a second constraint to the pseudo-inverse:

$$
\vec{\dot{\theta}} = \gamma \vec{\dot{\theta}}_c + (1 - \gamma) \vec{\dot{\theta}}_d \qquad (6)
$$

where $\vec{\dot{\theta}}_d$ is the derivative of the joint angle trajectory generated by the HMM after training, and γ is a factor used to tune the influence of the two different terms (reproduction of hand path or joint angle trajectories). For each candidate path and associated set of joint angle trajectories, we compute the value of the cost function J_{tot}. We, then, determine a local optimum for J_{tot}, by gradient-descent on γ, starting with $\gamma = 0$. The corresponding (locally) optimal trajectory is, then, run on the robot to reproduce the demonstration.

V. Experimental results

Table I shows the values of the weights w_i^j found for the different conditions, in the two sets of experiment (with or without the dots). As expected, we find little variation (w_i^j close to 1) in either the joint angle trajectories, the

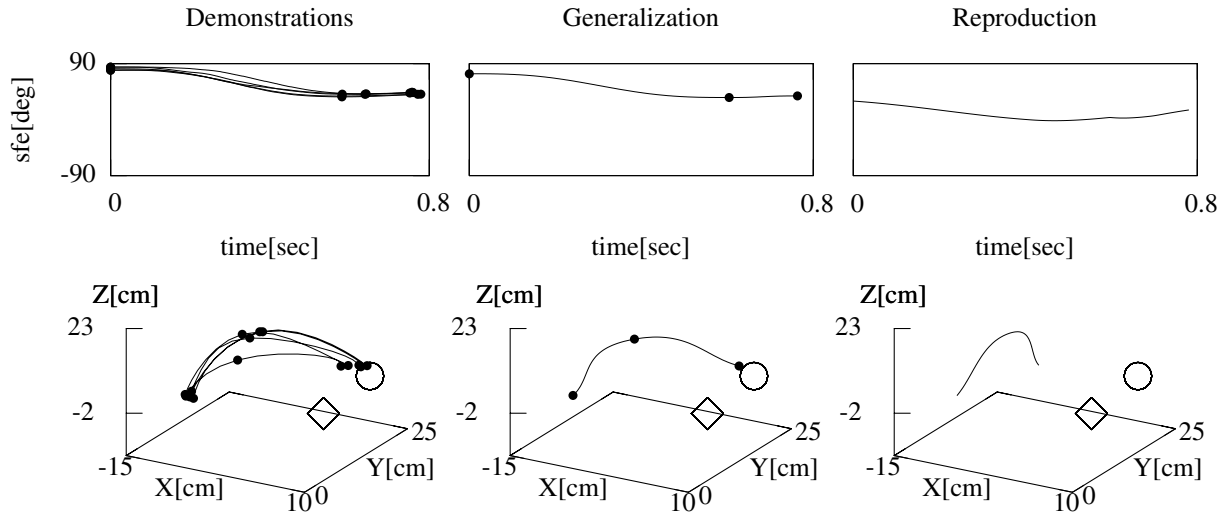


Fig. 6. Joint angle and hand path plots of 5 demonstrations of a contraleral motion with right hand (left column), the trajectory retrieved by the HMM model of the 5 demonstrations (middle column), and reproduction of a new motion by the robot to minimize the general cost function J_{tot}, with α_i set to the same value (right column). The points in the graphs represent the keypoints segmented and retrieved by the HMMs. The square and the circle show the position of the two dots on the table. Only the shoulder flexion-extension is represented for the joint angles. The trajectory is retrieved with a cosine fit for the joint angles, and with a 3rd-order spline fit for the hand path.

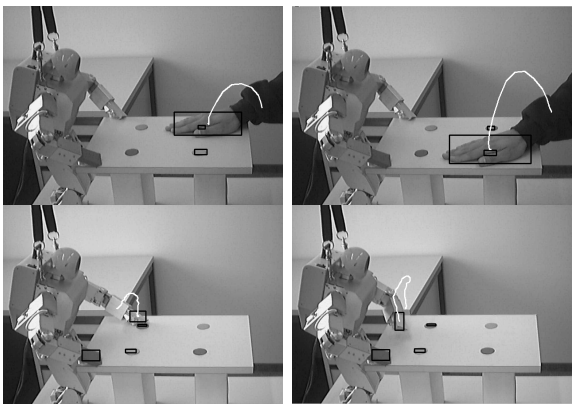


Fig. 4. *Top:* demonstration of an ipsi- (*top-left*) and contralateral (*top-right*) motion of the right arm. *Bottom:* reproduction of the motion candidate with lowest cost function J_{tot}, by using only statistics to extract the goals (α_1=α_2=α_3=α_4).

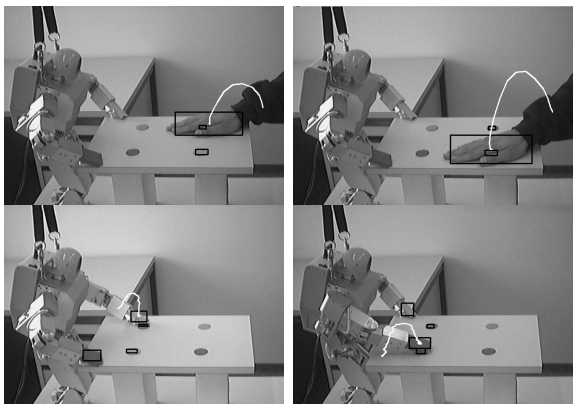


Fig. 5. Same experiment as in Figure 4, by adding a notion of goal-dominance to the general imitation cost function (α_1=$\frac{1}{2}\alpha_2$=$\frac{1}{4}\alpha_3$, α_4=0). This time, the contralateral motion is reproduced by doing an ipsilateral motion with the other arm, closer to the dot to touch.

hand paths, the hand-object distances or the laterality in any of the conditions, forcing the satisfaction of all constraints during the reproduction of goal-directed motion. In such a situation, since the robot does not share the same embodiment as that of the user, it will not be able to satisfy every constraint. The factors α_i of the cost function J_{tot} must, thus, prioritize the constraints to fulfill.

When the target dots are not present, their associated weights w_3^j values are null. Since there are no more object in the scene, the hand path and joint angle trajectories become the sole relevant features to reproduce.

In order to test the influence of setting the preference for each set of variables (i.e. goals) on the performance of imitation, we have computed J_{tot} with two sets of values: 1) α_1=α_2=α_3=α_4 (general cost function), i.e. no preference among the goals; 2) α_1=$\frac{1}{2}\alpha_2$=$\frac{1}{4}\alpha_3$, α_4=0 (goal-directed cost function).

For each set, we have selected the optimal controller. Table II gives the values of the cost functions J_{tot} in each case. Figures 4 and 5 show the resulting trajectories. The trajectories displayed on the images are the ones tracked by the vision system. Due to the limited range of motion

and short arm, the robot can not reach the goal with the contralateral motion. By adding a notion of goal-dominance to the general imitation cost function, the robot reproduces the contralateral motion of the demonstrator by doing an ipsilateral motion with the other arm, that is closer to the dot to touch (see Figure 5).

VI. Discussion and future work

This paper has presented a method to extract the constraints of a task and to use these to determine the optimal imitation strategy. The method has been validated on a humanoid platform to perform selective reproduction of a simple set of reaching tasks, and to illustrate an example of the goal-directed imitation. The method is generic and will be applied to more complex tasks in future work. However, the implementation has made a number of assumptions that will need to be further investigated. For instance, while the weights w_i^j can be extracted easily by statistics, the factors α_i (preference for the different goals), can not be determined solely through statistics. By fixing different values for α_i manually, we have observed different imitative behaviors. This notion of goal dominance is more difficult to learn only through passive observation: it requires feedback from the user or explicit teaching.

This experiment have shown that using a simple goal-directed cost function can be advantageous, when a statistical analysis of the data does not permit to determine the priorities of the constraints and/or when multiple constraints can not all be fulfilled.

In our everyday life, there is a rich variety of goals, and we select only a subset of goals to reproduce a task. Using a goal-directed approach in a robotic application has also the advantage to let the robot focus on only a subset of goals, if required. If a goal has been determined, the robot can then allow itself to loose track of the other events with lower priorities, and, thus, reduces the memory capacity used to compute these sensory information.

In future work, we plan to set up experiments to learn the parameters α_i of the cost function, so that the robot can determine to which extent it is more important to reproduce the same hand-object relationships, compared to the reproduction of the same hand paths, or the same joint angle trajectories. We have observed the effect that such factor can have in determining the laterality of the imitation. The choice of laterality in imitation depends on the type of actions to perform (as well as on handedness in humans). For example, the choice of hand used for pointing to an object might be less influenced by the demonstration than that of reproducing a grasping action. The factor α_4 could, then, be adjusted depending on the type of action being recognized in the demonstration.

In the experiments presented in this paper, the objects are static (dots on the table), and the relationships are the hand-object relative distances. Our next work will extend this notion to object-object and hand-object relationships, taking into account relative/absolute translation and rotation.

Acknowledgment

Thanks to the undergraduate students C. Dubach and V. Hentsch who helped to develop the motion tracking with x-sens and vision. This work was partially conducted within the EU Integrated Project COGNIRON funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020, and was supported by the Swiss National Science Foundation, through grant 620-066127 of the SNF Professorships program.

References

[1] A. Billard and R. Siegwart, "Robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 65–67, 2004.

[2] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[3] C. Nehaniv and K. Dautenhahn, "Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications," in *Interdisciplinary Approaches to Robot Learning*, J. Demiris and A. Birk, Eds. World Scientific Press, 2000, vol. 24, pp. 136–161.

[4] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 69–77, 2004.

[5] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Imitating with alice: Learning to imitate corresponding actions across dissimilar embodiments," *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, vol. 32, pp. 482–496, 2002.

[6] A. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots," in *Proceedings of the IEEE Intl Conference on Robotics and Automation*, 2002.

[7] A. Ude, C. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 93–108, 2004.

[8] J. Demiris and G. Hayes, "Imitation as a dual-route process featuring predictive and learning components: A biologically-plausible computational model," in *Imitation in Animals and Artifacs*, C. Nehaniv and K. Dautenhahn, Eds. MIT Press, 2001.

[9] R. Dillmann, "Teaching and learning of robot tasks via observation of human performance," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 109–116, 2004.

[10] J. Zhang and B. Rössler, "Self-valuing learning and generalization with application in visually guided grasping of complex objects," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 117–127, 2004.

[11] M. Skubic and R. Volz, "Acquiring robust, force-based assembly skills from human demonstration," in *IEEE Transactions on Robotics and Automation*, vol. 16:6, 2000, pp. 772 –781.

[12] A. Billard and S. Schaal, "A connectionist model for on-line learning by imitation," in *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, hawaii, 2001.

[13] H. Bekkering, A. Wohlschlger, and M. Gattis, "Imitation of gestures in children is goal-directed," *Quarterly Journal of Experimental Psychology*, vol. 53A (1), pp. 153–164, 2000.

[14] S. Calinon and A. Billard, "Stochastic gesture production and recognition model for a humanoid robot," in *Proceedings of the IEEE/RSJ Intl Conference on Intelligent Robots and Systems (IROS)*, Sendai, Japan, September 28 - October 2 2004, pp. 2769–2774.

[15] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77:2, pp. 257–285, February 1989.

[16] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, 1977.

To appear in: Proceedings of the IEEE International Conference on
Robotics and Automation (ICRA), 2006

On Learning the Statistical Representation of a Task and Generalizing it to Various Contexts

Sylvain Calinon, Florent Guenter and Aude Billard
LASA Laboratory, School of Engineering (EPFL), CH-1015 Lausanne, Switzerland
{*sylvain.calinon,florent.guenter,aude.billard*}*@epfl.ch*

***Abstract*— This paper presents an architecture for solving generically the problem of extracting the relevant features of a given task in a programming by demonstration framework and the problem of generalizing the acquired knowledge to various contexts. We validate the architecture in a series of experiments, where a human demonstrator teaches a humanoid robot simple manipulatory tasks. Extracting the relevant features of the task is solved in a two-step process of dimensionality reduction. First, the combined joint angles and hand path motions are projected into a generic latent space, composed of a mixture of Gaussians (GMM) spreading across the spatial dimensions of the motion. Second, the temporal variation of the latent representation of the motion is encoded in a Hidden Markov Model (HMM). This two-step probabilistic encoding provides a measure of the spatio-temporal correlations across the different modalities collected by the robot, which determines a *metric of imitation performance*. A generalization of the demonstrated trajectories is then performed using Gaussians Mixture Regression (GMR). Finally, to generalize skills across contexts, we compute formally the trajectory that optimizes the metric, given the new context and the robot's specific body constraints.**

I. Introduction

Recent advances in *Robot Programming by Demonstration* (RbD), also referred to as *Learning by Imitation*, have identified a number of key issues that need to be solved for ensuring a generic approach to transferring skills across various agents and situations. These have been formulated as a set of generic questions, namely *what-to-imitate*, *how-to-imitate*, *when-to-imitate* and *who-to-imitate* [1]. These questions were formulated in response to the large body of work in RbD that emphasized ad-hoc solutions to sequencing and decomposing complex tasks into *known* sets of actions performable by both the demonstrator and the imitator, see e.g. [2]–[4]. In contrast to these other works, the above four questions and their solutions aim at being generic in the sense of making no or little assumptions on the type of skills that may be transmitted.

Recent work in RbD addresses these questions at different levels [5]. One trend aims at extracting and encoding low-level features, e.g. primitives of motions in joint space [6]–[9] and makes only weak assumptions as to the form of the primitives or kernels used to encode the motion. In contrast, another body of works stresses the need of introducing prior knowledge in the way information is encoded to achieve fast and reusable learning in imitation of higher level features, such as complete actions, tasks and behaviors [10]–[12]. In this paper, we draw from the two approaches. We start from acquiring a low-level representation of the task, by learning a spatio-temporal statistical encoding of the joint motions. We, then, show that such generic representation can be used to decompose and reconstruct sequences of simple manipulatory motions. Finally, we show formally how such a statistical representation of the motions can be combined with classical solutions to the inverse kinematics problem, to find an optimal reconstruction of the motions and adapt it to various contexts.

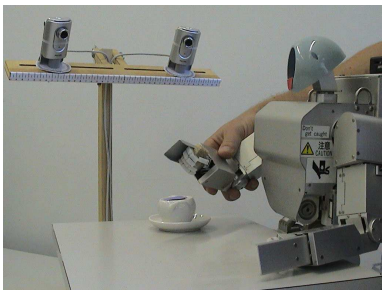


Fig. 1. Experimental setup. The robot is taught a new task (to bring a piece of sugar to its mouth) through kinesthetics, i.e. by the user moving the robot's arm through the steps of the task. A fixed pair of cameras tracks the 3-D position of the target (a color patch), here, the robot's mouth.

II. The experimental scenario

This paper presents an architecture for solving generically the problems of extracting the relevant features of a given task ("what to imitate") and the problem of generalizing the acquired knowledge to various contexts (part of the *how-to-imitate* issue [1]).

Fig. 1 presents the setup, while Fig. 2 illustrates these issues in a *Chess Task*. The task consists in grabbing the White Queen and moving it from position D8 to position D6. Left picture shows the path followed by the hand of the robot during training when starting from two different initial locations. In order to extract the relevance of each feature of the demonstration (i.e. to determine *what-to-imitate*), the robot computes the spatio-temporal variations and correlations across the variables. In the *Chess Task*, this analysis will reveal weak correlations at the beginning of the motion, as there is a large set of possible paths to reach for the Queen, depending on the hand's initial position. In contrast, the analysis will measure strong spatio-temporal correlations for, first, grabbing the piece and, then, pushing it toward the desired location without hitting the other pieces lying on the chessboard.

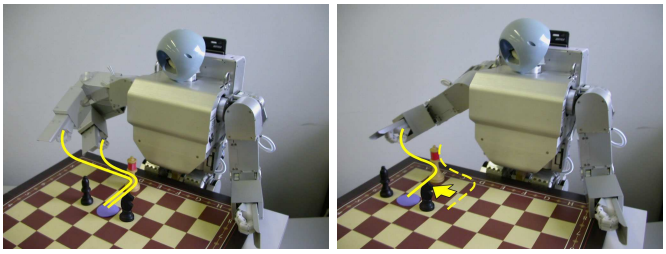


Fig. 2. *Left:* Illustration of the *what-to-imitate* issue. With two demonstrations of a *Chess Task*, we see that the constraints are varying along the motion. To approach the chess piece, a large set of paths are possible, depending on the initial position. Grabbing and pushing the piece requires higher constraints, i.e. the paths do not change much between two consecutive demonstrations. *Right:* Illustration of the *how-to-imitate* issue. To re-use the learned skill in different situation (here, different position of the chess piece), a cost function is used to select a controller that best fulfills the task constraints along the trajectories.

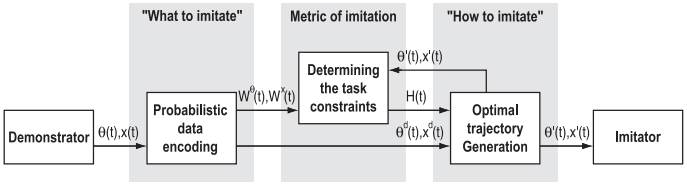


Fig. 3. Information flow across the complete system.

Fig. 2 right illustrates the *how-to-imitate* issue. Once trained to perform a particular task in a particular context, the robot must be able to generalize and to reproduce the same task in a different context. Here, the robot must be able to grab and place the White Queen between the Black Knight and Bishop, wherever these may be located on the chess board. However, joint angles and hand position can be mutually exclusive in the imitator workspace, i.e. both constraints can not be fulfilled at the same time. Depending on the situation, the robot may have to find a very different joint angles configuration than that first demonstrated, in order to avoid breaking its arm. To do so, the robot computes the trajectory that finds a optimal trade-off between satisfying the constraints of the task (spatio-temporal correlations across the variables) and its own body constraints[1].

III. The Architecture

Fig. 3 gives an overview of the input-output flow through the complete model. The model is composed of the following processes:

- *Probabilistic data encoding:* The signals are encoded in a two-stage process: First, we determine the latent space of the motions, by estimating the optimal *Gaussian Mixture Model* (GMM) to encode the motions. Second, we encode the dynamics of the motions (i.e. the transition across the states of the GMM), using a *Hidden Markov Model*.
- *Constructing the metric:* We then compute a time-dependent measure of the relative importance of each variable, and the dependencies across the variables, using the probabilistic description of the task. This measure is used to determine a *metric* of imitation performance for the particular task.
- *Optimal trajectory generation:* We then compute (using Lagrange optimization) the trajectory that optimizes the metric for a given context, given a set of robot's body constraints.

[1]Note that we assume that all demonstrations are relevant, and, thus, we do not address the *who-to-imitate* question.

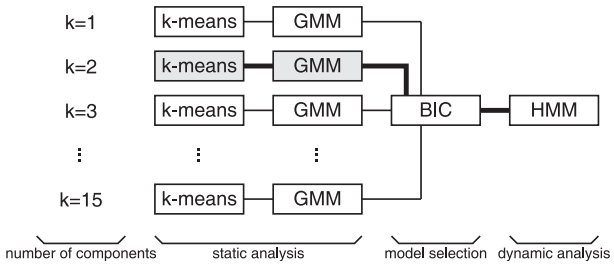


Fig. 4. Schematic of the *probabilistic data encoding* process. The selection of the number of components required to encode a task is performed in the static phase (estimation of several *Gaussian Mixture Models*). The dynamic analysis is then provided by a single estimation of *Hidden Markov Model*.

Next, we describe the computation carried out in each of these modules.

A. Probabilistic data encoding

To avoid making too many assumptions on the spatio-temporal variability of the dataset, we should, ideally, use a HMM with the most general architecture, such as a fully-connected continuous HMM, with full covariance matrix, describing the output variables distribution. However, using such a model requires the estimation of a large set of parameters, which can be achieved only with a large dataset. However, to program efficiently a robot by demonstration, the demonstrator should not have to perform more than a few (5 to 10) demonstrations. This means that the set of parameters to learn is quite large, compared to the amount of training data.

The standard *Expectation-Maximization* (EM) algorithm used to estimate the HMM parameters starts from initial estimates, and converges to the nearest local maximum of the likelihood function. Thus, initialization highly affects the model performance. To better estimate the state distribution of the HMM, we follow [13] and perform first a rough clustering of the data, using *k-means* clustering. Next, we estimate a *Gaussian Mixtures Model* (GMM) by *Expectation-Maximization* (EM), using the *k-means* clusters at initialization. Finally, the dynamics, i.e. transitions across states, are encoded in a HMM, with the GMM state variable distribution (see Fig. 4).

1) ***Gaussian Mixture Model (GMM):*** A dataset of N D-dimensional datapoints[2] $X = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ with $\vec{x}_n \in$

[2]The process is applied indifferently to joint angles and hand position

$\mathbb{R}^D$ is modelled by a multivariate Gaussian mixture of K-components (see e.g. [14]):

$$f(\vec{x}_n) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\vec{x}_n; \vec{\mu}_k, \mathbf{\Sigma}_k)$$

where π_k is the prior probability on the Gaussian component k, and $\mathcal{N}(\vec{x}_n; \vec{\mu}_k, \mathbf{\Sigma}_k)$ is the D-dimensional Gaussian density of component k:

$$\mathcal{N}(\vec{x}_n; \vec{\mu}_k, \mathbf{\Sigma}_k) = \frac{1}{\sqrt{(2\pi)^D |\mathbf{\Sigma}_k|}} \, e^{-\frac{1}{2}(\vec{x}_n - \vec{\mu}_k)^T \mathbf{\Sigma}_k^{-1} (\vec{x}_n - \vec{\mu}_k)}$$

$\vec{\mu}_k$ and $\mathbf{\Sigma}_k$ are, respectively, the mean and covariance matrix of the multivariate Gaussian k. $\{\pi_k, \vec{\mu}_k, \mathbf{\Sigma}_k\}$ are estimated using the *Expectation-Maximization* (EM) algorithm, i.e. we compute iteratively until convergence:

$$\begin{aligned} P(k|\vec{x}_n) &= \frac{\pi_k \mathcal{N}(\vec{x}_n; \vec{\mu}_k, \mathbf{\Sigma}_k)}{\sum_{k=1}^{K} \pi_k \mathcal{N}(\vec{x}_n; \vec{\mu}_k, \mathbf{\Sigma}_k)} && (\mathrm{E-step}) \\ \\ \pi_k &:= \frac{1}{N} \sum_{n=1}^{N} P(k|\vec{x}_n) \\ \vec{\mu}_k &:= \frac{\sum_{n=1}^{N} P(k|\vec{x}_n) \vec{x}_n}{\sum_{n=1}^{N} P(k|\vec{x}_n)} && (\mathrm{M-step}) \\ \mathbf{\Sigma}_k &:= \frac{\sum_{n=1}^{N} P(k|\vec{x}_n)(\vec{x}_n - \vec{\mu}_k)(\vec{x}_n - \vec{\mu}_k)^T}{\sum_{n=1}^{N} P(k|\vec{x}_n)} \end{aligned}$$

*2) **Model selection**:* The optimal number of components K in a model may not be known beforehand. We need to determine a trade-off between optimizing the model's likelihood (a measure of how well the model fits the data) and minimizing the parameters (i.e. the number of states used to encode the data).

To determine the number of states in a HMM, heuristic methods are often used, sometimes not adequately tuned for HMM. In our approach, model selection is performed in the GMM initialization phase. Multiple Gaussian Mixture Models are estimated, the best model is selected, and a single HMM estimation is performed.

We are using the *Bayesian Information Criterion* (BIC) [15] to select the optimal number of components K:

$$\begin{aligned} S_{BIC} &= -\mathcal{L} + \frac{n}{2} \, log(N) \\ n &= (K-1) + K \times \left(D + \frac{D \times (D+1)}{2} \right) \end{aligned}$$

where $\mathcal{L}$ is the log-likelihood of the model, n is the number of free parameters required for a mixture of K components with full covariance matrix. N is the number of D-dimensional datapoints. The first term of the equation measures how well the model fits the data, while the second term is a penalty factor that aims at keeping the total number of parameters low. In our experiments, we compute a set of candidate GMMs with up to 15 states and keep the model with the minimum score (see Fig. 4).

*3) **Hidden Markov Model (HMM)**:* Similarly to Gaussian Mixture Models, Hidden Markov Models use a mixture of multivariate Gaussians to describe the distribution of the data. The difference is that HMM also encapsulate the transitions probabilities between the Gaussians. It offers, thus, a way of describing probabilistically the temporal variations of the data[3].

Let $\{\mathbf{\Pi}, \mathbf{A}, \mathbf{B}\}$ be, respectively, the initial state distribution, the transition probabilities between the states (or components), and the multivariate output data distribution. In our experiments, we compute only $\{\mathbf{\Pi}, \mathbf{A}\}$ by *Baum-Welch* and set $\mathbf{B} = \{\vec{\mu}_k, \mathbf{\Sigma}_k\}_{k=1}^{K}$, where $\{\vec{\mu}_k, \mathbf{\Sigma}_k\}_{k=1}^{K}$ are the state distributions learned by the GMM (see section III-A.1).

Once trained, the HMM can be used to recognize gestures. In our experiments, this is used to decide whether a new demonstration belongs or not to the same task, following an approach similar to [9]. In order to measure the similarity between a new gesture and the ones encoded in the model, we run the *forward-algorithm*, an iterative procedure to estimate the likelihood that the observed data could have been generated by the model.

*4) **Gaussian Mixture Regression (GMR)**:* To reconstruct a signal from the GMM/HMM encoding, after training and generalization over the demonstrations, we apply a *Gaussian Mixture Regression* (GMR), see e.g. [17]. For a D-dimensional variable $\vec{x} \in \mathbb{R}^D$, the means and covariance matrices given by the GMM/HMM representation for component k are given by:

$$\begin{aligned} \vec{\mu}_{kX}^{H} &= \{\mu_{kx_1}^{H}, \mu_{kx_2}^{H}, \ldots, \mu_{kx_D}^{H}\} \\ \mathbf{\Sigma}_{kX}^{H} &= \begin{pmatrix} \Sigma_{kx_1}^{H} & \Sigma_{kx_1x_2}^{H} & \cdots & \Sigma_{kx_1x_D}^{H} \\ \Sigma_{kx_2x_1}^{H} & \Sigma_{kx_2}^{H} & \cdots & \Sigma_{kx_2x_D}^{H} \\ \vdots & \vdots & \ddots & \vdots \\ \Sigma_{kx_Dx_1}^{H} & \Sigma_{kx_Dx_2}^{H} & \cdots & \Sigma_{kx_D}^{H} \end{pmatrix} \end{aligned}$$

The regression is done along the time index. We compute the means and covariance matrices of the set of observations $\{t, \vec{x}(t)\}$ with dimension $(D+1)$. Note that sole the time-indexed covariances matrices and means are estimated, since the rest of the means and covariance matrixes $\{\vec{\mu}_{kX}^{H}, \mathbf{\Sigma}_{kX}^{H}\}$ have already been estimated:

$$\begin{aligned} \vec{\mu}_k^R &= \{\mu_{kt}^R, \mu_{kx_1}^H, \mu_{kx_2}^H, \ldots, \mu_{kx_D}^H\} \\ \mathbf{\Sigma}_k^R &= \begin{pmatrix} \Sigma_{kt}^R & \mathbf{\Sigma}_{ktX}^R \\ \mathbf{\Sigma}_{kXt}^R & \mathbf{\Sigma}_{kX}^H \end{pmatrix} \\ &= \begin{pmatrix} \Sigma_{kt}^R & \Sigma_{ktx_1}^R & \Sigma_{ktx_2}^R & \cdots & \Sigma_{ktx_D}^R \\ \Sigma_{kx_1t}^R & \Sigma_{kx_1}^H & \Sigma_{kx_1x_2}^H & \cdots & \Sigma_{kx_1x_D}^H \\ \Sigma_{kx_2t}^R & \Sigma_{kx_2x_1}^H & \Sigma_{kx_2}^H & \cdots & \Sigma_{kx_2x_D}^H \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \Sigma_{kx_Dt}^R & \Sigma_{kx_Dx_1}^H & \Sigma_{kx_Dx_2}^H & \cdots & \Sigma_{kx_D}^H \end{pmatrix} \end{aligned}$$

[3]People unfamiliar with HMM should refer to [16]

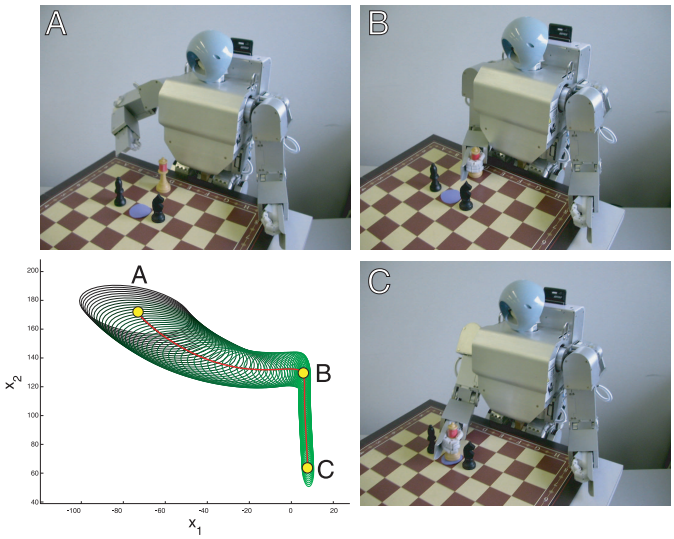


Fig. 5. *A,B,C:* Snapshots of 3 steps during the reproduction of the *"Chess Task"*. *Bottom left:* in solid line, projection along the 2 first dimension of the latent space of the desired trajectory $\vec{x}^d(t)$ for the robot's hand. The continuous time-dependent weight matrix $\mathbf{W}^x(t)$ is depicted as a superposition of ellipses, representing the task constraints along the trajectory.

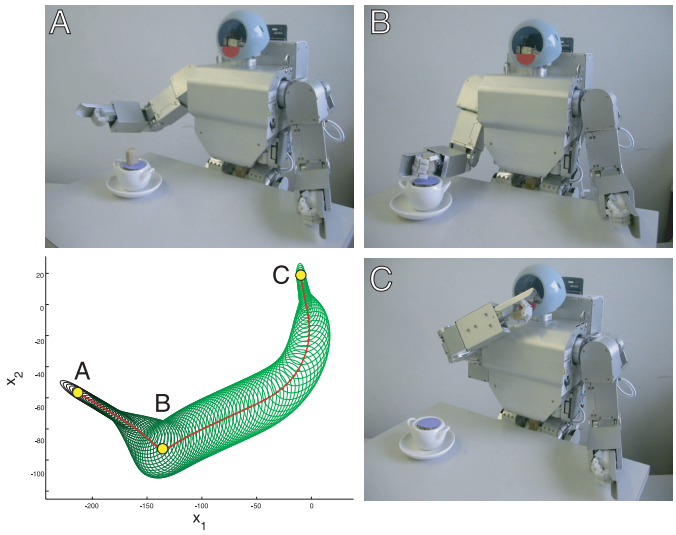


Fig. 6. Probabilistic encoding of the *"Sugar Task"* (bottom left), by projecting along the first 2 dimensions of $\vec{x}^d(t)$. The continuous generalized trajectory $\vec{x}^d(t)$ is represented in solid line, with associated weight matrix $\mathbf{W}^x(t)$ depicted as a superposition of ellipses.

The *Gaussian Mixture Regression* estimates:

$$\begin{aligned}\vec{x}^d(t) &= \sum_{k=1}^{K} w_k(t)\vec{x}^d_k(t)\\ w_k(t) &= \frac{\pi_k \mathcal{N}(t;\mu^R_{kt},\Sigma^R_{kt})}{\sum_{k=1}^{K}\pi_k\mathcal{N}(t;\mu^R_{kt},\Sigma^R_{kt})}\\ \vec{x}^d_k(t) &= \vec{\mu}_{kX} + \mathbf{\Sigma}^R_{kXt}{\Sigma^R_{kt}}^{-1}(t-\mu_{kt})\end{aligned}$$

$\vec{x}^d_k(t)$ are the regression output for each associated Gaussian component k and $w_k(t)$ the corresponding weight, that measures the relative influence of the Gaussian component k. π_k is the prior probability on the Gaussian component k. Finally, $\vec{x}^d(t)$ is the *desired trajectory*, a generalized form of the motion learned during training, which will be used during the reconstruction, as we explain next.

B. Constructing the metric

[18] proposed a general formalism for determining the cost function of an imitation task. Here, we extend this work and consider a time-dependent version of the original cost function H. This generic cost function measures the variations of the constraints and of the dependencies across the variables over time. It is continuous, positive, definite and can be estimated at any time along the trajectory, by interpolating between the finite set of Gaussians used to encode the trajectories.

Let $\{\vec{\theta}^d(t), \vec{x}^d(t)\}$ be the desired trajectories for the joints and hand path, generalized forms of the signals gathered during the demonstrations. Let $\{\vec{\theta}(t), \vec{x}(t)\}$ be the candidate trajectories for reproducing the motions. The metric of imitation performance (cost function for the task) H is given by:

$$\begin{aligned}H &= H(\vec{\theta}^d(t),\vec{\theta}(t),\vec{x}^d(t),\vec{x}(t))\\ &= \left(\vec{\theta}(t)-\vec{\theta}^d(t)\right)^T \mathbf{W}^\theta(t) \left(\vec{\theta}(t)-\vec{\theta}^d(t)\right)\\ &+ \left(\vec{x}(t)-\vec{x}^d(t)\right)^T \mathbf{W}^x(t) \left(\vec{x}(t)-\vec{x}^d(t)\right) \qquad (1)\end{aligned}$$

H=0 corresponds to a perfect reproduction. The diagonal elements of the $\mathbf{W}^\theta(t)$ (4×4 matrix) and $\mathbf{W}^x(t)$ (3×3 matrix) matrices give a measure of the relative importance of each set of variables, while the other elements give a measure of the correlations across the joint angles and the 3D hand position. These matrices are defined as:

$$\begin{aligned}\mathbf{W}^x(t) &= \left(\frac{1}{K}\sum_{k=1}^{K} p\left(k|\vec{x}^d(t)\right)\ \mathbf{\Sigma}^H_{kX}\right)^{-1}\\ \mathbf{W}^\theta(t) &= \left(\frac{1}{K}\sum_{k=1}^{K} p\left(k|\vec{\theta}^d(t)\right)\ \mathbf{\Sigma}^H_{k\Theta}\right)^{-1}\end{aligned}$$

$\mathbf{W}^x(t)$ (respectively $\mathbf{W}^\theta(t)$) is the inverse of a normalized sum of the covariance matrices in the GMM/HMM representation. $p\left(k|\vec{x}^d(t)\right)$ (respectively $p\left(k|\vec{\theta}^d(t)\right)$) weights the influence of each component k in the generalized position $\vec{x}^d(t)$ at time t (respectively $\vec{\theta}^d(t)$)[4]. As $\vec{\theta}^d(t)$ and $\vec{x}^d(t)$ can be estimated continuously by regression, $\mathbf{W}^x(t)$ and $\mathbf{W}^\theta(t)$ can also be estimated continuously. Fig. 5 and 6 show the generalized trajectories and associated weight matrix. Note that, for clarity in the notation, we will omit the vectorial notations and the time index for the rest of the developments.

[4]In practice, a filtered version of p was used, by applying a local regression smoothing process. This insures to have smoother transitions between covariance matrix

C. Optimal trajectory generation

Once the cost function and the relative influence of each constraint have been determined, we generate a trajectory that is optimal with respect to the cost function H. We must, however, take into account the body constraints of the robot and ensure that the desired joint trajectories are consistent with the desired Cartesian trajectories. Such coherence can be fulfilled by solving the forward kinematics equations of the robot's arm.

If we define the coordinates of a robot's manipulator as the n-dimensional vector of joint angles $\vec{\theta}$ and the position of the m-dimensional vector $\vec{x}$, the forward kinematics is given by: $\vec{x} = f(\vec{\theta})$. In classical control theory, the *inverse kinematics* usually refers to the inverse computation required to determine the position of each of the robot's joints for a given location of the robot's end-effector (usually its arm). The inverse kinematics is given by

$$\vec{\theta} = f^{-1}(\vec{x}) \tag{2}$$

where f is a continuous function ($f \in \mathbb{R}$).

When the problem is under-constrained, i.e. when the number of degrees of freedom n exceeds the number of given variables m, i.e. $n > m$, e.g. when controlling a 4 degrees of freedom robot arm given the 3D position of the target, (2) may have no solutions (degenerate case) or multiple solutions. Several solutions to this problem have been proposed, ranging from numerical solutions [19], [20], to geometrical solutions [21], [22]. Moreover, in order to deal with the non-linearity of the transformation, various methods, based on multiple locally linear models, have also been explored, using either neural networks [23] or regression techniques [24]. Each solution has its advantages and drawbacks. In this paper, we redevelop the pseudo-inverse with optimization method proposed in [19] (a locally linear approximation) and adapt its form to optimize our global cost function H.

Following [19], we consider an iterative, locally linear, solution to this equation, such that:

$$\dot{x} = \mathbf{J} \cdot \dot{\theta} \tag{3}$$

where $\dot{\theta}(t) = \theta(t) - \theta(t-1)$ and $\dot{x}(t) = x(t) - x(t-1)$ are the velocity vectors of the joint angles and the hand path. $\mathbf{J}$ is the Jacobian (4×3 matrix).

Similarly, by substituting $c_1(t) = \theta^d(t) - \theta(t-1)$ and $c_2(t) = x^d(t) - x(t-1)$ in (1), we obtain:

$$\begin{aligned} H(\dot{\theta}, \dot{x}) &= (\dot{\theta} - c_1)^T \, \mathbf{W}^\theta \, (\dot{\theta} - c_1) \\ &+ (\dot{x} - c_2)^T \, \mathbf{W}^x \, (\dot{x} - c_2) \end{aligned} \tag{4}$$

The problem is now reduced to finding a minimum of (4) when subjected to (3). Since H is a quadratic function, the problem can be solved analytically by Lagrange optimization. We define the *Lagrangian* as:

$$\begin{aligned} L(\dot{\theta}, \dot{x}, \lambda) &= (\dot{\theta} - c_1)^T \, \mathbf{W}^\theta \, (\dot{\theta} - c_1) \\ &+ (\dot{x} - c_2)^T \, \mathbf{W}^x \, (\dot{x} - c_2) \\ &+ \lambda^T (\dot{x} - \mathbf{J} \, \dot{\theta}) \end{aligned} \tag{5}$$

where λ is the vector of associated Lagrange multipliers. We compute $\nabla L(\dot{\theta}, \dot{x}, \lambda) = 0$, i.e.:

$$\frac{\partial L(\dot{\theta}, \dot{x}, \lambda)}{\partial \dot{\theta}} = 0 \; , \; \frac{\partial L(\dot{\theta}, \dot{x}, \lambda)}{\partial \dot{x}} = 0 \; , \; \frac{\partial L(\dot{\theta}, \dot{x}, \lambda)}{\partial \lambda} = 0 \tag{6}$$

Deriving along λ, we find again (3). Deriving along $\dot{\theta}$, we get:

$$-2W^\theta(\dot{\theta} - c_1) - \mathbf{J}^T \lambda = 0 \tag{7}$$

Deriving along $\dot{x}$, we get:

$$-2W^x(\dot{x} - c_2) + \lambda = 0 \tag{8}$$

Using (3),(7) and (8), we find:

$$W^\theta(\dot{\theta} - c_1) + \mathbf{J}^T W^x (\mathbf{J} \, \dot{\theta} - c_2) = 0$$

Solving for $\dot{\theta}$, we obtain:

$$\dot{\theta} = (\mathbf{W}^\theta + \mathbf{J}^T \mathbf{W}^x \mathbf{J})^{-1} (\mathbf{W}^\theta c_1 + \mathbf{J}^T \mathbf{W}^x c_2)$$

We can then recompute the joint angle trajectories, using $\theta(t) = \theta(t-1) + \dot{\theta}(t)$. The final gesture $\{\vec{\theta}'(t), \vec{x}'(t)\}$ is computed and evaluated using this method, and reproduced by the robot.

IV. Experiments

We conducted two experiments to demonstrate the validity of our model for teaching a humanoid robot simple manipulatory tasks. The tasks consisted in *"moving a chess piece, on a chessboard, while avoiding other pieces"* (see Fig. 2 and 5), and *"bringing a piece of sugar to the mouth"* (see Fig. 1 and 6). Note that, in these experiments, control affected only the 4 degrees of freedom of the arm and the opening and closing of the robot's hand was hard-coded and activated by the user. In each case, the robot was shown the task five times. Once trained, the robot was required to reproduce each task under different constraints, by placing the obstacles and goals at different locations in the robot's workspace. This procedure aimed at demonstrating the robustness of the system when the constraints were transposed to different locations within the robot's workspace.

A. Experimental setup

The experiments were conducted with a Fujitsu HOAP-2 humanoid robot with 25 degrees of freedom (DOF), of which only the 4 DOFs of the right arm were required in the experiments. The remaining DOFs of the torso and legs were set to a constant position, so as to support the robot in an upright posture, facing a table, see Fig. 2.

A color-based stereoscopic vision system tracks the 3D-position of the different objects used in the experiments at a rate of 15Hz, with an accuracy of 10mm. The system uses two Phillips web-cams with a resolution of 320x240 pixels, see Fig. 1. The tracking is based on color segmentation for detecting the skin color and the color of the different objects in the YCbCr color space[5].

[5]Only Cb and Cr are used, to be robust to changes in luminosity

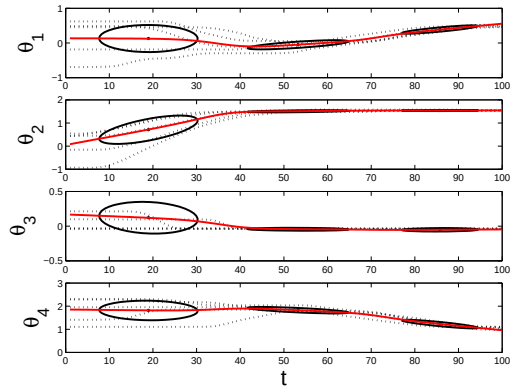


Fig. 7. *Dotted lines:* Trajectories of the joints during the five demonstrations of the *Chess Task*, see Fig. 5. *Straight Line:* Desired trajectory, a generalized form of the demonstrated signals, reconstructed through regression over the mixture of Gaussians (superimposed as ellipses) estimated by the GMM/HMM.

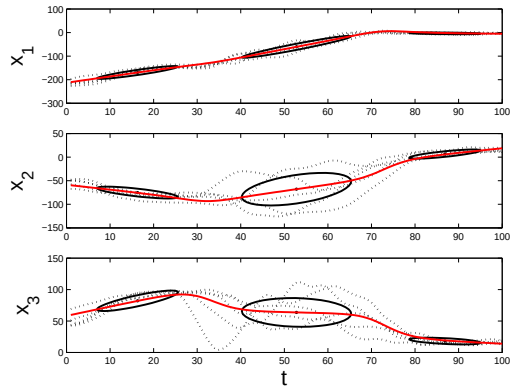


Fig. 8. *Dotted lines:* Hand path during the five demonstrations of the *Sugar Task*, see Fig. 6. *Straight Line:* Desired trajectory, a generalized form of the demonstrated signals, reconstructed through regression over the mixture of Gaussians (superimposed as ellipses) estimated by the GMM/HMM.

In the experiments reported here, the robot was taught through kinesthetics, i.e. by the demonstrator moving its two arms through each of the task's steps. To achieve this, the robot's motors were set in a passive mode, whereby each limb could be moved by the human demonstrator. The kinematics of each joint motions was recorded at a rate of 1000Hz during the demonstration and was, then, downrated to 15Hz to match the tracking rate of the cameras.

B. Experimental results

Fig. 7 shows the encoding and reconstruction of the joint trajectories for the *Chess Task*. Three states were found to encode optimally the trajectories, according to the BIC criterion, see Section III-A.2. These correspond roughly to the different phases of the motion (approaching the object, adjusting the hand posture and pushing the object). When looking at the width of the ellipses (which represent the variance along the joint trajectories across the demonstrations), we observe that the first phase of the motion shows a lot of variability (and, thus, does not require to follow a precise path), whereas the

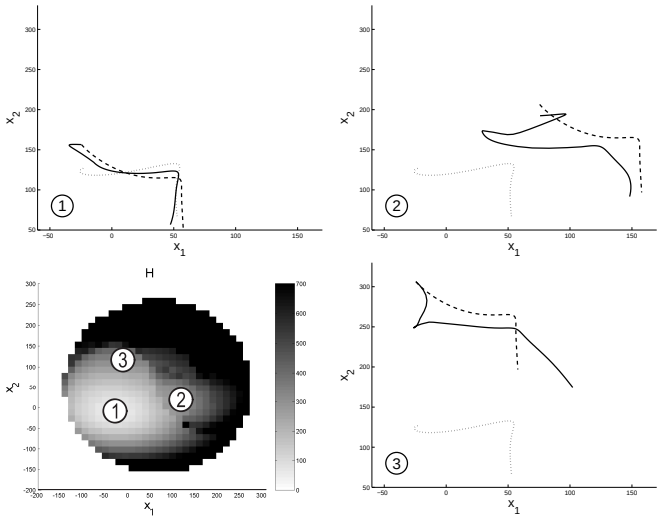


Fig. 9. *Bottom left:* Mean values taken by the cost function H for the *Chess Task* for various final locations of the White Queen on the chess board. *1,2,3:* Reconstructed hand paths $\vec{x}'(t)$ for the corresponding three locations on the workspace. The solid lines represent the hand path $\vec{x}'(t)$ reproduced by the robot, the dashed lines the desired hand path $\vec{x}^d(t)$, with respect to the goal and the dotted lines the path resulting from the desired joint angles, i.e. $f(\vec{\theta}^d(t))$.

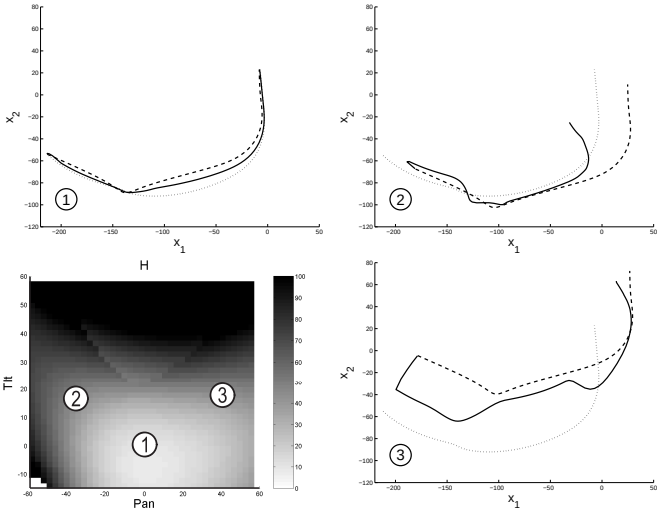


Fig. 10. *Bottom left:* Mean values taken by the cost function H for the *Sugar Task* for various pan/tilt orientation of the head (see Fig. 9 for additional comments).

two other phases are much more precise. This change in the constraints along the task will be encapsulated in the weights $W^{\theta}(t)$ of the cost function H.

Three states were also found to encode efficiently the *Sugar Task*, depicting the different phases (grabbing the piece of sugar, approaching the mouth, putting the piece of sugar in the mouth), see Fig. 8. Since the position of the piece of sugar and of the mouth do not vary across demonstrations, it results in highly constrained trajectories, which are reflected by the narrow ellipses at the beginning and at the end of the motion.

Fig. 9 and 10 show the mean values of the cost function H,

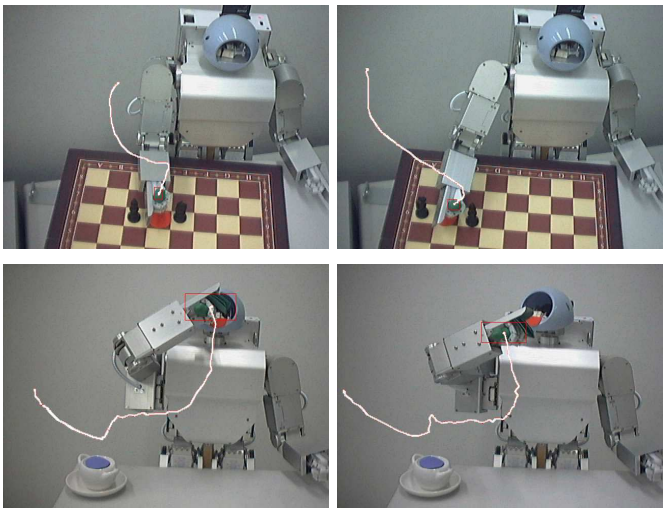


Fig. 11. Experimental results for the *Chess Task* and *Sugar Task*, when performed under different situations (different chess piece position and different head pan/tilt). The trajectory of the hand was tracked by the vision system, and superimposed to the image.

when reconstructing the motions for different locations of the target (end-position of the White Queen or Robot's mouth). The colored regions represent the target locations for which the system finds a solution (i.e. locations in space that are within the robot's workspace). Fig. 9, 10 and 11 show the path followed by the robot's hand for different locations of the target in the workspace. In each case, the system finds a correct solution, that does not hit the obstacles (in the *Chess Task*) and reaches correctly the desired locations for the White Queen, the piece of sugar or the mouth. However, the farther the target is from its original position, the higher the cost function and the less straight the path. This is expected and entirely acceptable, since achieving the high-level goals of the task (avoiding the obstacles and reaching for the targets) takes precedence over any other classical control constraint, such as running in a straight trajectory.

V. Discussion and Conclusion

This paper presented a method to: 1) extract the important features, i.e. the spatio-temporal correlations across the multivariate dataset of the task, 2) to determine a generic metric to evaluate the robot's imitation performance, and, finally, 3) to optimize the robot's reproduction of the task when placed in a new context according to the task metric. The method was validated in two experiments where a robot was taught simple manipulation tasks.

We showed that, in each case, the robot managed to adapt its motions correctly, so as to reproduce the important qualitative features of each task, namely grabbing and pushing the chess piece to the correct location, grabbing the piece of sugar and bringing it to its mouth. However, none of these high-level goals were explicitly represented in the robot's control system, but, were nevertheless correctly extracted by our probabilistic system.

The system we presented for solving the *what-to-imitate* and *how-to-imitate* issues is generic, in the sense that it makes no assumption on the robot's configuration (number of degrees of freedom and length of segments). As first stressed out by Nehaniv and colleagues [25], there is a multitude of correspondence problems, when trying to transfer skills across various agents and situations. One may consider:

- *different embodiments:* the demonstrator does not share the same morphology and characteristics as the imitator.
- *different situations:* the environment and constraints during the demonstration and the reproduction are different

In the experiments presented in this paper, the robot was being taught through kinesthetics. By showing kinesthetically how to perform a task, the user "embodies" the robot's body. Thus, this way, we simplified the correspondence problem and overlooked the problem of having different embodiments. However, by testing the system in different situations than those taught, we tackled the second aspect of the correspondence problem.

Note that, in the experiments reported here, we assumed implicitly that the kinematics of joint angle trajectories and hand path is sufficient to describe the skill, and that dynamics is of less importance. This may not be true and taking into account the forces applied on the object may certainly be very important in certain task. However, it is very likely that these are not learned through imitation, but, through more generic motor learning processes.

References

[1] C. Nehaniv and K. Dautenhahn, "Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications," in *Interdisciplinary Approaches to Robot Learning*, J. Demiris and A. Birk, Eds. World Scientific Press, 2000, vol. 24, pp. 136–161.

[2] M. Kaiser and R. Dillmann, "Building elementary robot skills from human demonstration," in *IEEE Intl Conference on Robotics and Automation*, 1996.

[3] M. Skubic and R. Volz, "Acquiring robust, force-based assembly skills from human demonstration," in *IEEE Transactions on Robotics and Automation*, vol. 16:6, 2000, pp. 772 –781.

[4] M. Yeasin and S. Chaudhuri, "Toward automatic robot programming: learning human skill from visual data," in *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 30:1, 2000.

[5] A. Billard and R. Siegwart, "Robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 65–67, 2004.

[6] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 15, 2002, pp. 1547–1554.

[7] A. Ude, C. Atkeson, and M. Riley, "Programming full-body movements for humanoid robots by observation," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 93–108, 2004.

[8] A. Shon, K. Grochow, and R. Rao, "Robotic imitation from human motion capture using gaussian processes," in *IEEE-RAS International Conference on Humanoid Robots*, 2005.

[9] S. Calinon and A. Billard, "Recognition and reproduction of gestures using a probabilistic framework combining pca, ica and hmm," in *Proceedings of the International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[10] R. Zöllner, M. Pardowitz, S. Knoop, and R. Dillmann, "Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Barcelona, Spain, April 18-22 2005.

[11] J. Zhang and B. Rössler, "Self-valuing learning and generalization with application in visually guided grasping of complex objects," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 117–127, 2004.
[12] J. J. Steil, F. Röthling, R. Haschke, and H. Ritter, "Situated robot learning for multi-modal instruction and imitation of grasping," in *Robotics and Autonomous Systems*, 2004, vol. 47:2-3, pp. 129 –141.
[13] M. Bicego and V. Murino, "Investigating hidden markov models' capabilities in 2d shape classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 281–286, February 2004.
[14] J. Verbeek, "Mixture models for clustering and dimension reduction," Master's thesis, University of Amsterdam, 2004.
[15] G. Schwarz, "Estimating the dimension of a model," *Annals of Statistics*, vol. 6, pp. 461–464, 1978.
[16] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77:2, pp. 257–285, February 1989.
[17] H. Sung, "Gaussian mixture regression and classification," Master's thesis, Rice University, 2004.
[18] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47:2-3, pp. 69–77, 2004.
[19] A. Liegeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, 1977.
[20] G. Tevatia and S.Schaal, "Inverse kinematics for humanoid robots," in *IEEE Intl Conference on Robotics and Automation*, 2000.
[21] T. Asfour and R. Dillman, "Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem," in *IEEE/RSJ Intl Conference on intelligent Robots and Systems*, 2003.
[22] X. Wang and J. Verriest, "A geometric algorithm to predict the arm reach posture for computer-aided ergonomic evaluation," *The Journal of Visualization and Computer Animation, vol. 9, pp. 33-47*, 1998.
[23] E. Oyama, A. Agah, K. MacDorman, T. Maeda, and S. Tachi, "A modular neural network architecture for inverse kinematics model learning," *Neurocomputing*, 2001.
[24] A. D'Souza, S. Vijayakumar, and S. Schaal, "Learning inverse kinematics," in *IEEE/RSJ Intl Conference on Intelligent Robots and Systems*, 2001.
[25] C. L. Nehaniv, "Nine billion correspondence problems and some methods for solving them," in *Proceedings of the International Symposium on Imitation in Animals & Artifacts*, 2003, pp. 93–95.

Imitating Using JABBERWOCKY to Achieve Corresponding Effects in Context

Aris Alissandrakis, Chrystopher L. Nehaniv, Kerstin Dautenhahn, and Joe Saunders

***Abstract*— This article presents JABBERWOCKY, a system that uses captured motion data of a human demonstrating a task to produce action commands that if executed by an imitating agent result in achieving corresponding effects. The system is able to generalize across dissimilar initial object configurations given the task sub-goal granularity and appropriate effect metrics, generating action commands targeted for multiple imitator platforms, both simulated in software and in hardware.**

Towards a characterization of the *space of effect metrics*, we explore relative/absolute angle and displacement aspects and focus on overall arrangement and trajectory of manipulated objects. From the examples shown it becomes apparent that the matching of effects is a more sophisticated issue than generally acknowledged. Using different (given) combinations of effect metrics to imitate the same demonstrated task illustrates the qualitatively different character of the resulting imitation behaviours and can in turn help to inform the development of algorithms for automatic metric extraction.

The use of the number of *exceptions* (here defined as events where an optimal displacement and/or rotation that minimize the dissimilarity metrics used to generate a corresponding imitative behaviour cannot be directly achieved in the particular context) is suggested as a possible performance measure by a robotic or software imitator of an object manipulation behaviour.

***Index Terms*— imitation and social learning, correspondence problem, space of effect metrics, human-robot interaction, programming by demonstration.**

I. INTRODUCTION

IMITATION is a powerful learning tool when humans and robots interact in a social context. Having a robot observe and learn to perform a task from an experienced teacher presents a more flexible and adaptive solution than explicit preprogramming and restrictive hardwiring. The learning process can be faster as no direct programming is required. The expert should, just by performing and thus demonstrating the task, pass the required knowledge to the robot which in turn may be used as a model to be imitated by other robots. Robotics researchers are inspired from imitation and social learning in animals and humans to create controllers for their autonomous robots, using suitable behaviours for adaptive learning [1]–[6].

A robotic companion at home could for example acquire knowledge of arranging some household objects on a table from observing its human owner. Acquiring such skills socially requires matching different aspects of the effects that the human actions have on objects in the environment. Also the various contexts within which a task is replicated might require its generalization to various settings (see examples below) and to other types and shapes of manipulated objects.

A fundamental problem when learning *how* to imitate is to create an appropriate (partial) mapping between the actions afforded by particular embodiments to achieve corresponding states and effects by the model and imitator agents (solving a *correspondence problem*) [7]. The solutions to the corresponding problem will depend to the sub-goal granularity and the metrics used to evaluate the similarity between actions, states and/or effects, resulting in qualitative different imitative behaviours [8], [9]. The related problem of *what* to imitate addresses the choice of metrics and sub-goal granularity that should be used for imitating, depending on the context.

The authors are with the School of Computer Science, Adaptive Systems Research Group, University of Hertfordshire, College Lane, Hatfield, Hertfordshire AL10 9AB, U.K. (e-mail: a.alissandrakis@herts.ac.uk).

II. EFFECT METRICS

Towards a characterization of the *space of effect metrics*, we explored absolute/relative angle and displacement aspects and focused on overall arrangement and trajectory of manipulated objects. Focusing on aspects of orientation and displacement of the manipulated objects, two types of effect metrics can be used, *displacement* and *angular*. The first type relates an object's movement and position on the workspace (e.g. *relative displacement*, *absolute position*, *relative position* or *mirror displacement*, see Fig. 1), and the second type the object's orientation (e.g. *rotation*, *orientation*, *mirror rotation* or *parallel orientation* effect metrics, see Fig. 2). Using these metrics, one can evaluate the similarity between the *effects* on the environment (object displacement and/or rotation) of the model and the imitator, without considering the *state* or the *actions* of the agents that caused them.

Depending on the solution to the *what to imitate* question, i.e. which metric(s) and sub-goal granularity an imitator should use, qualitatively dissimilar imitation behaviours can result, depending on the particular context (see Fig. 3).

Detailed definitions for these effect metrics are provided in the appendix.

III. THE JABBERWOCKY SYSTEM

In previous work we have developed ALICE (**A**ction **L**earning via **I**mitating **C**orresponding **E**mbodiments), a generic framework for solving the correspondence problem [8], [9]. The ALICE framework builds up a library of actions from the repertoire of an imitator agent that can be executed to achieve corresponding actions, states and/or effects to those of a model agent (according to given metrics and granularity).

The ALICE framework provides a functional architecture that informs the design of robotic systems that can learn

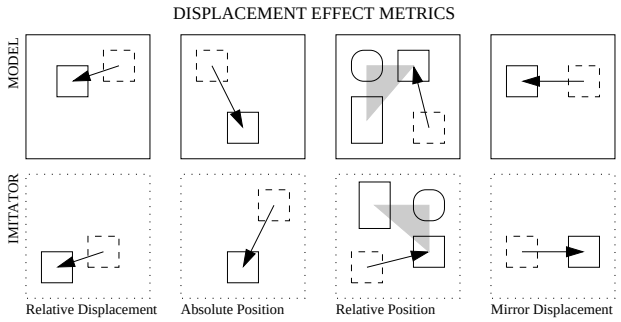


Fig. 1. **A selection of *displacement* effect metrics.** To evaluate the similarity between object displacements, the *relative displacement*, *absolute position*, *relative position* or *mirror displacement* effect metrics can be used. The second row shows the way the corresponding object (in a different workspace) needs to be moved (from dashed to solid outline) by an imitator to match the corresponding effects. The grey triangles are superimposed to show that for the *relative position* effect metric, the relative final positions of the objects are the same.

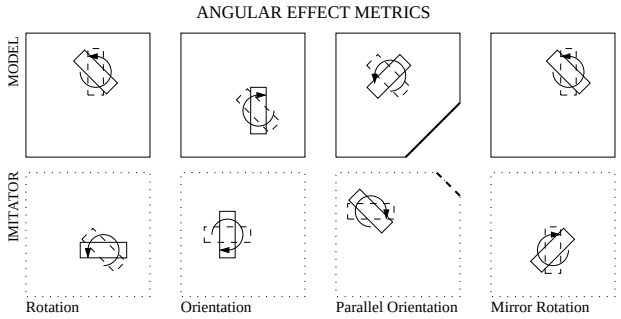


Fig. 2. **A selection of *angular* effect metrics.** To evaluate the similarity between object rotations, the *rotation*, *orientation*, *mirror rotation* or *parallel orientation* effect metrics can be used. The second row shows the way the corresponding object (in a different workspace) needs to be rotated (from dashed to solid outline) by an imitator to match the corresponding effects. Note that the shape of the two workspaces in the *parallel orientation* example is different, and the objects align with the highlighted diagonal edge.

socially from a human demonstrator. For the COGNIRON[1] project we are currently developing JABBERWOCKY, a system that uses captured data from a human demonstrator to generate appropriate action commands (see Fig. 4). The action commands can be targeted for various software and hardware platforms. These actions will allow the imitating agent to achieve corresponding actions, states and/or effects, depending on the given (relevant to the demonstrated task and context) metrics and granularity (provided by a *what to imitate* module), embodiment restrictions and constrains (imposed by the targeted imitator platform), and possibly different initial state of the objects in the environment.

The system bears some similarity to the one presented by Kuniyoshi et al. in [6], but with the main differences being that it is more flexible in that it can use any given metric and granularity and that it is designed to be able to generate action commands targeted for a variety of target platforms, both in software and hardware to match different aspects and achieve various types of social learning.

[1] `http://www.cogniron.org`

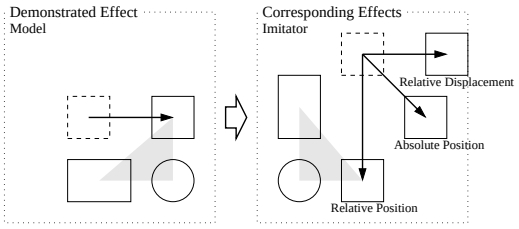


Fig. 3. Depending on the *effect metric(s)* used, qualitatively dissimilar imitation behaviours can result from dissimilar object configurations (here displacement only). The figure illustrates three different examples of displacement effect metrics. The grey triangles are superimposed to help visualize the relative positions of the three objects.

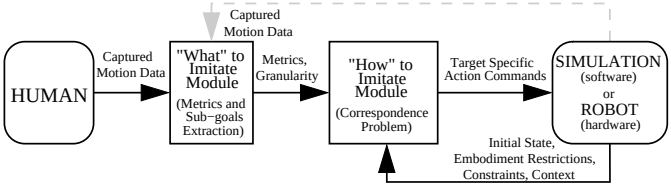


Fig. 4. **The JABBERWOCKY system architecture.** Using data captured from a human and given appropriate metrics and sub-goal granularity, the multi-target system can produce action command sequences that when executed by a software or hardware agent can achieve corresponding actions, states and/or effects. The corresponding actions, states and effects as demonstrated by the imitator can also be captured and used as a demonstration for another imitating agent. Differently embodied and constrained target systems in various contexts need to be supported.

A. Demonstrated Tasks

The system uses demonstrations of tasks, performed by human users. The entire task needs to be captured before the system can extract the metrics and sub-goal granularity.

Guided by the assumption that the manipulation of objects will be the most important aspect of the demonstrated behaviours that users would like a robotic companion to imitate in a home environment (e.g. fetching objects or arranging them in particular ways) the work is initially concentrated on *effects*. In the future, the system will be extended to consider also the *state* and *action* aspects of a demonstration.

In the work described in this article, the demonstrated task consists of the manipulation of three coloured block objects (red, green and blue) moved and rotated on a 2D workspace surface by a human acting as the demonstrator (see Fig. 5, left). Using the Polhemus LIBERTY^{TM} motion capture system, a sensor is attached on top of each object, giving the position of the object's center and also the object's orientation, relative to one of the workspace corners (each frame sampled every 15 - 20 msec). The examples described below focus on object manipulation and arrangement, so only the *effects* (the position and the orientation of the objects) are captured, omitting the demonstrator's actions (arm movements) and states (body posture). In ongoing work, three (or more) additional sensors can be used, one attached to the human torso and one at each hand, providing additional information about the demonstrator's actions and states.

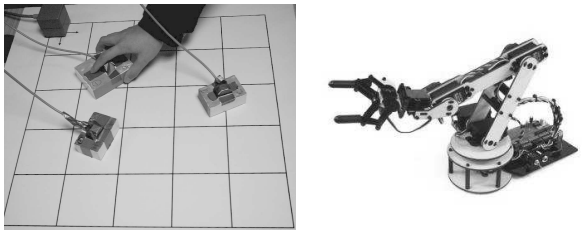


Fig. 5. **A picture of the experimental set-up for the demonstrator (left), and a picture of one of the target imitator platforms, the Lynx-6 robotic arm (right).** On the left, the human demonstrator manipulates (moves and rotates) three objects inside the workspace grid.

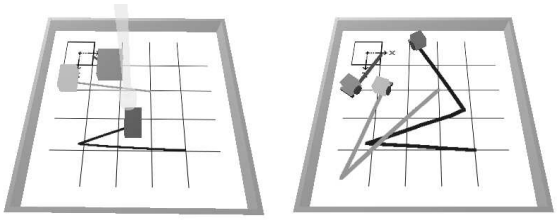


Fig. 6. **Two software imitator platforms.** On the left, three objects can be moved and rotated by a manipulator (visualized as a vertical cylinder mounted at the end of a bar positioned above the workspace). On the right, each object corresponds to a mobile robot. In both testbeds, the objects/robots leave a trail to help visualize their trajectories.

B. What to Imitate

The *what to imitate* module should extract the metrics and sub-goal granularity from the captured human demonstration, and pass them on to the *how to imitate* module. Currently, this module is not yet implemented in the JABBERWOCKY system.

In the examples presented here, the choice of metrics is given (see section II for a selection of effect metrics) instead of automatically extracted.

Determining how precisely the demonstrated behaviour is to be imitated, the sub-goal granularity in the system is given by finding the *critical points* in the trajectories of the manipulated objects (when considering the *effects* aspect of the demonstrated task). A critical point for an object occurs when the direction of its trajectory and/or the orientation changes beyond a certain threshold and consists of its co-ordinates and orientation. For each of the objects, only the sequential order of the critical points is considered without timing information. Therefore, in order to synchronize the events, when a critical point is found for an object, critical points must be created for the rest of the objects as well.

C. How to Imitate

The *how to imitate* module uses the metrics and sub-goal granularity provided by the *what to imitate module*, plus the initial state, embodiment restrictions and constrains of the imitator (for the target imitator platforms see next subsection), to generate appropriate action commands that if executed by the imitator will result in corresponding actions, states and/or effects.

The current version of JABBERWOCKY uses a simulation of a 2D workspace that can handle a number of 'block' objects that are moved and rotated around, accounting for object collisions and workspace confines. The simulation replays the captured demonstration at the given sub-goal granularity (see previous subsection), displaying the trajectory and orientation of the objects from the initial configuration to the current frame (critical point). The appropriate corresponding displacement/rotation that satisfies the given effect metric(s) for the particular frame is calculated (using the equations defined in the Appendix), taking into consideration the configuration (position and orientation) of the objects in the demonstrator's and imitator's workspaces. For each object (and at each frame), a different metric can be used (or even a combination of metrics).

Concentrating on effects alone and ignoring the correspondence of actions and states, can allow the *how to imitate* module to *partially* ignore the constraints of the particular embodiment of the imitator and provide a plan of how the corresponding objects should be moved/rotated, assuming the imitator already has the capability to move and rotate the objects. Of course, if required, constraints like the size and the effective reach of the imitator can be considered by modifying the geometry of the workspace.

Some displacements or rotations, although satisfying the given metric, might be invalid if the path is obstructed or final position is occupied by other objects or confines of the workspace. The *how to imitate* module can then either ignore the particular displacement/rotation (assuming that the problem will have been resolved by the next time the object must be manipulated) or discover an alternative path (if possible) to achieve the same effect according to the metric. The choice greatly depends on the context (e.g. is the obstacle likely to move, is there enough free space to go round), possibly requiring interaction from the user to resolve any ambiguities. The current version of JABBERWOCKY attempts to match the effects by proposing a displacement as close as possible to the desired displacement (if the end-point is blocked by another object), in some cases navigating around the obstructing object (if the path is blocked), or no displacement at all in extreme cases (when the object cannot be moved without disturbing the surrounding objects).

D. Target Imitator Platforms

Depending on the targeted imitator platform, the *how to imitate* module produces appropriate action commands. For matching the effects these commands indicate where the imitators must move their effectors on the workspace and once there, whether to pick up an object or place down (possibly rotated) an already obtained object. The JABBERWOCKY system provides a plan according to which the imitator (which is assumed capable of manipulating the objects itself) can successfully achieve an imitative behaviour based on a demonstration from a human.

1) Simulated Software Platforms: Two imitator platforms were implemented using the Webots[2] robot simulation software. The imitator's workspace contains three objects, of

[2]`http://www.cyberbotics.com/products/webots`

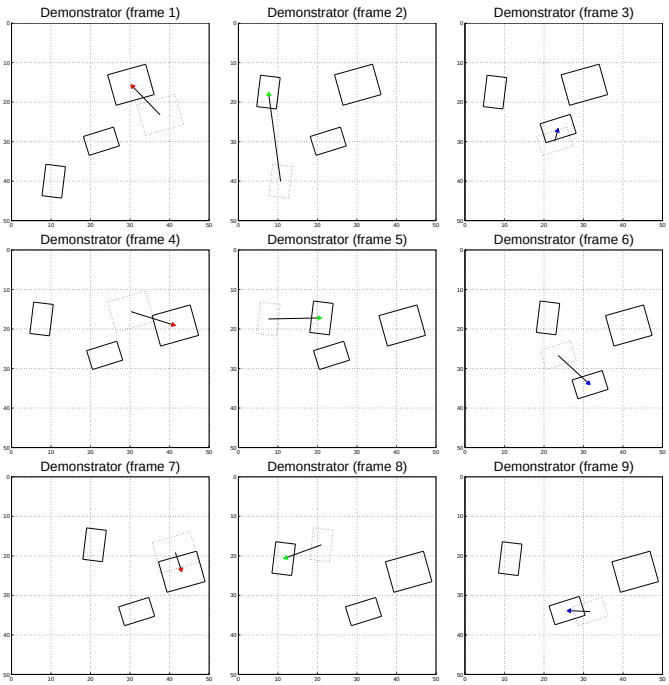


Fig. 7. **An example of a demonstration used by the JABBERWOCKY system.** The figure shows a sequential visualization of the effects of a demonstrated behaviour. In this case the behaviour is generated, not captured from a human user. At each frame, the displacement of an object is shown (vector) from its previous (dotted outline) to its current position (solid outline).

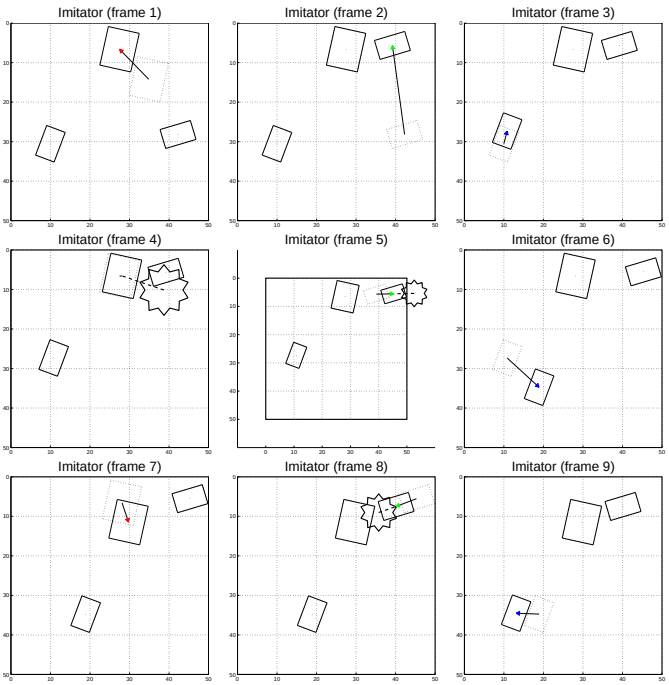


Fig. 8. **Based on the demonstration shown in Figure 7 and with the objects starting from a dissimilar initial configuration (positions and orientations), the JABBERWOCKY system can generate a sequence of action commands for the appropriate imitation behaviour.** The figure shows a sequential visualization of the corresponding effects (the *relative displacement* effect metric was used for all three objects throughout). The stars indicate the *exceptions*, cases when the displacement that minimizes the metric was not directly achievable (because either an object has to move outside workspace or is obstructed by another) and had to be modified accordingly (shown as a dotted line) for the action commands to result in valid effects.

the same size and color as the corresponding objects in the demonstrator's workspace. The imitator is embodied as a single arm manipulator, positioned above the workspace and able to pick-up, move and rotate the three objects (see Fig. 6, left). This embodiment, although dissimilar to the one of the human demonstrator, is nevertheless able to match both *displacement* and *angular* effect aspects of the demonstration.

In another target imitator platform (also implemented in software), the imitator's workspace contains no objects. Instead, the imitator is 'embodied' as three mobile robots, each corresponding to one of the objects manipulated by the demonstrator (see Fig. 6, right). Each robot is of the same size (so in this case, besides dissimilar demonstrator-imitator embodiments, there is also dissimilar object correspondence, mapping the objects to mobile robots). The robots can follow the individual trajectories of the objects as arranged by the demonstrator, but cannot match the orientation (while moving) because they are differential wheel robots. Therefore the *angular* effect aspect will be ignored when they imitate, matching only the *displacement* effect aspect.

Experiments using the above described simulated platforms, showing the multitargetability of the JABBERWOCKY system and its ability to generalize across different initial configurations of manipulated objects, were presented in [10], [11].

2) Robotic Hardware Platforms: The JABBERWOCKY system can also generate action commands targeted for the Lynx-6, a 6 DOF robotic arm (see Fig. 5, right).

IV. System Performance Evaluation

If the objects start from the same positions in the imitator's workspace as in the demonstrator's workspace, all the displacement effect metrics are equivalent (i.e. using them, the same trajectories, identical to the ones of the demonstrator, will be generated), similarly if the objects start in the same orientations all the angular effect metrics are equivalent (i.e. the objects rotate in the same way as in the demonstration). But if the objects start in an initial configuration (positions and orientations) dissimilar to that of the demonstration, the choice of metrics affects qualitatively the character of the resulting imitation behaviour (see Figs. 7 and 8 for an example).

Depending on the context (initial object configuration, imitator embodiment restrictions and environment constrains), sometimes the displacement and/or rotation of an object (in order to minimize the metric used) is not directly achievable. In those cases, the system generates alternative action commands that result instead in a modified displacement and/or orientation that can be carried out[3]. We call these events *exceptions*, and we can distinguish between two 'types':

- *type A exceptions* occur when an object, in order for the effect metric to be minimized, has to be moved

[3]We note that other approaches are possible and may be appropriate to resolving exceptions in different circumstances: employing alternative actions (as here), waiting for any moving obstructions to move on thus resolve the exception, seeking guidance from the human, or other methods.

and/or rotated in such a way that it drifts outside the imitator's workspace. The generated displacement and/or rotation action commands will keep the object inside the workspace (see Fig. 8, frame 5 for an example).

- *type B exceptions* occur when an object, in order for the effect metric to be minimized, has to be moved and/or rotated in such a way that its path or final position is obstructed by the other objects in the workspace. The generated displacement and/or rotation action commands will either guide the object around the obstacle, or allow it to approach as close as possible to the desired position (see Fig. 8, frames 4 and 8 for examples).

Exceptions occur when the metric used cannot be minimized and the system has to explore alternative solutions when generating action commands for an imitative attempt (given the particular context).

Experimental runs using the JABBERWOCKY system presented in [12] illustrate how the *number of exceptions* can be used as a performance measure by a robotic or software imitator of an object manipulation behavior. A low number of exceptions would indicate that the system was able to produce an imitative behaviour mostly similar (according to the metrics used) for a given context, while a high number would indicate the contrary.

The results from a small pilot user study (see [12] for details) indicate that there is a correlation between this quantitative performance measure (from the systems perspective) and the way subjects qualitatively evaluate (from a human perspective) the imitation attempts. The choice of effect metrics used to generate the imitative behaviours was shown to affect the subjects assessment of the similarity to the demonstration.

V. Discussion and Conclusion

Depending on the initial object configuration used, the JABBERWOCKY system is able to generalize based on the demonstrated task and produce appropriate corresponding action commands that when executed by a targeted imitator platform result in imitative behaviour, successful according to the sub-goal granularity and the metrics used to match the different aspects of the demonstration. Aspects captured by *effect metrics* described in section II can all successfully be matched. The action commands can be targeted for multiple imitator platforms, both in hardware and simulated in software. The multi-platform targetability of the JABBERWOCKY system to map human demonstrated manipulations to matching robotics manipulations (in simulation) is shown in [10] and its ability to generalize across different initial configurations of manipulated objects is shown in [11]. The number of *exceptions* (events occuring when, given the particular context, the metric used cannot be minimized and the system has to explore alternative solutions when generating action commands for an imitative attempt) can be used for measuring the performance of JABBERWOCKY and related systems [12].

Robots programmed to learn human demonstrated tasks and skills will need mechanisms to match according to different aspects demonstrated. Compared to a restrictive preprogrammed strategy, such a robot will be able to learn how to perform tasks in a more flexible way, adapting its execution according to the observed demonstrations by its human users. A wide selection of metrics and sub-goal granularity can be supported by JABBERWOCKY and related systems which tell the robots how to imitate, generalizing across different initial configurations. From the examples shown here and in [10], [11], it becomes apparent that the relative/absolute position and rotation of objects are important aspects of a demonstrated task to match (or not) according to effect metrics, depending on the state of the objects in the environment and the context. The exploratory characterization of the space of effect metrics reveals that matching of results is a more sophisticated issue that generally acknowledged. This wide range of possible effect metrics illustrates that even the effect aspect of the correspondence problem for human-robot interaction by itself is already quite complex. Goal extraction in terms of effect metrics and granularity may have many different solutions that might not all be appropriate according to the desired results or context. Depending on the constraints of the imitator embodiment, a 'many-to-one' or 'one-to-many' correspondence between imitator and model sub-goals may be required for specific parts of the task. It is also possible that an imitating agent has to switch metrics and granularity during the imitation attempt. This has not been emphasized at all in the literature so far (but see [8]). This creates particular problems and challenges for sub-goal and metric extraction systems that can be used in programming robots by demonstration. The use of repeated demonstrations [13], saliency detection [14] and goal-marking via deixis and non-verbal signaling by humans [15]–[17] may help contribute solutions to these problems. Other research questions yet to be addressed include the importance of order effects in manipulation and establishing object-object correspondence.

Appendix
Definitions of the Effect Metrics

This appendix provides detailed definitions for the effect metrics discussed in this article. When they appear in the parameters below, subscripts M and I indicate the model and the imitator, respectively.

A. Displacement Effect Metrics

The model is moving an object from position X_M to position X'_M on the workspace, achieving an object displacement $\Delta X_M = X'_M - X_M$, where $X_M = \begin{bmatrix} x_M \\ y_M \end{bmatrix}$, $X'_M = \begin{bmatrix} x'_M \\ y'_M \end{bmatrix}$, and $\Delta X_M = X'_M - X_M = \begin{bmatrix} x'_M - x_M \\ y'_M - y_M \end{bmatrix}$. The imitator should move the same (or corresponding) object from position X_I to position X'_I on the workspace, with a displacement $\Delta X_I = X'_I - X_I$, such that a displacement metric is minimised.

The **Relative Displacement Effect Metric** is minimized if $\Delta X_I = \Delta X_M$ and $X'_I = X_I + \Delta X_M = \begin{bmatrix} x_I \\ y_I \end{bmatrix} + \begin{bmatrix} x'_M - x_M \\ y'_M - y_M \end{bmatrix} = \begin{bmatrix} x_I + x'_M - x_M \\ y_I + y'_M - y_M \end{bmatrix}$.

The **Absolute Position Effect Metric** is minimized if $X'_I = X'_M$ and $\Delta X_I = X'_M - X_I = \begin{bmatrix} x'_M - x_I \\ y'_M - y_I \end{bmatrix}$.

The **Relative Position Effect Metric** is minimized if the object is moved to a similar position relative to other objects in the workspace. Here it is defined for three objects in the workspace. Let the position of the manipulated object be $A = \begin{bmatrix} x_A \\ y_A \end{bmatrix}$, and the positions of the other two objects $B = \begin{bmatrix} x_B \\ y_B \end{bmatrix}$ and $C = \begin{bmatrix} x_C \\ y_C \end{bmatrix}$. The imitator should move the same (or corresponding) object to form a triangle so that it is the "same" as the triangle formed by the model, i.e. the angles $C\hat{A}B$, $A\hat{B}C$ and $B\hat{C}A$ are equal. The triangle sides $\overline{AB}$, $\overline{BC}$ and $\overline{CA}$ can be equal only if the objects were in the same initial configuration for both the model and the imitator, so in general only the equality of the angles can be used. Given $C\hat{A}B_M$, $A\hat{B}C_M$, $B\hat{C}A_M$ and $\overline{BC}_I$, we can find the other two sides $\overline{AC}_I = \sqrt{\frac{(1-cos^2(A\hat{B}C_M))\times\overline{BC}_I^2}{(1-cos^2(C\hat{A}B_M))}}$ and $\overline{AB}_I = \sqrt{\frac{(1-cos^2(B\hat{C}A_M))\times\overline{BC}_I^2}{(1-cos^2(C\hat{A}B_M))}}$, to satisfy the equalities $C\hat{A}B_I = C\hat{A}B_M$, $A\hat{B}C_I = A\hat{B}C_M$ and $B\hat{C}A_I = B\hat{C}A_M$. Assuming that side $\overline{BC}_I$ lies on the $(0, +\infty)$ x-axis with points $\mathcal{B} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\mathcal{C} = \begin{bmatrix} |\overline{BC}_I| \\ 0 \end{bmatrix}$ corresponding to B_I and C_I, we can then find a point $\mathcal{A} = \begin{bmatrix} \frac{a_I^2-b_I^2+c_I^2}{2\times a_I} \\ \frac{\sqrt{(-a_I+b_I-c_I)\times(-a_I-b_I+c_I)\times(-a_I+b_I+c_I)\times(a_I+b_I+c_I)}}{2\times a_I} \end{bmatrix}$† corresponding to A_I, such that the equalities $\overline{\mathcal{AB}} = \overline{AB}_I$, $\overline{\mathcal{BC}} = \overline{BC}_I$ and $\overline{\mathcal{CA}} = \overline{CA}_I$ are satisfied. To find A_I we need to rotate and translate $\mathcal{A}$ in respect to the actual co-ordinates of $B_I = \begin{bmatrix} x_B \\ y_B \end{bmatrix}$ and $C_I = \begin{bmatrix} x_C \\ y_C \end{bmatrix}$ in the imitator's workspace: $A = \begin{bmatrix} x_A \\ y_A \end{bmatrix} = \begin{bmatrix} cos\phi & sin\phi \\ -sin\phi & cos\phi \end{bmatrix} \times \mathcal{A} + \begin{bmatrix} x_B \\ y_B \end{bmatrix}$, where $\phi = tan^{-1}\left(\frac{y_C-y_B}{x_C-x_B}\right)$. The relative position effect metric is minimized if $X'_I = A$ and $\Delta X_I = A - X_I = \begin{bmatrix} x_A - x_I \\ y_A - y_I \end{bmatrix}$.

B. Angular Effect Metrics

The model is rotating an object from orientation θ_M to orientation θ'_M on the workspace, with a rotation $\Delta\theta_M = \theta'_M - \theta_M$. The imitator should rotate the same (or corresponding) object from orientation θ_I to orientation θ'_I on the workspace, with a rotation $\Delta\theta_I = \theta'_I - \theta_I$, such that an angular metric is minimized (see Fig. 2).

The **Rotation Effect Metric** is minimized if $\Delta\theta_I = \Delta\theta_M$ and $\theta'_I = \theta_I + \Delta\theta_M$.

The **Orientation Effect Metric** is minimized if $\theta'_I = \theta'_M$ and $\Delta\theta_I = \theta'_M - \theta_I$.

C. Other Effect Metrics

Depending on the initial configuration of the corresponding objects in the imitator's workspace, or the particular task that the imitator would like to achieve, it might be desirable to use also other effect metrics that take into account mirror symmetry, both positional and angular, to features of the environment or other agents. For example:

†Where $a_I = \overline{BC}_I$, $b_I = \overline{AC}_I$ and $c_I = \overline{AB}_I$.

The **Mirror Displacement Effect Metric** is minimized if $\Delta X_I = -\Delta X_M$ and $X'_I = X_I - \Delta X_M = \begin{bmatrix} x_I \\ y_I \end{bmatrix} - \begin{bmatrix} x'_M - x_M \\ y'_M - y_M \end{bmatrix} = \begin{bmatrix} x_I - x'_M + x_M \\ y_I - y'_M + y_M \end{bmatrix}$.

The **Mirror Rotation Effect Metric** is minimized if $\Delta\theta_I = -\Delta\theta_M$ and $\theta'_I = \theta_I - \Delta\theta_M$.

The **Parallel Orientation Effect Metric** is minimized if $\theta'_I = \vartheta$ and $\Delta\theta_I = \vartheta - \theta_I$, where ϑ is the orientation of a feature in the environment (e.g. one edge of the workspace). If these features in the workspace of the imitator are the same as the model's, then $\vartheta \equiv \theta'_M$ and this metric becomes equivalent to the *orientation effect metric*.

D. Combinations of Effect Metrics

To evaluate both the movement and the orientation of an object, both displacement and angular metric types must be used. To match the observed effect, the (corresponding) object needs to be moved on the workspace according to the displacement given by the displacement effect metric and rotated according to the angular effect metric used.

A weighted combination of more than one of the displacement metrics can be also used, by a (convex) weighted sum of the displacement vectors that minimise each metric. For example, if $\Delta X_i = \begin{bmatrix} \Delta x_i \\ \Delta y_i \end{bmatrix}$ is the displacement that minimises a displacement effect metric i, and ω_1, ... , ω_n are the weights of the n displacement effect metrics to be combined, the displacement that minimizes this composite metric is then given by $\Delta X = \begin{bmatrix} |\Delta X| \times cos(\phi) \\ |\Delta X| \times sin(\phi) \end{bmatrix}$, where $|\Delta X| = \omega_1 \times \sqrt{{\Delta x_1}^2 + {\Delta y_1}^2} + ... + \omega_n \times \sqrt{{\Delta x_n}^2 + {\Delta y_n}^2}$ and $\phi = \omega_1 \times tan^{-1}\left(\frac{\Delta y_1}{\Delta x_1}\right) + ... + \omega_n \times tan^{-1}\left(\frac{\Delta y_n}{\Delta x_n}\right)$.

ACKNOWLEDGMENT

The work described here was conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

REFERENCES

[1] J. Demiris and G. Hayes, "Imitation as a dual-route process featuring predictive and learning components: A biologically-plausible computational model," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 327–361.

[2] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Science*, vol. 6, pp. 481–487, 2002.

[3] A. Billard, "Learning motor skills by imitation: a biologically inspired robotic model," *Cybernetics and Systems*, vol. 32, no. 1-2, pp. 155–193, 2001.

[4] M. N. Nicolescu and M. M. Matarić, "Learning and interacting in human-robot domains," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 31, no. 5, pp. 419–430, 2001.

[5] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[6] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observations of human performance," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 799–822, November 1994.

[7] C. L. Nehaniv and K. Dautenhahn, "Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation," in *Proceedings European Workshop on Learning Robots 1998 (EWLR-7), Edinburgh, 20 July 1998*, J. Demiris and A. Birk, Eds., 1998, pp. 64–72.

[8] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, "Towards robot cultures? – Learning to imitate in a robotic arm test-bed with dissimilar embodied agents," *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, vol. 5, no. 1, pp. 3–44, 2004.
[9] ——, "Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments," *IEEE Trans. Systems, Man & Cybernetics: Part A*, vol. 32, no. 4, pp. 482–496, 2002.
[10] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders, "Achieving corresponding effects on multiple robotic platforms: Imitating using different effect metrics," in *Proc. Third International Symposium on Imitation in Animals and Artifacts – Hatfield, UK, 12-14 April 2005*. Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2005, pp. 10–19.
[11] ——, "An approach for programming robots by demonstration to manipulate objects: Considerations on metrics to achieve corresponding effects," in *Proc. 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '05)*, 2005.
[12] ——, "Evaluation of robot imitation attempts: Comparison of the system's and the human's perspectives," submitted.
[13] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47:2-3, 2004.
[14] B. Scassellati, "Imitation and mechanisms of joint attention: A developmental structure for building social skills," in *Computation for Metaphors, Analogy and Agents*, C. L. Nehaniv, Ed. Springer Lecture Notes in Artificial Intelligence, Volume 1562, 1999, pp. 176–195.
[15] G. Butterworth, "Pointing is the royal road to language for babies," in *Pointing: Where Language, Culture, and Cognition Meet*, S. Kita, Ed. Lawrence Erlbaum Assoc Inc, 2003, pp. 9–26.
[16] J. Call and M. Carpenter, "Three sources of information in social learning," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002.
[17] H. Bekkering and W. Prinz, "Goal representations in imitative actions," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 555–572.

Aris Alissandrakis received his Ph.D. degree in Computer Science at the Adaptive Systems Research Group, Computer Science Department, University of Hertfordshire, England in 2003 and the M.Eng. degree in Cybernetics from the Department of Cybernetics, University of Reading, England in 1999. He is currently a postdoctoral research fellow in the European Integrated Project Cogniron ("The Cognitive Robot Companion"), studying imitation in the context of robot-human interaction.

Chrystopher L. Nehaniv received his Ph.D. degree from the University of California at Berkeley in 1992, and is currently Professor of Mathematical and Evolutionary Computer Sciences at the University of Hertfordshire, in Hatfield, England. He is a founding member of the IEEE Working Group on Artificial Life and Complex Adaptive Systems, and serves as Associate Editor for the journals *BioSystems: Journal of Biological and Information Processing Sciences* and *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, as well as Director of the U.K. Engineering and Physical Science Research Council and Interactive Network on Evolvability in Biological and Software Systems.

Kerstin Dautenhahn received her Ph.D. degree from the Biological Cybernetics Department of the University of Bielefeld, Bielefeld, Germany, in 1993. She is Professor of Artificial Intelligence in the School of Computer Science and coordinator of the Adaptive Systems Research Group at the University of Hertfordshire in England. She has published more than 100 research articles on social robotics, robot learning, human-robot interaction and assistive technology. Prof. Dautenhahn edited several books and frequently organises international research workshops and conferences. She is involved in several European robotics projects and is Editor in Chief of the journal *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*.

Joe Saunders received the M.Sc. degree with distinction in Artificial Intelligence from the Department of Computer Science, University of Hertfordshire, England, in 2003. He is currently pursuing a Ph.D. degree at the Adaptive Systems Research Group, School of Computer Science, University of Hertfordshire, England. He previously directed large software development projects for a number of international investment banks. His research interests include social robotics and imitation in artificial systems.

Correspondence Mapping Induced State and Action Metrics for Robotic Imitation

Aris Alissandrakis, Chrystopher L. Nehaniv, and Kerstin Dautenhahn

***Abstract*— This article addresses the problem of body mapping in robotic imitation where the demonstrator and imitator may not share the same embodiment (degrees of freedom (DOFs), body morphology, constraints, affordances and so on). Body mappings are formalized using a unified (linear) approach via correspondence matrices, which allow one to capture partial, mirror symmetric, one-to-one, one-to-many, many-to-one and many-to-many associations between various DOFs across dissimilar embodiments. We show how metrics for matching state and action aspects of behaviour can be mathematically determined by such correspondence mappings, which may serve to guide a robotic imitator. The approach is illustrated and validated in a number of simulated 3D and robotic examples, using agents described by simple kinematic models and different types of correspondence mappings.**

***Index Terms*— imitation and social learning, state and action metrics, correspondence problem, programming by demonstration.**

I. Introduction

IMITIATION is a powerful learning tool when a number of agents interact in a social context. The demonstrator and imitator agents may or may not belong to the same species (a parent teaching a child, a human training an animal) or even be biological and artificial entities (e.g. in human-robot interaction). The latter is a very interesting paradigm explored in computer science and robotics, with researchers influenced by work on biology, ethology and psychology in order to design controllers that would allow their robots to be programmed and learn more easily and efficiently [12], [9], [8], [17], [20], [14].

A fundamental problem when learning *how* to imitate is to create an appropriate (partial) mapping between the actions afforded by particular embodiments to achieve corresponding states and effects by the model and imitator agents (solving a *correspondence problem*) [16]. The solutions to the correspondence problem will depend on the subgoal granularity and the metrics used to evaluate the similarity between actions, states and/or effects, resulting in qualitative different imitative behaviours [2], [1]. The related problem of *what* to imitate addresses the choice of metrics and sub-goal granularity that should be used for imitating, depending on the context. See [7], [19] for robotic examples and [10], [11], [6] for ethological and psychological aspects.

Related to work on solving the correspondence problem for imitation learning in robotics is the ALICE generic imitation framework [1], [2] and the JABBERWOCKY system [4], [5], [3]. Both are generic approaches addressing multiple demonstrator and target imitator embodiments using different metrics to achieve different types of social learning, matching different behavioural aspects.

The authors are with the School of Computer Science, Adaptive Systems Research Group, University of Hertfordshire, College Lane, Hatfield, Hertfordshire AL10 9AB, U.K. (e-mail: a.alissandrakis@herts.ac.uk).

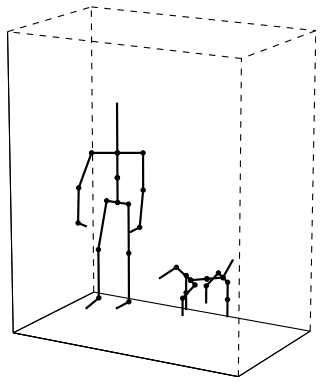


Fig. 1. **Some embodiment examples using simple kinematic models.** A human (left), and an AIBO robot (right).

In the current work, we introduce a novel generic approach to the correspondence problem, via body-mapping for the cases of state and/or action matching. In particular, partial, relative and mirror matching all arise as special cases of such correspondence mappings. Moreover, an infinite set of metrics (parameterized by correspondence matrices) for imitation performance are induced via such body correspondences.[1] This contributes towards a characterization of types of matching in social learning. Previously we studied the space of effect metrics [5], while in this article the focus turns to *state* and *action* metrics. The approach is illustrated and validated via a number of simulated 3D and robot examples mapping across dissimilar embodiments and is applicable to robot programming by human demonstration.

II. Different Bodies

Different agent bodies can be described as simplified kinematic models, comprising of a rooted acyclic connected graph of *segments* (see Fig. 1). Each segment has a base and a tip end, and is described by of the quintuple (i,ℓ_i,p_i,θ_i,ϕ_i), where

- i is the index number of the segment,
- ℓ_i is segment length,
- p_i is the index number of the $parent$ segment that the base of this segment is attached to the tip of,

[1] That is, to say that a correspondence mapping "induces" a metric means exactly that it mathematically determines the metric.

- and θ_i and ϕ_i are the *azimuth* and *polar* angles for the spherical coordinates (l_i,θ_i,ϕ_i) that indicate how the segment is positioned in 3D space (relative to the end of its parent segment). NB: In general the range of the angles θ_i and ϕ_i may be constrained within given respective ranges.

The values of θ_i and ϕ_i are *relative* for each segment, but *absolute* angles for segment i, Θ_i and Φ_i, can be obtained inductively starting from the next segment after the root segment[2]:

$$\begin{aligned} \Theta_i &= \theta_i + \Theta_{p_i} \\ \Phi_i &= \phi_i + \Phi_{p_i} \end{aligned}$$

See Appendix I for detailed kinematic equations.

The **state** of such a kinematic model can be defined as the vector containing the values of the degrees of freedom (DOF), i.e. the values of the azimuth and polar angles for each segment. The number of DOFs for each agent will in general be twice the number of segments, depending on embodiment restrictions. Depending on whether the relative or the absolute angle values are used, for an embodiment with n segments, two different state vectors can be considered:

$$\begin{aligned} S_{relative} &= [\theta_1\ \phi_1\ \theta_2\ \phi_2\ \ldots\ \theta_n\ \phi_n] \\ S_{absolute} &= [\Theta_1\ \Phi_1\ \Theta_2\ \Phi_2\ \ldots\ \Theta_n\ \Phi_n] \end{aligned}$$

For the rest of the article, the notation S_j will be used to refer to the state value of the j^{th} DOF of an agent.

An **action** can be defined as the difference between two consecutive state vectors S and S':

$$A = S' - S$$

Using either the relative or absolute representation of state vectors for calculating an action vector produces mathematically equivalent results. Note that depending on the embodiment, a change in the relative values of the j^{th} DOF can influence the absolute values of subsequent DOFs (see Appendix I for the kinematic equations). For this article, the actions will be defined using the relative state vectors.

Effects can be defined as changes to the body-world relationship (e.g. location) of the agent and/or to positions, orientations and states of external objects. In the current article we will only consider state and action behaviour aspects and metrics. For characterization of the *space of effect metrics* and some mathematical definitions of effect metrics, see [5].

III. State and Action Metrics

To evaluate the similarity of behaviour, with respect to states and actions, between an agent β imitating another agent α, we define and use appropriate *metrics*. For the moment let us assume that both agent embodiments have the same number of DOFs, n.

[2]For mathematical convenience, the root node is treated as a segment of length $\ell_0 = 0$, but θ_0 and ϕ_0 can have non-zero values, to orient the entire model. For expository purposes, without loss of generality, in this article we ignore the latter possibilities ($\Theta_0 = \theta_0 = 0$, $\Phi_0 = \phi_0 = 0$).

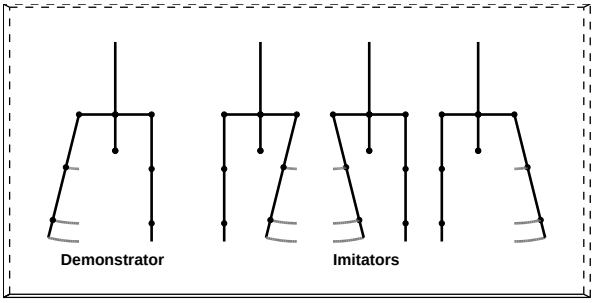


Fig. 2. **Examples of symmetry via correspondence mapping.** The figure shows a demonstrator (left) and three imitators, facing the reader, each with an upper human torso embodiment. The demonstrator is moving its right arm to its left. Each of the three imitators are using different correspondence mappings: mapping the demonstrator's right arm to the left arm of the imitator (second from the left), using a weight of minus one, but maintaining the same arm mapping (second from the right) and finally both mapping the demonstrator's right arm to the left arm of the imitator and using a weight of minus one (right). The grey lines trace the movement of the arms.

A first global **state metric** can be defined as

$$\mu_{state} = \sum_{j=1}^{n} |S_j^\alpha - S_j^\beta|, \tag{1}$$

where S_j^α and S_j^β are the values of the state vectors for the two agents. Depending on whether the relative or absolute state vectors are used, the state metric is called *relative* or *absolute*, respectively.

A first global **action metric** can be defined as

$$\mu_{action} = \sum_{j=1}^{n} |A_j^\alpha - A_j^\beta|, \tag{2}$$

where A_j^α and A_j^β are the values of the action vectors for the two agents.

An agent performing actions so as to minimize one (or a weighted combination) of these two metrics would successfully imitate a demonstrator in respect to states and/or actions. For effect metrics, see [5]. In the following sections, some more complex metrics are defined.

IV. Correspondence Mapping

For two agents, demonstrator α and imitator β with n and m DOFs respectively, a $n \times m$ **correspondence matrix** can be defined as

$$\mathcal{C} = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,m} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \ldots & w_{n,m} \end{bmatrix},$$

where the $w_{i,j}$ values are real-valued weights, determining how the j^{th} DOF of the imitator β depends on the i^{th} DOF of the demonstrator α. The j^{th} column of the matrix can be thought as a vector indicating how the DOFs of the demonstrator influence the j^{th} DOF of the imitator. Depending on how many of the weights have a non-zero value, this correspondence mapping can be *one-to-one*, *one-to-many* (or *many-to-one*) or *many-to-many*. If *partial* body imitation is

desired, some DOF of the imitator (and/or the demonstrator) can be omitted by setting an entire column (resp. row) to zero in the correspondence matrix.

Assuming both agents share the same embodiment (and as a result have the same number of DOFs), a simple example of a one-to-one correspondence mapping would be using the identity matrix as a correspondence matrix. Alternatively, if some *mirror symmetry* is wanted, then the DOFs for the right arm and leg of the demonstrator (see example in Fig. 2, left) could be mapped to the DOFs for the left arm and leg of the imitator, and vice versa (Fig. 2, second from the left). Another possible form of symmetry results from mapping some of the demonstrator's DOF using a weight of minus one (e.g. if the demonstrator raises its hand, the imitator should lower its hand or if the demonstrator turns its head to the left the imitator should turn to the right, see example in Fig. 2, second from right).

If the agents do not have the same number of DOFs (or depending on their particular morphology), it may be useful to map a single DOF to many DOFs. For example, consider correspondences between a human body as model to a dolphin-like imitator[3]: A dolphin using its mouth corresponding to either human arm (grasping an object), or its tail to toss a ball back to a human that used both arms comprise real-world examples of many-to-one mappings. These two examples also illustrate that the correspondence need not to be static – the human hand(s) are mapped to different dolphin body parts in each case – but can be adapted depending on the context and the tasks involved.

V. Induced State and Action Metrics

The metric definitions in section III are appropriate for the most simple one-to-one mapping (the identity mapping), with both agents sharing the same number of DOFs (and probably a very similar morphology). But in general, using a correspondence matrix, other metric definitions can be induced.

First, the state vector S^α and the action vector A^α of the demonstrator can be multiplied with the correspondence matrix

$$\mathcal{S} = S^\alpha \times \mathcal{C} \tag{3}$$

$$\mathcal{A} = A^\alpha \times \mathcal{C} \tag{4}$$

producing two new vectors in imitator coordinates.

Combining Eqs. 1, 3 for the state metric gives

$$\mu^{\mathcal{C}}_{state} = \sum_{j=1}^{m} |\mathcal{S}_j - S^\beta_j| \epsilon_j,$$

where the corrective term

$$\epsilon_j = \begin{cases} 0, \text{ if } \sum_{i=1}^{n} w^2_{i,j} = 0 \\ 1, \text{ otherwise} \end{cases}, \tag{5}$$

takes the value zero if the j^{th} column of the correspondence matrix contains only zeros (effectively omitting the imitator's j^{th} DOF). Intuitively, the components of $\mathcal{S}$ and $\mathcal{A}$ (for such $\epsilon_j \neq 0$) can be thought as current subgoal state and action target values. As in the previous definition, this state metric is called *relative* or *absolute* depending on whether the relative or absolute state vectors are used, respectively.

Finally, combining Eqs. 2, 4 and 5 for the action metric gives:

$$\mu^{\mathcal{C}}_{action} = \sum_{j=1}^{m} |\mathcal{A}_j - A^\beta_j| \epsilon_j$$

These $\mu^{\mathcal{C}}_{state}$ and $\mu^{\mathcal{C}}_{action}$ metrics are called the *induced state* and *action metrics for the linear correspondence* $\mathcal{C}$.

Depending on the correspondence mapping used, a plethora of new complex metrics (also allowing for dissimilar embodiments) can be induced considering state or action aspects. The next section will illustrate a variety of examples.

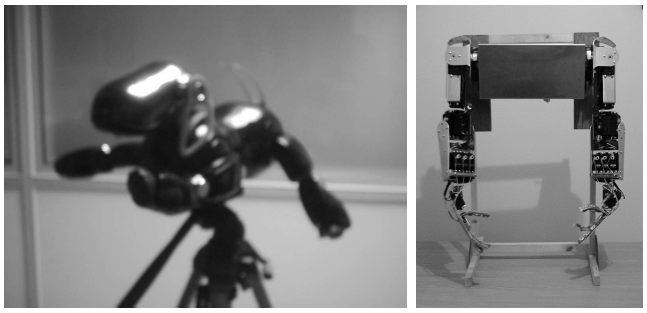


Fig. 3. **Example robotic platforms.** On the left, a dog-like Sony AIBO robot is positioned suspended on a tripod. On the right, a pair of human-size robotic arms is shown attached to a wooden rack (the robot arms were designed and made by Michael Walters of the University of Hertfordshire).

VI. Mapping Across Dissimilar Bodies

Using a system implemented in MATLAB, we are able to describe a variety of agent embodiments as simple kinematic models (defined in section II with kinematic equations as in Appendix I). These embodiments include models of robotic platforms as seen in Fig. 3.

For a given demonstrator and imitator embodiment pair, the imitator attempts to match the behaviour of the demonstrator by minimizing a given metric (or a combination of metrics). This can be done continuously (*immediate imitation*) or after the completion of the demonstration (*deferred imitation*) [15]. Moreover, the *granularity* or "fineness" of the matching of actions, states and/or effects determines a sequence of subgoals for the imitator to achieve, and the appropriate level of granularity may be different depending on context and task. Different correspondence mappings can be defined between the two agents, yielding qualitatively different types of matching behaviours.

A. Examples

Using the system, we conducted a series of simulation runs, using a variety of agent embodiments and correspondence mappings. The demonstrator performs a series of actions and the imitator tries to minimize the correspondence induced

[3]Different mappings do appear to be employed by real-life dolphins in imitating humans [13].

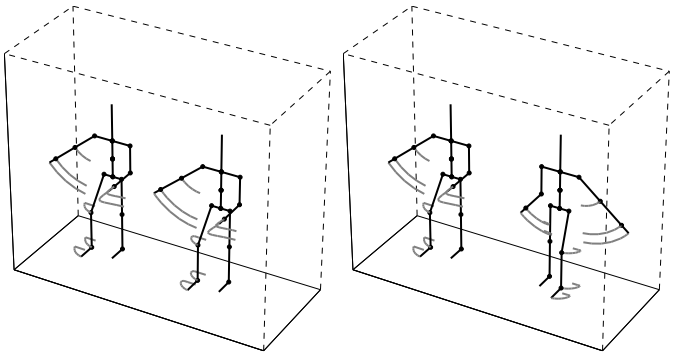


Fig. 4. **Two examples of imitation across similar embodiments (humanoid).** Both demonstrator (left in both examples) and imitator (right in both examples) share the same humanoid embodiment. In the example on the left, the identity mapping is used as the correspondence mapping. In the example on the right, the left arm and leg of the demonstrator are mapped on the right arm and leg of the imitator (and vice versa) with a weight of minus one, resulting in a mirror symmetry. The grey traces visualize the body part trajectories.

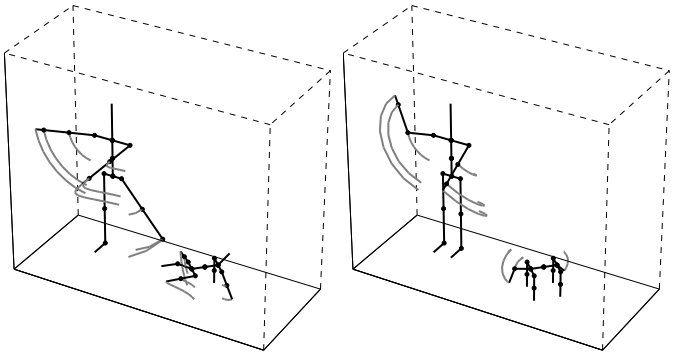


Fig. 5. **Two examples of imitation across dissimilar embodiments (humanoid and dog-like).** The demonstrator (left in both examples) is embodied as a humanoid, while the imitator (right in both examples) is embodied as a dog-like AIBO robot (see Fig. 3, left). In the example on the left, a simple one-to-one correspondence mapping is used, mapping the first two (out of three) segments of the demonstrator's arms and legs to the two segments of the four imitator's legs. In the second example on the right, the demonstrator's first segment of the left arm is mapped on the imitator's tail, and the demonstrator's first two right arm segments to the neck and head of the imitator's head. This results in the demonstrator controlling 'puppeteer-like' the head and tail of the robot. The grey traces visualize the body part trajectories.

relative state metric. Continuously using the components of $\mathcal{S}$, for which $\epsilon_j \neq 0$, as the current subgoals for each DOF j, the imitator performs actions which attempt to reduce the contribution of error in each such component. Here, the rate of change was restricted to half the component-wise error. Of course, many other selection mechanisms are possible for both immediate or deferred imitation.

1) Identity and Mirror Symmetry Mappings: Two examples of imitation across similar embodiments are shown in Fig. 4. Both demonstrator and imitator are humanoid. In the first example the identity correspondence mapping is used. In the second example, using the same demonstration, symmetry is achieved by mapping the left body parts of the demonstrator to the right body parts of the imitator and vice versa (see also examples in Fig. 2).

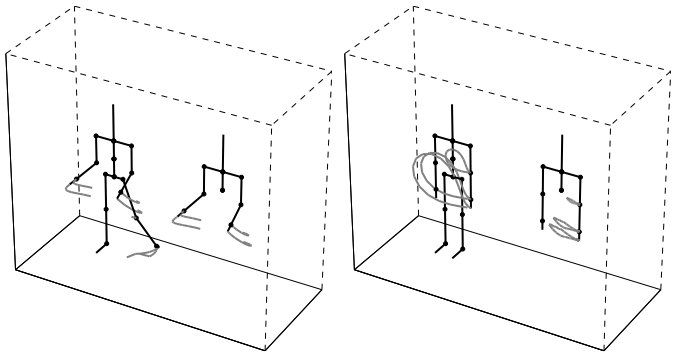


Fig. 6. **Two examples of imitation across dissimilar embodiments (whole and upper torso only humanoids).** The demonstrator (left in both examples) is embodied as a humanoid, while the imitator (right in both examples) is embodied as an upper torso humanoid robot (similar to the one shown in Fig. 3, right). In the first example on the left, the arms of the demonstrator are mapped using a weight of one to the arms of the imitator. Note that the movement of the demonstrator's left leg is ignored as these demonstrator's DOFs are omitted (via a zero row in the correspondence matrix). In the example on the right, the same mapping is used, but the rate of movement of the imitator is severely limited, resulting in *impersistence* (see further discussion in section VII). The grey traces visualize the body part trajectories.

2) Multiple Mappings between Dissimilar Bodies: The model of an AIBO robot is used as an imitator in the examples shown in Fig. 5. In the first example, the right arm of the demonstrator is mapped on the right front leg of the robot, the left arm to the left front leg, the right leg to the back right leg and the left leg to the back left leg. As each of the arms and legs of the demonstrator consists of three segments, and the imitator's legs consist of two segments, only the first two segments are mapped. In the second example, the imitator's head and tail are controlled 'puppeteer-like', by mapping the first two segments of the right arm and the first segment of the left arm of the demonstrator to them, respectively.

3) Partial Mappings: An example of partial mapping is shown in Fig. 6 (left). As the imitator is an upper torso humanoid, the DOFs in the lower body parts of the (whole humanoid) demonstrator are ignored (via zero rows in the matrix), with a unity one-to-one mapping used for the upper body.

4) One-to-Many Mappings: When the perception of the demonstrator by the imitator is limited, a complicated one-to-many correspondence mapping could be used. Assuming a human acting as a demonstrator, but providing the system and the imitator with only the coordinates of three motion sensors, one attached to her/his waist and one in each hand. Filtering perception through this sensory apparatus yields a reduced representation of the demonstrator embodiment that can be modeled as a 'V' letter shape kinematic model[4]. The θ and ϕ of each arm segment of the 'V' embodiment can be mapped on each of the segments of the corresponding arms of a humanoid imitator, with different weights. For example,

[4]Note that as the human moves her/his arms around, the lengths of the two segments of the 'V' will change accordingly and not remain constant. But this can be ignored since, for the correspondence mapping, the important parameters are the azimuth and polar angles. These can be found from the (relative to the waist sensor) Cartesian coordinates of each arm sensor.

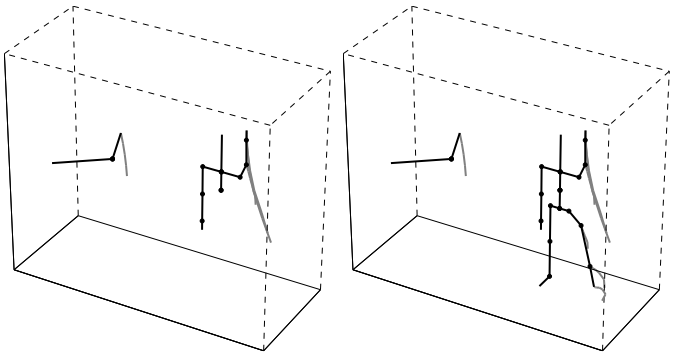


Fig. 7. **Two examples of imitation across dissimilar embodiments using one-to-many correspondence mapping.** The demonstrator (left in both examples) is embodied as an abstract 'letter V' shape, visualizing the three motion sensors attached to a human (one to waist and one on each hand), while the imitator (right in both examples) is embodied as a humanoid. In the example to the left, the left segment of the demonstrator is mapped with different weight values to the imitator's left arm segments. In the second example to the right, this mapping is extended by also mapping the left segment of the demonstrator with different weight values to the imitator's left leg segments. The grey traces visualize the body part trajectories.

in Fig. 7 (left), the following mapping

$$\begin{aligned} w_{\theta_{LA},\theta_{LS}} = w_{\phi_{LA},\phi_{LS}} &= 1 \\ w_{\theta_{LA},\theta_{LE}} = w_{\phi_{LA},\phi_{LE}} &= 0.5 \\ w_{\theta_{LA},\theta_{LW}} = w_{\phi_{LA},\phi_{LW}} &= 0.25 \end{aligned}$$

is used, where LA is the demonstrator's left 'V' arm segment and LS, LE and LW are the imitator's left shoulder, elbow and wrist segments, respectively. As a result, as the human demonstrator (not shown) lifts her/his left arm, the left segment of the 'V' model also rises and the left arm of the humanoid imitator rises as well. In Fig. 7 (right), an extension of this mapping is used with

$$\begin{aligned} w_{\theta_{LA},\theta_{LH}} &= 0.2 \\ w_{\phi_{LA},\phi_{LH}} &= 1 \\ w_{\theta_{LA},\theta_{LK}} &= -1 \\ w_{\theta_{LA},\theta_{LAn}} &= 0.05 \\ w_{\phi_{LA},\phi_{LAn}} &= 0.05 \end{aligned}$$

where LH, LK and LAn are the imitator's left hip, knee and ankle segments. As a result, both the left arm and leg of imitator rise when the demonstrator's left 'V' segment moves. These mappings are presented here only as indicative examples of complex one-to-many mappings.

B. Evaluation Using State and Action Metrics

The values of the induced *absolute state*, *relative state* and *action* metrics during the simulations shown in Figs. 4 to 7 are plotted in Figs. 8 to 15.

The rate of movement of the imitator during the simulations was limited to reflect real-world constraints. The current subgoal is constantly updated using the current state of the demonstrator during the run. When each demonstration finishes, the simulation does not terminate until the imitator manages to reach the final subgoal of the demonstrator.

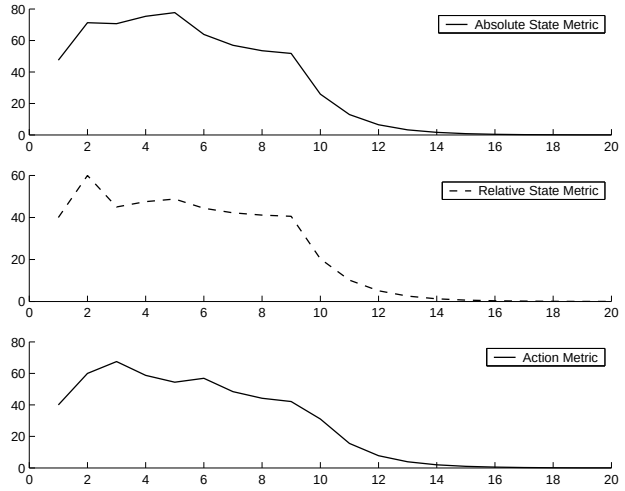


Fig. 8. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 4 (left), plotted during the simulation.**

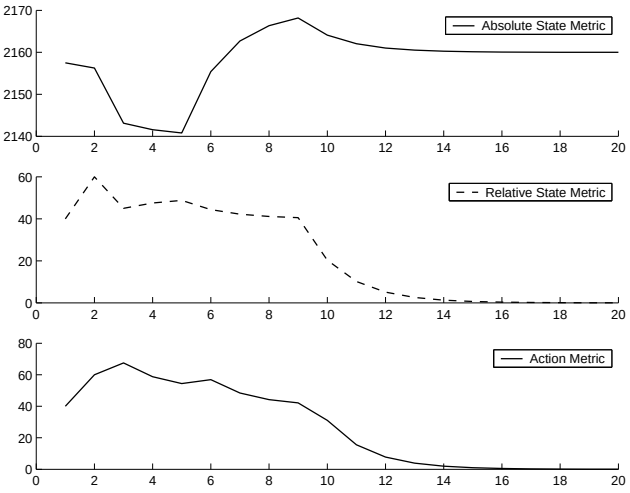


Fig. 9. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 4 (right), plotted during the simulation.**

As the imitator tries to minimize the relative state errors, the value of the relative state metric in all plots is shown to converge to zero. Depending on the embodiment and the correspondence mapping, the value of the absolute state metric also converges to a value that may (e.g. see Fig. 8) or may not (e.g. see Fig. 9) be zero. The value of the action metric trivially converges to zero (since towards the end of each simulation both the demonstrator and the imitator stop performing any actions).

C. Mapping to Robotic Target Platforms

Capturing human demonstration via the 'V'-perception mechanism (see section VI-A.4 above), implemented using the Polhemus Liberty motion capture system, and mapping to robotic target platforms (AIBO and robot arms – see Fig. 3) according to different correspondence matrices, as described

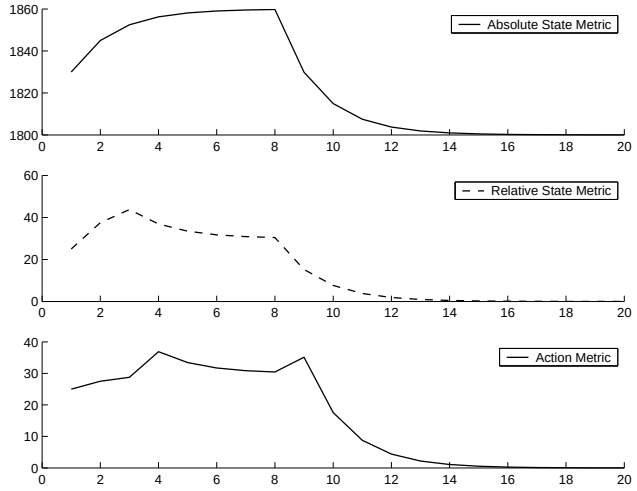


Fig. 10. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 5 (left), plotted during the simulation.**

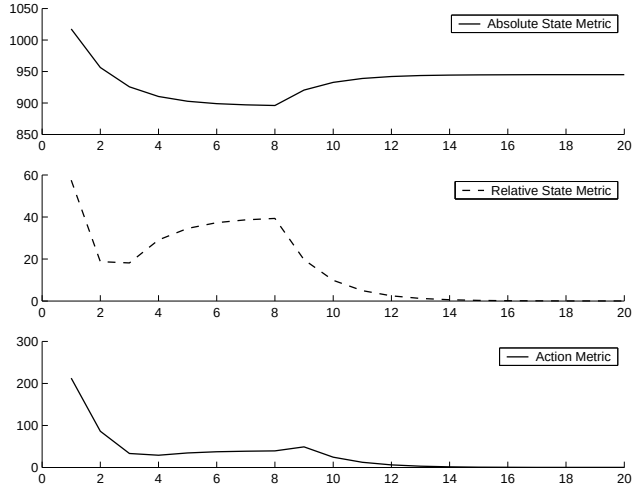


Fig. 11. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 5 (right), plotted during the simulation.**

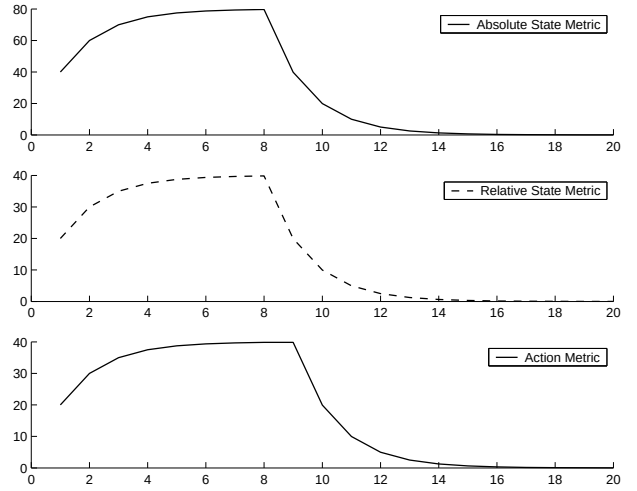


Fig. 12. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 6 (left), plotted during the simulation.**

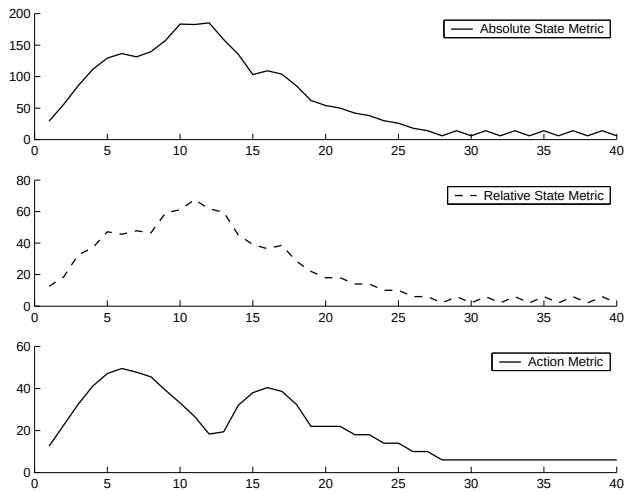


Fig. 13. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 6 (right), plotted during the simulation.**

above, allows the achievement of qualitatively dissimilar types of robotic imitation based on human demonstration.

VII. Discussion and Future Work

We have shown how partial, mirror symmetric, one-to-one, one-to-many, many-to-one and many-to-many body mappings can be characterized by (linear) *correspondence matrices*. These correspondences induce an infinite variety of absolute and relative state and action metrics that can be used to guide robotic imitation across dissimilar embodiments – even radically different ones in which neither the size of body parts, nor their type, nor number of DOFs need be preserved. The study of non-linear correspondences for achieving matching behaviour in states, actions and/or effects would extend this set of metrics.

Currently the correspondence mapping in this system is given for each experimental run, but finding the correspondence can be approached using reinforcement learning and an experiential history (adding memory), as in our previous work with the ALICE generic imitation framework [2], [1].

The system currently is purely reactive with no memory (and as a result has no learning). This results in certain limitations. For example in Fig. 6 (right), the demonstrator is moving its left arm on a trajectory loop, but since the imitator is continuously reacting but with a limited rate of movement, the imitator is unable to reach the current subgoal before it changes (until finally the demonstrator completes the entire demonstration). This inability to sustain appropriate actions is called *impersistence* [18] and can be solved by adding memory to the system, containing the sequence of subgoals.

Future work would naturally also address the derivation of, and switching between, appropriate correspondence mappings depending on the needs of the imitator agent in the social and task context. The developed system could eventually serve as

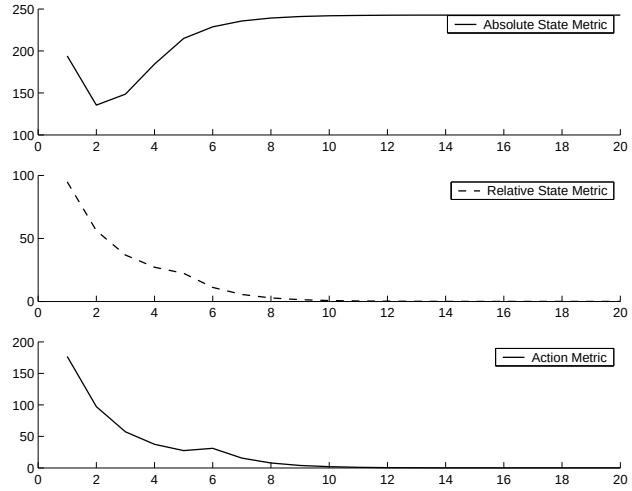


Fig. 14. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 7 (left), plotted during the simulation.**

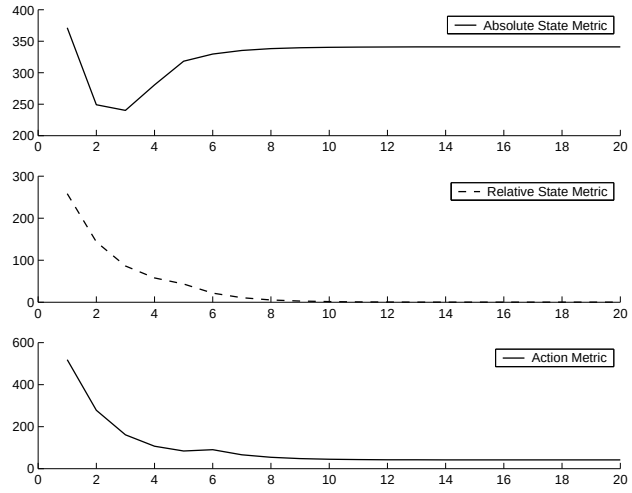


Fig. 15. **The induced *absolute state*, *relative state* and *action* metric values for the imitation example shown in Fig. 7 (right), plotted during the simulation.**

a *correspondence engine for imitation learning*, incorporating aspects of discovering *what* to imitate, depending on context and interaction history.

APPENDIX I
KINEMATIC EQUATIONS FOR THE MODELS

A kinematic model (as defined in section II) with n segments, can be positioned in the 3D workspace by assigning the values (x_0, y_0, z_0) as the coordinates of both the base and the tip of the root segment. Inductively, the Cartesian coordinates (x_i, y_i, z_i) of the base of each subsequent segment i are equal to the coordinates of the tip of its parent segment p_i:

$$x_i = x'_{p_i}$$
$$y_i = y'_{p_i}$$
$$z_i = z'_{p_i}$$

The Cartesian coordinates (x'_i, y'_i, z'_i) of the tip of each segment i can be found by using the spherical coordinates $(\ell_i, \theta_i, \phi_i)$ describing the current (relative) position of the segment:

$$x'_i = x_i + \ell_i \cos(\phi_i + \Phi_{p_i}) \cos(\theta_i + \Theta_{p_i})$$
$$y'_i = y_i + \ell_i \cos(\phi_i + \Phi_{p_i}) \sin(\theta_i + \Theta_{p_i})$$
$$z'_i = z_i + \ell_i \sin(\phi_i + \Phi_{p_i})$$

ACKNOWLEDGMENTS

The authors would like to thank Michael Walters and Assif Mirza for their help with robots for this article. Both are with the Adaptive Systems Research Group at the University of Hertfordshire.

The work described here was conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion" - `www.cogniron.org`) and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn, "Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments," *IEEE Trans. Systems, Man & Cybernetics: Part A*, vol. 32, no. 4, pp. 482–496, 2002.

[2] ——, "Towards robot cultures? – Learning to imitate in a robotic arm test-bed with dissimilar embodied agents," *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, vol. 5, no. 1, pp. 3–44, 2004.

[3] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders, "Imitating using JABBERWOCKY to achieve corresponding effects in context," forthcoming.

[4] ——, "Achieving corresponding effects on multiple robotic platforms: Imitating using different effect metrics," in *Proc. Third International Symposium on Imitation in Animals and Artifacts – Hatfield, UK, 12-14 April 2005*. Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2005, pp. 10–19.

[5] ——, "An approach for programming robots by demonstration to manipulate objects: Considerations on metrics to achieve corresponding effects," in *Proc. 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '05)*, 2005.

[6] H. Bekkering and W. Prinz, "Goal representations in imitative actions," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 555–572.

[7] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal, "Discovering optimal imitation strategies," *Robotics and Autonomous Systems*, vol. 47:2-3, 2004.

[8] A. Billard, "Learning motor skills by imitation: a biologically inspired robotic model," *Cybernetics and Systems*, vol. 32, no. 1-2, pp. 155–193, 2001.

[9] C. Breazeal and B. Scassellati, "Robots that imitate humans," *Trends in Cognitive Science*, vol. 6, pp. 481–487, 2002.

[10] G. Butterworth, "Pointing is the royal road to language for babies," in *Pointing: Where Language, Culture, and Cognition Meet*, S. Kita, Ed. Lawrence Erlbaum Assoc Inc, 2003, pp. 9–26.

[11] J. Call and M. Carpenter, "Three sources of information in social learning," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002.

[12] J. Demiris and G. Hayes, "Imitation as a dual-route process featuring predictive and learning components: A biologically-plausible computational model," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 327–361.

[13] L. M. Herman, "Vocal, social, and self imitation by bottlenosed dolphins," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 63–108.

[14] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observations of human performance," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 799–822, November 1994.

[15] J. Nadel, C. Guerini, A. Peze, and C. Rivet, "The evolving nature of imitation as a format of communication," in *Imitation in Infancy*, J. Nadel and G. Butterworth, Eds. Cambridge, 1999, pp. 209–234.
[16] C. L. Nehaniv and K. Dautenhahn, "Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation," in *Proceedings European Workshop on Learning Robots 1998 (EWLR-7), Edinburgh, 20 July 1998*, J. Demiris and A. Birk, Eds., 1998, pp. 64–72.
[17] M. N. Nicolescu and M. M. Matarić, "Learning and interacting in human-robot domains," *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, vol. 31, no. 5, pp. 419–430, 2001.
[18] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, "An experimental comparison of imitation paradigms used in social robotics," in *Proc. IEEE RO-MAN 2004, 13th IEEE International Workshop on Robot and Human Interactive Communication, September 20-22, 2004 Kurashiki, Okayama Japan*. IEEE, 2004, pp. 691–696.
[19] B. Scassellati, "Imitation and mechanisms of joint attention: A developmental structure for building social skills," in *Computation for Metaphors, Analogy and Agents*, C. L. Nehaniv, Ed. Springer Lecture Notes in Artificial Intelligence, Volume 1562, 1999, pp. 176–195.
[20] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

Aris Alissandrakis received his Ph.D. degree in Computer Science at the Adaptive Systems Research Group, Computer Science Department, University of Hertfordshire, England in 2003 and the M.Eng. degree in Cybernetics from the Department of Cybernetics, University of Reading, England in 1999. He is currently a postdoctoral research fellow in the European Integrated Project Cogniron ("The Cognitive Robot Companion"), studying imitation in the context of robot-human interaction.

Chrystopher L. Nehaniv received his Ph.D. degree from the University of California at Berkeley in 1992, and is currently Professor of Mathematical and Evolutionary Computer Sciences at the University of Hertfordshire, in Hatfield, England. He is a founding member of the IEEE Working Group on Artificial Life and Complex Adaptive Systems, and serves as Associate Editor for the journals *BioSystems: Journal of Biological and Information Processing Sciences* and *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, as well as Director of the U.K. Engineering and Physical Science Research Council and Interactive Network on Evolvability in Biological and Software Systems.

Kerstin Dautenhahn received her Ph.D. degree from the Biological Cybernetics Department of the University of Bielefeld, Bielefeld, Germany, in 1993. She is Professor of Artificial Intelligence in the School of Computer Science and coordinator of the Adaptive Systems Research Group at the University of Hertfordshire in England. She has published more than 100 research articles on social robotics, robot learning, human-robot interaction and assistive technology. Prof. Dautenhahn edited several books and frequently organises international research workshops and conferences. She is involved in several European robotics projects and is Editor in Chief of the journal *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*.

Experimental Comparisons of Observational Learning Mechanisms for Movement Imitation in Mobile Robots

Joe Saunders, Chrystopher L. Nehaniv and Kerstin Dautenhahn

***Abstract*— Research into robotic social learning, especially that concerned with imitation, often focuses at differing ends of a spectrum from *observational learning* at one end, to where a closer shared context is considered, for example, *following* or *matched-dependent behaviour* at the other. We study the implications and differences that arise when carrying out experiments both at the extremes and within this spectrum. Physical Khepera robots with minimal sensory capabilities are used, and after training, experiments are carried out where an imitating robot perceives the dynamic movement behaviours of another model robot carrying a light source. It learns the movement behaviour of the model by either statically observing the model, dynamically observing the model or by following the model. It finally re-enacts the learnt behaviour. We compare the results of these re-enactments and illustrate the differences and trade-offs that arise between static observational and reactive following learning methods. We also consider circumstances where, for this robotic embodiment, dynamic observation has both advantages and disadvantages when compared to static observation. We conclude by discussing the implications that arise from using and combining these types of social learning.**

***Index Terms*— Robot Imitation, Social Learning Paradigms, Social Robotics, Impersistence, Matched-Dependent Behaviour, Observational Learning, Correspondence Problem**

I. Introduction

ONE of the fundamental aims of robotics and AI research is the study of how intelligent behaviour can be acquired. We consider that it is the social dimension of behaviour that holds the key to making robots behave more intelligently [1], an approach inspired from studies of social animals (e.g. apes) and the 'social intelligence hypothesis' [2], which proposes that intelligent behaviour in primates has its origins in dealing with complex social dynamics.

Many within the robotics community have suggested that endowing robots with the ability to imitate may be a key attribute in capturing the essence of intelligence, for example [1], [3]. The benefits accruing from this research, from the ability of robots to imitate each other or humans both singly and in groups, would be easier robot task acquisition, increased behavioural complexity and a more natural human-robot interface. Ultimately some form of cultural transmission [4] may be possible with the hope of creating a virtuous circle - imitation leading to increased skill levels, which leads to more complex task behaviour which in turn leads to increased skill levels.

The studies described in this paper take an agent based perspective in focusing on the social dimensions in the interactions between demonstrator and imitator as well as the physical environment and internal/external perceptions.

An understanding of the mechanics of the behavioural transformation occurring in imitation has been described through the 'correspondence problem' formalism [5]. These concepts provide a basis for an investigation of social learning and the interaction between both human/robot and robot/robot pairs to understand the social dimension of imitative behaviour. It is this research which forms the starting point for the work described in this document.

A. Kinds of Observations for Social Learning

Our research considers the perspectives of both the imitator/pupil and the imitatee/teacher and the problems of perception and action encountered by both. There are *various kinds of observation* which can be involved in social learning listed below. The first three are situated along a continuum in which the movement decision is varied:

i) *reactive following* - the imitator closely follows the demonstrator. In this form of observational learning the imitator *must move*. Following is achieved reactively and thus the imitator does not maintain state or sub-goal information on the demonstrator position.
ii) *static observation* - the imitator remains stationary whilst observing the demonstrator and thus the imitator *cannot move*. A series of subgoals
are maintained by the imitator as waypoints in the demonstrators perceived path.
iii) *dynamic observation* - the imitator re-enacts the model's behaviour and *may move* and/or rotate whilst still observing the model's movements. State/sub-goal information is maintained through way points. However the number of these may vary dependent on the decision as to either observe or move.

Two further mechanisms can augment any of the above:

iv) *predictive observation* - the imitator attempts to predict the movements of the demonstrator based on movement patterns already known to the imitator.
v) *direct teaching* - the demonstrator provides feedback and/or actively helps the imitator/learner to learn the demonstrated behaviour.

This paper focuses on the first three items above in comparing static and dynamic observational learning and matched dependent or following behaviour as parts of a spectrum of imitative

perspectives. We examine the relative tradeoffs, implications and impacts of these different types of social learning which are known to occur in nature. We do not aim to select one as best, but to understand the circumstances in which a particular method may be appropriate in different contexts. In particular we exclude types of social learning involving feedback, scaffolding or any form of direct instruction from a teacher. These types of social learning, corresponding to items (iv) and (v) above, are examined in our other work [6]. Instead we focus on the case where the model acts independently of the imitator, observes it and learns.

II. Social Learning Paradigms

A. Following and Observation

From a psychological/ethological viewpoint reactive *following* is more rightly considered as a form of *matched-dependent behaviour* [7]. For example rats can be trained to follow a lead rat through a maze which they then learn to navigate [8]. The rats may have no idea of intentionality of the lead rat and can be trained to follow any other salient (including non-animal) stimuli. Sometimes this behaviour is called *discriminated following* [7].

Observational learning is a common mechanism in many instances of social learning by animals and humans. We can classify different types of observation; firstly *static observation* where the behaviour of the demonstrator is copied after it is observed carrying it out and the observer does not move. Secondly, *dynamic observation* where the observer can move, the behaviour of the demonstrator being re-enacted either after the demonstration (immediate imitation) or during it (synchronic imitation). Typically the demonstrator and imitator operate within a shared context but at a distance from one another. For example Norway Rats apparently develop food preferences by smelling the breath of a conspecific [9], without reference as to whether the demonstrating rat becomes ill or dies. The mechanism is very simple - perceive what others do and imitate it [10].

'Following' can be equated with experiencing a very close shared context with a demonstrator, such that the imitator "feels" and experiences similar internal and external sensory feedback to that of the demonstrator. Observation however is assumed to take place where the shared context is more distant, especially so if the observer does not move, and therefore does not experience environmental sensory feedback directly. Where observation is more dynamic, for example where the observer starts to imitate the actions of the demonstrator before the demonstrator has finished its demonstration the context again becomes closer.

It would seem sensible to assume that animals use both mechanisms as necessary. This appears to be likely if we examine our own human responses when attempting an imitation event.

In some instances observation alone is sufficient to duplicate a task. For example, if another person were to draw a familiar geometric shape in space such as a circle or triangle most people would easily duplicate this shape, they may wait for the complete shape to be drawn and then repeat it themselves. Sometimes the replay may begin before the demonstrator has finished although this may be more difficult to achieve if the shape is ambiguous. However observation alone may be insufficient, for example when the shape is complex or previously unknown (e.g. a character in Arabic or Japanese to a user of the Roman alphabet) or if the imitation depends also on knowing the 'feel' of something e.g. imitating a musician or a delicate piece of calligraphy.

These examples signal some interesting but not widely researched features of imitative behaviour in the relationship between static or dynamic observation of, and active participation in, an event to be imitated. In the simple shape example there is observation followed by imitation, with the imitating event either happening after the termination of the demonstration or part way through it. In the second example of a more complex shape, observation is usually insufficient to completely duplicate the task and further mechanisms are employed – the demonstrator may slow down and start to teach the imitator, and the imitator may start to directly track or follow the demonstrator very closely in order to capture the movements.

B. Paradigm Examples in Robotics

Both the following and observational mechanisms are used in robotics imitation research. For example [11], [12] both used following to replicate a demonstrator's actions and to improve an imitator's learning capability. Following is successful since it allows the imitator to more closely share context as the demonstrator performs the behaviour. As the imitator follows it can map its sensory experiences directly to its motor outputs, which are matched to, and depend on, the demonstrator's actions. It can thus learn the necessary perception-action couplings directly and use them in similar situations in the future without the teacher being present. Following is used in [13] as a means of testing an imitation model based on reducing perceptual errors, the imitative behaviour being achieved by continually adjusting motor outputs when presented with a difference between perceived states and goal states.

The observational paradigm is also extensively used, often employing complex vision processing in for example [14]. Both physical and simulated robots are used in [15] to statically observe a continuously changing game (air hockey or marble maze). Observation is used in [16] to recognise what to imitate in attempts to build robots that imitate people.

From the imitator robot's viewpoint these are very different perspectives and it is highly likely that improved robot-human interaction may be possible if the characteristics and key issues involved in these learning paradigms were better understood. We begin this process by offering some observations comparing the social learning paradigms in an implementation study with Khepera robots.

III. Framework

An imitation framework for describing the sequences of *actions*, *states* and/or *effects* can be described using the *correspondence problem* formalisation [17]. The correspondence problem formalism allows the model's given actions, states

and/or effects and the desired actions, states and/or effects in the imitation sequence to be compared using metrics or measures of dissimilarity; the number of matched actions, states and/or effects, i.e. the fineness of the imitation attempt can be described as a measure of granularity.

Our focus is to compare the following and observational paradigms. To do this we simplify the context of the imitation and restrict the actions/states and effects. In the experiments we use a physical Khepera-1 robots as both imitator and model. A Khepera is a 5cm diameter, non-holonomic robots equipped with 8 Infra-red sensors arranged at intervals around its circumference. The imitating Khepera perceives the model Khepera by means of a bright light mounted on the model. The goal for the imitator is simply to replicate movements of the model. Actions are limited to two possibilities: either turn by a given angle and/or move a given distance. States are described as the perceived vector from the centre of the robot to the light. For example figure 1 shows the state/action sequence derived if the model's actions were to describe a triangle.

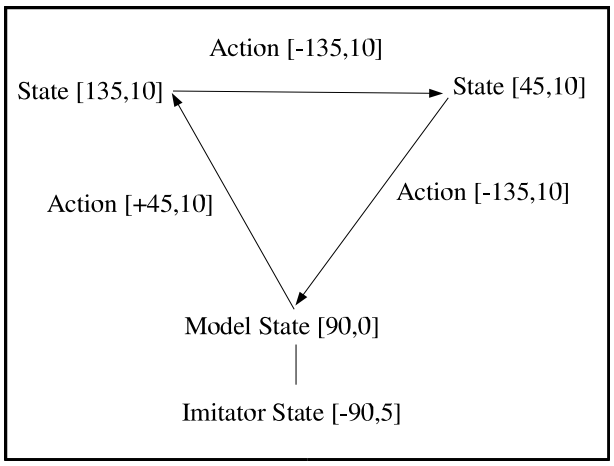


Fig. 1. *States and Actions from Model's Initial Perspective. States are specified by (degrees θ, distance r (cm)). The triangle shows each state achieved by executing each action. The model is initially facing forward at 90°. The imitator is shown placed behind the model at -90°). Action can be interpreted as the necessary movement $(\Delta\theta, \Delta r)$ to achieve the given state (θ, r).*

A. Experimental Outline

In the first experiments learning via a purely reactive following behaviour is contrasted against learning via a static observational behaviour using Khepera miniature robots. These ideas are further developed with experiments investigating the effects of allowing a more dynamic observational approach. These successively augment static observation, respectively, by adding orienting rotational changes to allow the imitator to track the demonstrator or by adding rotation and translation. In all of the experiments both learning and re-enactment of the imitation is performed and the imitation attempts subsequently compared. The four experiments performed are as follows:

i) *reactive following* - the imitator shares close context with the model. In this experiment the Khepera acting as imitator closely follows the khepera acting as model as it transcribes a geometric shape. The imitator then re-enacts the movements of the model.
ii) *static observation* - the imitator remains stationary whilst observing the model, the context is shared but distant. The imitating Khepera remains in place and observes the moving Khepera. Once the observation is complete the imitator re-enacts the model's movements.
iii) *dynamic observation with rotation only* - this behaviour augments static observation by allowing the imitator to rotate in place whilst observing the demonstrator. In this way the imitator can track the model throughout the two-dimensional movement space, however it must cope with the additional complexity of reference frame adjustment and motor control. Once the observation is complete the imitator re-enacts the model's movements.
iv) *dynamic observation with free movement* - the imitator can rotate and move during the imitation sequence. The imitation re-enactment starts before the demonstrator has finished its movement. This ensures that the imitator maintains closeness of context depending on the length of the observation period prior to moving, compared to the movement period prior to the next observation phase. The smaller the observation phase the closer the shared context.

There are a number of distinctions between between the reactive following and the static and dynamic observation experiments. Firstly the observation experiments maintain an internal set of subgoals (held as way-points) which are computed and subsequently used in the behaviour re-enactment. The number of sub-goals varies between the static and dynamic forms of observation. The former maintaining a full set from its complete observation of the demonstrator, the latter a partial set dependent on how far forward the observation is allowed to proceed before movement of the imitating robot is attempted. The reactive following experiment does not maintain any state and therefore has no sub-goals during following. The first two experiments additionally investigated the effect of further environmental contraints on the robot. A number of small plastic strips were placed in the robot's path which effectively slowed it down when it was moving at constant speed. The demonstrating robot was programmed to accelerate over these strips to keep its velocity constant by balancing the additional acceleration against the slowing effect of the strips. For effective imitation the imitator should replicate these motor changes at the same point on the transcribed shape during the re-enactment.

To facilitate these behaviours three controllers were designed. The first two controllers were for the observation experiments and were based on calculating angles and distances to the model. The third controller was for the following experiment and was based on calculating vectors to the model. In all cases the imitating robot is equipped with the standard 8 sensors capable of measuring both infra-red at short distances (around 10cm) and ambient light to longer distances (around 30cm).

All experiments were carried out in real-time on physical robots (i.e. simulation was not used) on a desktop in a typical busy academic laboratory environment with light levels varying during the day. The model robot was programmed to follow a simple script which described the appropriate geometric shape. The controllers for the both robots ran on a PC with commands sent to the robot by radio links or via a serial cable. We examined the behaviour of the imitator when imitating triangles, circles and letter shape patterns.

B. Controller for a Following Robot - experiment (i)

We used a 'Motor Schema Vector Fields' methodology [18] to facilitate *following* behaviour. As the robot observes a moving light source the perceptions afforded by the sensors are converted to motor outputs. In this approach following the model is achieved with an *attractive* vector which provides both the direction and magnitude of the light source. We poll the front six IR sensors of the imitator to provide an ambient light sensor reading. For each sensor reading a vector is formed using the sensor reading and the fixed angle of the sensor in relation to the body of the robot. Each vector is formed by comparing the sensor reading against a fixed sensor range. If outside the sensor range the vector components are set to zero, otherwise the vector angle is set to the sensor angle and the vector magnitude is set to $(1 - \frac{sensorReading}{sensorRange})$. Each of the six resultant vectors, one for each sensor, are then summed. The magnitude of this vector gives a proxy for distance to the robot within the specified range. The angle of this vector provides the angle of turn to the light source. If the light source is within the specified range the magnitude is used to provide a velocity to the robots actuators. The natural curve of the light envelope read by the sensors means in effect that the further away the robot from the light the faster the robot will move. By applying an additional gain factor to this magnitude it can be balanced against a repulsive vector described below.

The *repulsive* vector works in the opposite way to the attractive vector. Here the the IR sensors are used to detect obstacles at a distance with 0-10cm from the model. As for the attractive vector the sensor readings and fixed sensor angles are formed as vectors and summed to yield a single vector. Here the magnitude of the result increases as the imitator approaches the model. This magnitude is also made subject to a gain factor (note that this vector contributes to the imitator's behaviour not the model's). In order to assess "pure" following and observational learning methods the model does not modify its behaviour in response to the imitator i.e. there is no feedback or directed teaching in these experiments (unlike Billard and Dautenhahn [11] above).

The addition of the attractive and repulsive vectors after adjustment by the appropriate gain factors allows the forces to be balanced. This ensures that the imitator does not bump into the model and that it stays at an approximately fixed distance from the model if the model is moving at constant speed. If the model accelerates away from the imitator the imitator will respond by accelerating to catch up with the model. Similarly the imitator will slow down if the model slows down.

The light angle returned from the attraction vector is used so that the higher the displacement angle the faster the robot will turn towards the light. This speed factor is controlled by a further gain parameter. If for example, the gain is set high, then the robot will turn towards a static light very fast but often overshoot the correct position. Conversely if the gain is low the robot will move slowly but accurately towards a static light without an overshoot. This parameter can then be used to control how quickly the robot responds to changes in direction of the model.

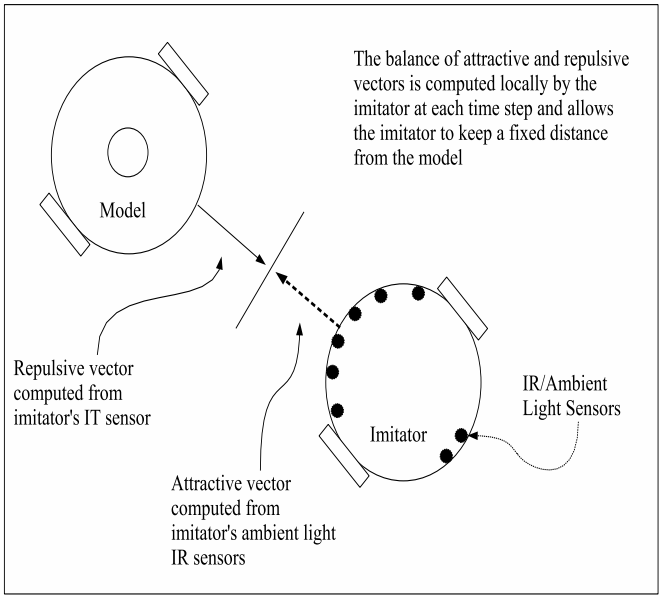


Fig. 2. *Using Potential Fields to Faciliate Following Behaviour. The diagram shows both Khepera robots. The demonstrator has a small light bulb on top of it. The imitator computes the sum of an attractive and repulsive vectors locally at each time step. The angle of the resulting vector provides a turn direction and the magnitude provides the velocity for the imitator.*

C. Controller for an Observing Robot - experiments (ii),(iii) and (iv)

Learning to Measure Angles and Distances. Before any imitation, it is necessary for the imitator to be able to perceive the model's behaviour - in particular the angle and distance to the model from the imitator's perspective. To do this, we implemented and assessed several methods (see Appendix I) including the *differential light compass* [19] which is similar to computing the angle by using *vector summation* of the inputs to each of the light sensors [18] (described in the section on 'Following' above) and a new method called *environmental sampling*. The latter method proved superior for the Khepera platform and was subsequently adopted. It is based on automatically building a feed-forward neural network based on the sensory modalities of the particular Khepera robot and weighting the connections between the network nodes on a normalised sensor vector obtained from sampling the light source at various intervals. Distance measurements used a similar sampling method based on the angle measurement above and recorded the sensory state of the robot at fixed distances from the light source. These measurements were then used to form a lookup table that was subsequently sampled by the robot when observing distance. *Environmental sampling* is

described in detail in Appendix I.

Observing Angles. After the learning phase is complete the system operates by feeding a normalised sensor vector to the input layer of the dynamically built neural network and receiving an angle from the output layer. The network effectively operates as a pattern matching mechanism. Automatic interpolation between observed values is achieved by setting the middle layer 'winning nodes' to a value greater than 1.

Observing Distance. During the observation phase the angle is computed, followed by magnitude of the vector summation, the two values providing the key to the lookup table to yield distance.

Altering the Angle of Observation. In experiments (ii,iii and iv) the robot collects a set of angles/distances from itself to the model whilst the model is moving. The imitator does not poll its sensors when it itself is moving. Thus in a fixed time period the number of possible observations when the imitator is not moving will be higher than when the imitator is moving. In experiment (iii) the imitator can either not move or rotate to face the model once a threshold angle has been exceeded (see figure 3).

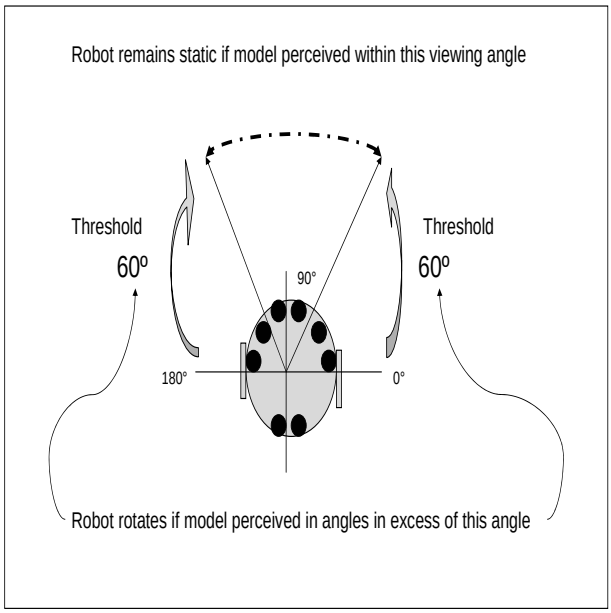


Fig. 3. *Rotation Threshold. In this example the imitator will not move at values between* 60° *and* 120°*. Outside* 61° *and* 121° *the imitator will rotate to face the model.*

The lower the threshold angle the greater the rotational movement of the robot to face the imitator when the angle is exceeded. The higher the angle, the smaller the rotational movement, but the robot will move more often.

In the static observation experiment(ii) the position of the imitator was fixed and the model was at all times in front of the imitator and therefore within range of angle/distance computation mechanism. In the dynamic observation experiment(iii) the model was allowed to be both in front of the imitator and at any angle around the imitator. By varying the rotation threshold we examined both the effect of rotational movement size and the effect of frequency of movement on the imitation attempt.

Time Averaging and Way Points. In both the static and dynamic observation experiments(ii,iii and iv) the recorded observations are smoothed using a simple moving average. The smoothed trajectory is then thresholded to yield a set of way points. This procedure is necessary for two reasons. Firstly to eliminate the effect of noisy observations and secondly to avoid two observation points being too close to one another - this closeness causing large and potentially damaging changes in the robot's motor systems if replayed directly. The imitator uses the derived way points to then imitate the model's trajectories. In the fourth experiment this procedure is only applied when computing the required movement. Any unused observations (which result from the movement index being less than the observation index - see *observe and move* experiment below), remain unmodified as these may be subject to geometric transformation following the actual movement of the imitator.

IV. Results

In our experiments we compared behaviour on three simple imitations. These were a triangle, circle and the letter T. The triangle was chosen because of the sharp changes of direction at each vertex, the circle because of its continuous shape and the letter T because of the need to reverse direction and remap the shape. We emphasise that our goal was not to design robots that perfectly imitate geometric shapes but rather to understand the role of the method of observing and its consequences for the imitation attempt in the various learning paradigms.

A. Following

Observations of robot behaviour using the 'following' controller highlighted some issues with using a simple reactive architecture, one of which is a familiar problem for path following mobile robots [20], [13]. This is where the robot fails to follow the path of the imitator with precision due to the tight reactive cycle between the sensors and the motors. In our experiments this is shown in figure 4: for the triangle and circle the robot either cut the triangle corners or inscribed a smaller circle inside the model circle. The T-Shape exhibited this cutting problem and also showed the imitator *avoiding the model* when the model started to reverse towards it when inscribing the T-shape. These problems arise due to what we call *impersistence*, i.e. the inability to sustain appropriate actions, and is a consequence of the imitator always reacting to the latest perception vector whilst effectively 'forgetting' the previous perceptions. This is a symptom of the reactive following method not holding state or sub-goal information. Also the reaction of the imitator's motors always lags behind that of the sensors (for example the imitator may have partly turned towards the model when a new reactive cycle begins).

The *impersistence problem* in robotics appears to be unsolved, but partial solutions have been attempted by providing further dampening to the system, by delaying the imitator response, slowing down the model or alternatively using 'vector' or 'pure' pursuit methods (see [20]) although both the latter methods rely on accurate measurement of a distant 'goal'

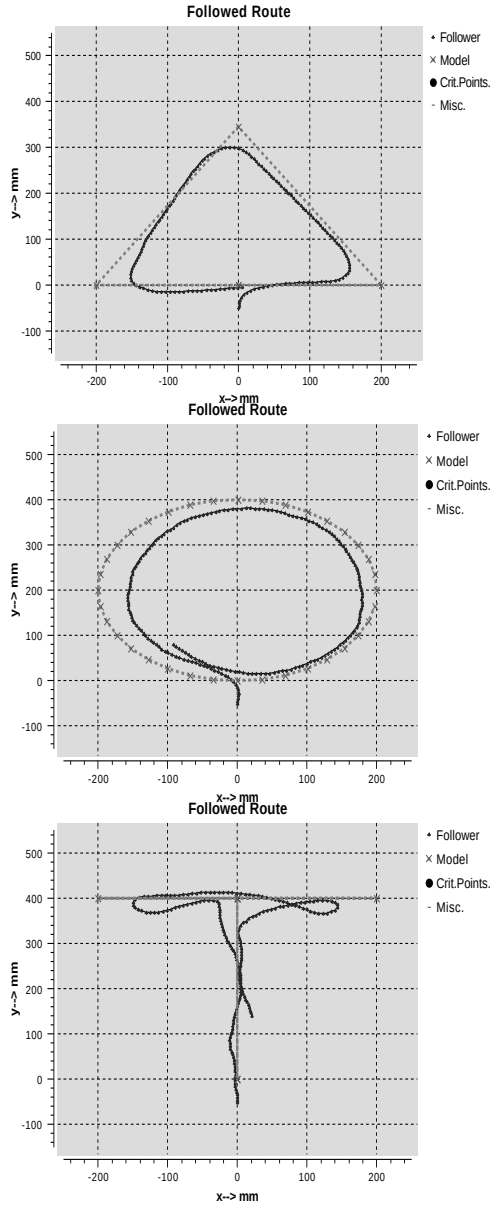


Fig. 4. ***Following Behaviours for triangle, circle and T shape.*** *The graphics represent typical behaviours of the following mechanism. Dotted line is the model, bold line is the imitator.*

point. A novel solution proposed by [13] uses an additional camera which can move independently of the robot body controlled through a biologically inspired neural network. However precise imitation, both when and after following, is difficult and the solutions can limit the behaviours of both model and imitator severely. Indeed, there is a deep issue relating what an imitator's sensors are telling it and what it is actually doing when following. A further issue is that as the imitator reacts, its reference frame is changing in relation to the model. Calculation of these changes is not possible from using the reactive vector alone. This means the robot cannot retain a memory of the imitation from its sensor perceptions; it can however do so by storing its motor outputs (i.e. the velocity of each wheel during each time-cycle). Thus it imitates what it 'feels' rather than what it 'sees'. However this can be beneficial when imitating the model moving at constant speed over the plastic strips. An imitating Khepera initially moving at constant velocity will typically slow down when moving over these strips. However, when following and in order to maintain contact with the imitator the reactive vector will produce an increase in motor outputs as the model moves away from the imitator. This change then becomes part of the memory of the imitation. Thus the imitator manages to imitate more faithfully by reference to its internal state rather than an external observed state.

B. Static Observation

A similar set of tests for the static observation experiment are shown in figure 5. The results appear promising with the robot imitating the model with reasonable accuracy, with vertices accurately tracked and little evidence of the impersistence problem.

However a number of points should be borne in mind. Firstly the performance of the tracking mechanisms depends crucially on how the observed data is filtered. Time averaging and a significant event extraction are both based on thresholds. The choices of threshold have to be carefully chosen to achieve this level of accuracy. Secondly, the method required to compute the path is complex as compared to the relatively simple vector summation methods used in the following experiment. Finally and most importantly, by using observation alone the imitator would have no way of learning any parts of the imitation which were not addressable from observation alone. For example, the static imitator only records a constant velocity when observing the model traversing the tape strips discussed in 'Following' above; when imitating it fails to increase its motor outputs over the strips to maintain velocity. It seems that this need to modify motor outputs may only be obtained from experiencing the situation.

C. Dynamic Observation - Observe and Rotate.

The first dynamic observation experiment extends the static observational perspective by allowing the robot to alter its orientation so as the better exploit it sensory facilities. The embodiment of the Khepera robot is such that the majority of the light sensors are in front of the wheels, with two sensors at the back. The estimation of distance is therefore more accurate when the robot is able to employ all of its front facing sensors as it is receiving more information from the environment. To ensure that these sensors are in an optimal position we program the circular Khepera robot to rotate in place orienting toward the model. The rotation is such that the imitator will attempt to directly face the model if the model's angle with respect to the imitator exceeds a given threshold. However, if the imitator has to rotate to achieve this then all subsequent observations must be converted back to the original reference frame in order to replay the imitation. To achieve this conversion, accuracy in measuring how far the robot has turned is critical to this process. We tested threshold angles of 0, 30, 60 and 90 degrees. In both this and the experiment described below the model was preprogrammed to make 4 geometric shapes. The first was a 10cm radius

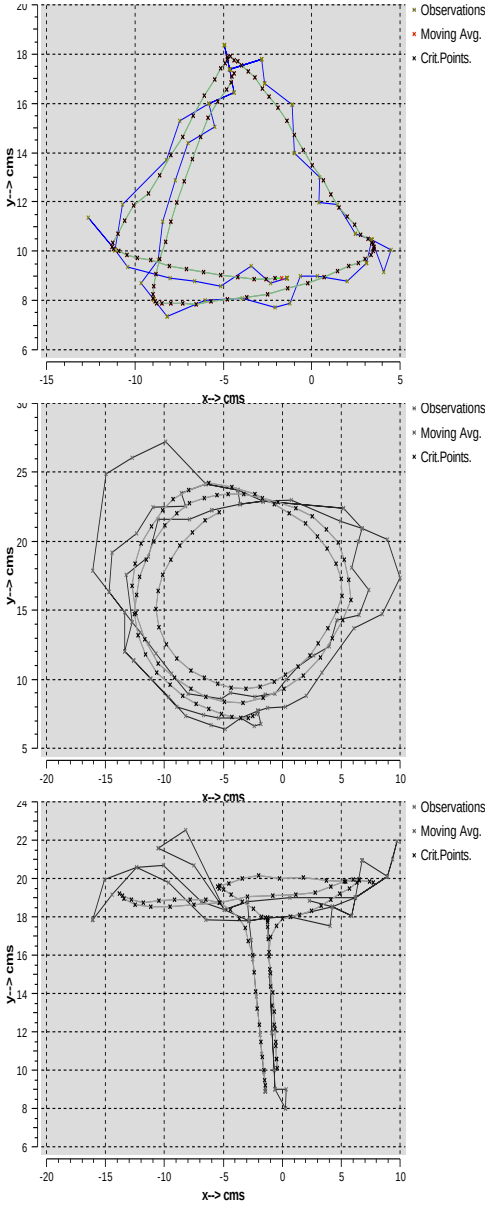


Fig. 5. ***Observation Behaviours for triangle, circle and T shape.*** *Dotted line is imitator's imitation attempt. Unbroken line is unfiltered observation. Based on 100 observations at 100ms intervals, 10 point moving average.*

circle around the imitator, the second a 10cm circle 5cm in front of the imitator. The third and fourth a triangle and T-Shape 5cm in front of the imitator.

Details of Set-up. In each case the imitator is placed at the centre of the experimental platform (shown as point 0,0 on the graphs in figure 6) facing forward (at 90° along the positive Y-axis). The model is pre-programmed to move according to the prescribed shape. A threshold rotation angle is then set and the imitation run commenced for a fixed period. The threshold supplies a range of values around the front of the robot. For example, setting a threshold of say 60° means that if the imitator perceives the model within a forward range of $60-120^\circ$ (see figure 3) no rotation will be applied.

If however the model moves to, say, 50° the imitator will rotate so that the model is directly in front of it, and thus be, from the imitator's new perspective, at 90°. The higher the threshold angle (to the limit of 90°) the more often the imitator will move to match the model but it will rotate by a smaller amount. If the threshold is set to zero, then the imitator will only move when the model is outside the range $0-180^\circ$, however the robot will then rotate by at least one quarter of its circumference.

Results. Figure 6 shows the results from a test with the enclosing circle. The robot is placed facing forward along the positive Y-axis. After the run the imitator robot attempts to re-enact the observed behaviour. The large dots on the graphs show the way points, these being the path that the imitator will take when replaying the imitation. The first graph shows the imitation when no rotation has been applied and thus where only static observation is taking place. As expected at angles outside the angle/distance range the imitation is poor. The second graph shows the first example of a dynamic observation with the imitator moving only when the model moves outside the range $0-180^\circ$. Two extreme observation points are shown reflecting the inability of the distance/angle sensor to correctly measure the distance. However, once the 180° or 0° angle is exceeded the robot turns and starts again to make reasonably accurate readings. On imitation replay the outlying readings are smoothed away. The situation is further improved at 30° when the sensory apparatus is always in range but the number of moves small. However at 60° and 90° the situation is ambiguous. We would suggest that the imitation is slightly less accurate. This may be due to the increasing effect of odometry errors as the number of moves increases. This is especially true at 90° where there would be a small movement for every 1° change on the model's position.

Results for the forward circle and triangle (not shown) were less marked, however the robot was subject to less movement due to the constrained angles presented by both shapes. The nature of the T-shape (see figure 7) meant that at 0°, 30° and 60° the imitated trajectories were broadly similar, however at 90° the robot was affected again by odometry drift and a similar worsening of readings ensued.

Analysis. These effects show some of the advantages and disadvantages of a tracking mechanism described above. Observing whilst not moving (static observation) has the key advantages of being fast and thus able to make more observations in a given time period (given the sequential nature of the observe/move scenario presented here). There are no odometry concerns as the imitator is not moving and the energy required would be lower than for a moving imitator. The major disadvantage is of course that the model can move into imitator blind spots. The advantages of the tracking imitator (dynamic observation with rotation only) is that blind spots can now be seen, however this is offset by the disadvantages of increasing odometry errors as more movement is carried out, a higher energy cost, and more complex computation as reference frame adjustments are continuously required. However at a particular movement/rotation ratio, which for this robot appears to be around 30°, there appears to be a point where accuracy is optimised. This suggests an clear

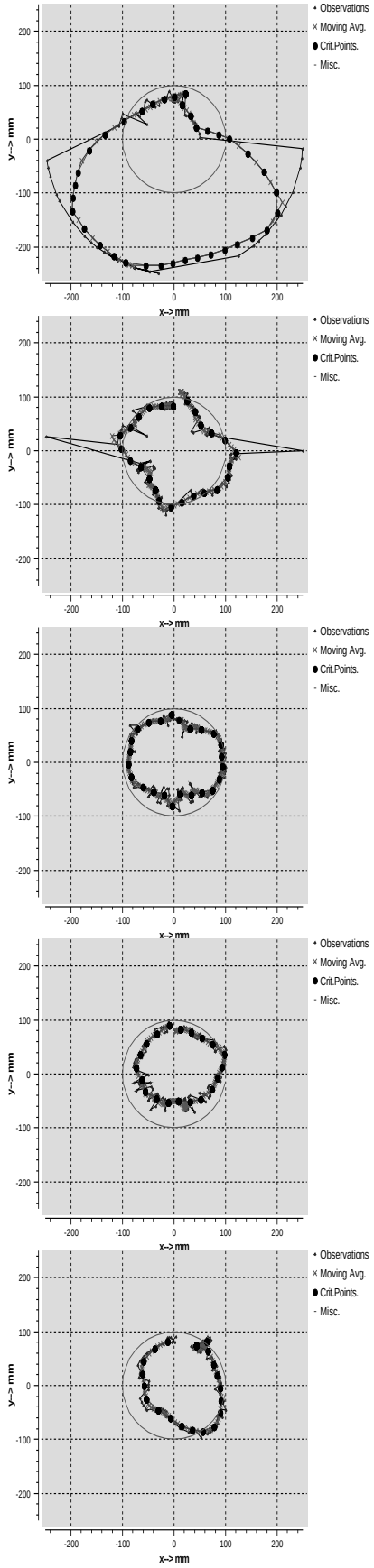


Fig. 6. ***Dynamic Observation with Rotation Only.*** *Imitative Behaviours for an enclosing circle. The first diagram shows the result with no rotation, the final four graphs show rotation at* $0°$*,* $30°$*,* $60°$ *and* $90°$ *degrees thresholds. The continuous line shows the path of the model, dotted line the imitator observations, crosses the smoothed observation and large dots the way points which are replayed by the imitator.*

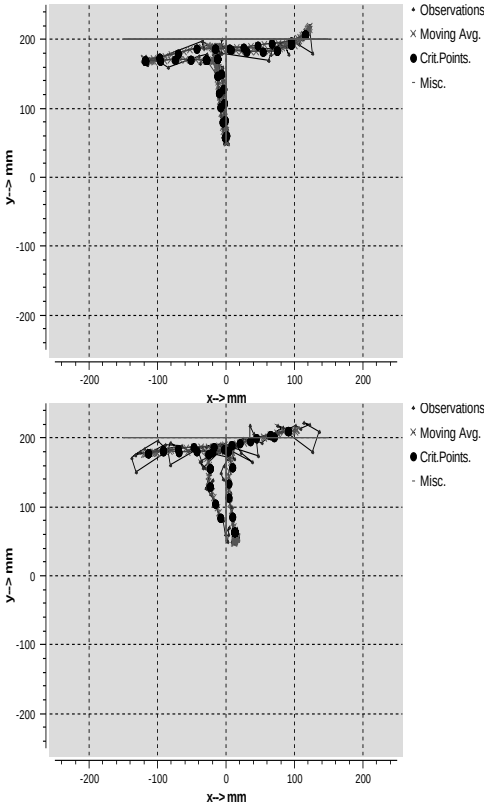


Fig. 7. ***Dynamic Observation with Rotation Only.*** *Imitative Behaviours for T-Shape The two graphs show results with rotation at* $60°$ *and* $90°$ *thresholds.*

strategy - expend energy and computational costs by moving only when not to do so would give incorrect results. Or more simply - keep still until movement is almost necessary (in our case when the model goes beyond the $30°$ threshold into 'peripheral' vision).

D. Dynamic Observation - Observe and Move.

The second dynamic observation experiment allows the robot to record a sequence of observations of the model and then attempt to use a given subset of these observations to imitate the model's movement sequence. Once the imitator has completed this part of the imitation it recommences observing.

In a two-dimensional parameterisation of the spectrum, different social learning mechanisms are given by varying both the number n of *observations* and the number m of *movements* made by the imitator. A single observation is an estimation by the imitator of the model's angle and distance from the imitator. A movement is the transformation and execution by the imitator of observations to motor-commands in order to achieve the same effect.

These mechanisms however present a series of challenges due to the fact that after each movement sequence the robot's memory of previous observations will be from a different perspective from the current observation set. This is because the imitator, after partially replaying the imitation (by transforming a subset of the observed vectors) will find that the remaining observations need to take account of the new observation position. Furthermore, the new observation position

may not be optimal for accurate readings, therefore a rotation (as in the *observe and rotate* experiment) will be necessary. To then replay the next part of the imitation the effect of the rotation must be reversed and subsequently a transformation of the observations re-performed.

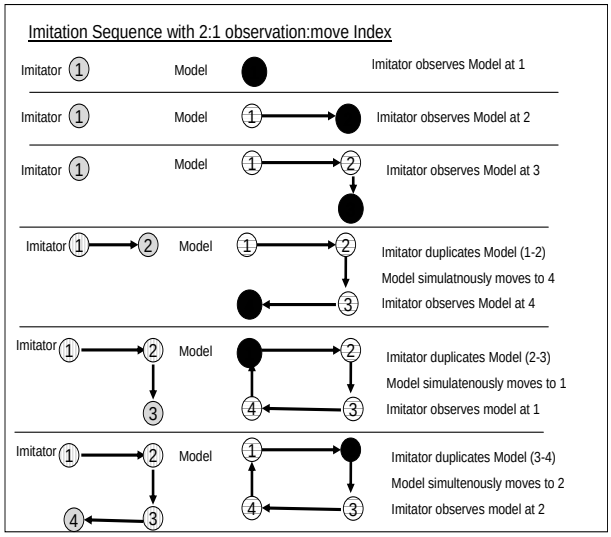


Fig. 8. *Analysis of Observation and Movement Index. In the analysis the model describes a square pattern. The imitator uses an observation:movement index of 2:1 and successfully matched the square. Similar successful matching will always occur when the movement index is set to 1 regardless of the observation ratio.*

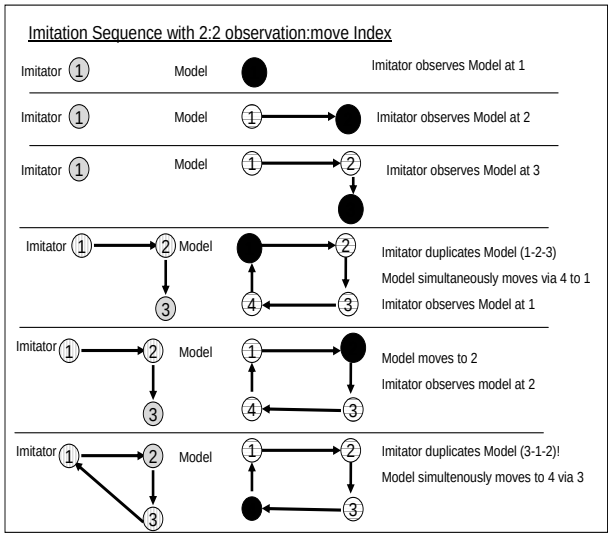


Fig. 9. *Analysis of Observation and Movement Index. This example shows the imitator an index of 2:2. When the movement index is set above 1 the imitator always fails to match the pattern.*

E. Dynamic Observations: Varying Observation and Movement Cycles

Theoretical Results and Detailed Set-up. This experiment explored how movement and observation might be intermixed. This was attempted by varying the number of look-ahead observations against an equal or smaller number of moves. Thus the robot would first make an initial observation [1] and then subsequently observe for n cycles and then move, based on these observations, m times. This procedure iterated throughout the imitation attempt. Prior analysis of this method, using the imitator and model represented as points (see figures 8 and 9) suggested that accurate imitation may only be possible if the number of moves were set to 1. To simplify the analysis we assumed that the imitator and model moved at approximately the same constant speed. Additionally, due to the control system of the robot, observation and movement execution are not possible in the same time step. We then imagined three scenarios cyclically alternating n observations (o) of model transitions with m moves (x) by the imitator (note: it is not possible to imitate further than our observed sequence, and therefore n is always larger or equal to m). The first scenario was of n observations to 1 move e.g. 1:1 *o-o x o x o x o x* ..., secondly a scenario where there are an equal number of observations and moves but where both are greater than 1, e.g. 2:2 *o-o-o x-x o-o x-x o-o x-x* ... and finally where n is greater than m and both are greater than 1 e.g. 3:2 *o-o-o-o x-x o-o-o x-x o-o-o x-x* .... Figure 9 shows an example of the failure to correctly match the movement pattern when the move index is set higher than 1. This occurs because the imitator has failed to observe one or more critical points in the model's move sequence. The effect is similar to the *impersistence* problem we noted when analysing 'following' behaviour [21], however rather than failure to complete or persist in its goal, as was the case for following, here the problem is one of 'inattentiveness'. The imitator is blind to the moves of the model. This problem occurs at all values of n and m which are larger than 1.

Results. The robot was tested on a series of index values on each geometric shape presented by the model. Figure 10 shows an example of the physical robot using a 5:1 n:m index on the triangle shape. The imitator fails to match the model. Similar failures occurred in all attempts with the physical robot on all shapes. This was initially surprising, however the difficulty became clear once the actual imitator movement was considered.

Analysis. The simplicity of the point analysis above hides some crucial implementation issues. For example the robot can only move in the direction of its fixed wheels (i.e. it cannot arbitrarily move sideways), therefore a rotation may be necessary to orient the robot to the correct movement vector signalled from the model. Also, the Khepera has a fixed placement of sensors around its circular wheelbase. In order to correctly 'focus' on the model the robot must be in the appropriate sensor range. Thus the rotation mechanism described in *observe and rotate* was employed.

Therefore in addition to the move or moves calculated from the observations we may have up to 2 additional moves: one to focus the sensors on the model and the other to orient the imitator for its move. Whilst these moves are being carried out the problem of 'inattentiveness' is compounded. Two further issues were also apparent. Firstly, each move is accompanied

[1]For each move two observation vectors are required, therefore at the start of the run one additional observation is made.

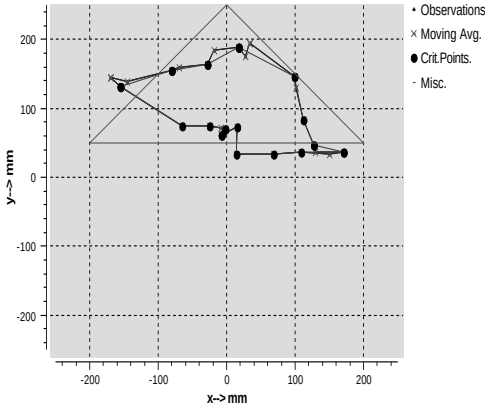


Fig. 10. ***Dynamic Observation: varying Observation and Movement cycles*** *The results show the inability of the imitator to correctly re-enact the model's trajectory - in this case a triangle shape.*

by a small odometry error. The total error therefore increases as the number of moves increases. Secondly, the smoothing effect of time averaging has little or no effect when attempting to model a small number of moves. This means that unsmoothed noisy observations are replayed leading subsequently to a poor imitation attempt.

We believe that the failure of the imitation is due primarily to the constraints imposed by the embodiment of this particular robot, however non-holonomic robots with fixed sensors are typical of many mobile robots. The issue may be obviated if the sensory apparatus was independent of the actuator mechanism e.g. distance/angle sensors which rotated and were focusable independently of movements of the main robot body. Such a mechanism is in fact used by [13] in their experiments. In fact there are no known imitating animals whose observation sensors cannot focus at least somewhat independently of the orientation of their bodies.

V. Summary and Discussion

A. Following and Static Observation

The table shown in figure 11 illustrates the various dimensions and trade-offs that we identified in the course of the 'following' and 'static observation' experiments. The examples are simple in that no explicit communication is permitted between the model and imitator, in fact the sensory information is basically the perceived brightness of a light bulb. We are examining social learning in the setting of no feedback or direct teaching from the model. The results indicate that there is a clear trade-off between positional accuracy obtained from static observation and the advantages of direct perception-action coupling available from following. This is perhaps unsurprising, given that static observation is certainly the more complex and engineered method in these and most other robotics experiments, however the issue is more simply that 'following' lacks accuracy. This appears to be due primarily to reactive impersistence and model interference problems. Impersistence may be soluble with more complex algorithms [20], however model interference may be unavoidable where a close shared context is employed.

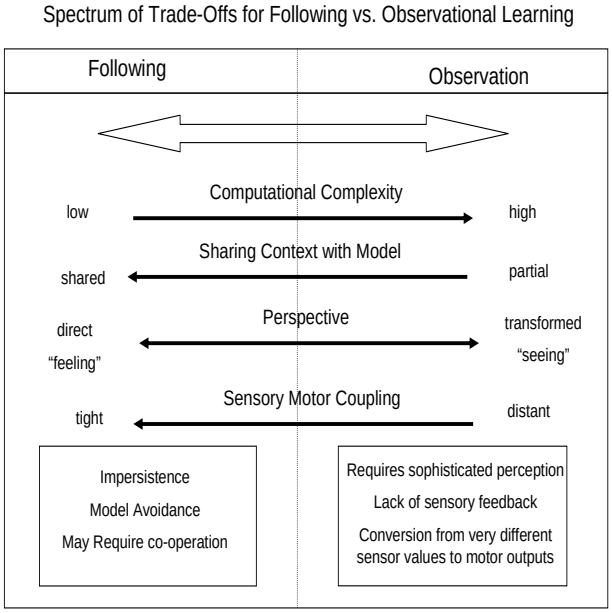


Fig. 11. *The table summarises the key aspects revealed by the experiments with extremes of each aspect shown (see text). Comparative costs are shown in the boxes. Mixed approaches might allow the balance of these costs and benefits.*

The relative simplicity of the following paradigm also hides some key advantages, in that the robot is able to directly map its perceptions against its motor actions. It is thus able to learn much about the environment directly and relatively cheaply. However to achieve positional accuracy, more complex observational sensorimotor capabilities or algorithms are required, but observation alone without "experiencing" the context may be insufficient to correctly assess the physical complexities of the environment. There might be an argument for suggesting that observation can be most effective when combined with a following episode, i.e. observation can fine-tune already stored movement patterns. This may also be the case in biology. For example, when imitating, humans and some animals appear to use both mechanisms as necessary and there is evidence from human babies that in order to learn the perception-action couplings a period of following behaviour may be necessary, where the infant moves its arms, face and body in response to its mother's actions ('motor babbling') while reducing perceptual errors [22]. The issue of model interference with the demonstrator whilst following also suggests that a pure following strategy may be impractical without some form of co-operation from the model.

Animals use both seeing and feeling in imitation and there may be an appropriate time to see (observe) as opposed to feel (follow) in social learning. A mixed approach may be valuable, this approach corresponding to intermediate positions or strategy switching in the spectrum table shown. One could imagine for example cases where the observation is less static e.g. several follow-observe-follow cycles, or where a series of static observations are made prior to each episode of following behaviour.

B. Dynamic Observation

We stated above that a 'following' behaviour, although limited in its imitative accuracy especially due to the impersistence problem and model avoidance, has the major advantage of computational simplicity, and the added value of direct interaction with the environment through proprioceptive polling of actuators whilst moving. We do not suggest that this opportunity to 'feel' the environment is exclusive to a following strategy however it does have the straightforward merits described above. It is also true that both a follower and a static observer are necessarily out of phase with the model and for this reason it seems that the follower's sensory cues may not be more appropriate than an observer's, however work by [11] showed that the these cues are dependent on the distance between a follower and the model and within a critical distance range the follower's sensory cues become very relevant. We describe here an initial attempt to provide a movement mechanism to an observer in order to combine the advantages of observational accuracy with the feedback obtained from actively exploring the environment. Clearly a simple and modular solution to this task would be to keep to the 'extreme' behaviours and simply apply each strategy in turn e.g. follow-statically observe-follow. One of the aims of this research has been to explore the challenges faced in combining these strategies whilst retaining the positive aspects of both. The experiments themselves are clearly limited as we are constrained both by the sensor embodiment of the robot and its internal control system, but we believe valuable lessons still emerge.

Lessons for Imitators Dynamically Observing from a Fixed Location. Our first experiment showed that dynamic observation with rotation was successful in that it allowed the model to pass out of view of the imitator and be reacquired. It was superior to static observation alone in this respect and it appeared that the benefit of tracking accuracy could be balanced against the cost of rotation frequency and rotational movement based on a turn threshold. Thus to retain observational accuracy, rotational movement should be limited so that odometry errors are minimised in their effect on the geometric transforms required to replay the imitation. Thresholds near the periphery of vision balance these factors. In robotics the issue of errors from odometry drift is clearly not new, however the literature on robotic observational imitation seem rarely to cite it as being a problem for a moving imitator.

Lessons for Imitators that Observe and Move. Our second experiment showed that with this particular robot, dynamic observation with movement of the imitator was extremely difficult and failed to replicate with reasonable accuracy the model's path. Theoretical analysis suggested that the 'inattentiveness' problem may be soluble for a dynamic observational imitator where the movement value is set to unity. This region in our spectrum corresponds with the methods of other research [20] where a single solution to this issue is considered. However the need to make additional movements over and above those required to track the model means that the movement value can never be unity for an embodiment where the sensor orientation is completely fixed for a fixed body orientation. Otherwise the imitation is likely to be poor unless compensation can be made for not perceiving whilst moving.

Possible Solutions. A solution to this might be *independent sensing and actuator mechanisms*. We envisage that such a system would additionally employ independent computation facilities for both mechanisms to allow continuous and parallel calculation of model position. Thus appropriate movement vectors could be sent to the actuators reducing unnecessary movements and the associated additional odometry drift. The sequential nature of the move-sense cycle on our robot may mean that accurate dynamic observation is very difficult, however other control systems employing a *parallel cycle* may provide solutions. There may also be simpler alternatives, for example the model may *repeat the pattern* and the imitator might manage to fill the gaps caused by earlier inattentiveness, or the model might simply *wait for the imitator*. These latter solutions imply that the model is interacting with the imitator, effectively *directly teaching* the imitator. This aspect of observational learning is part of our current research [23] in this area.

Even in our own human experience it often appears much harder to both partially carry out an imitative behaviour whilst simultaneously observing the model before the model has finished its actions. Humans and some other animals in fact may have partially obviated this issue by evolving alternative mechanisms. In this respect the recent neurological evidence of 'mirror neurons' in primates and humans [24] and their role in action perception suggests they may be exploited in static observational learning with the imitator experiencing perhaps as good a correlation to its own behavioural patterns whilst statically observing as when attemping to match movements directly. In this respect study into the generalisation of the learning and how a mechanism of *predictive observation* may be used forms another promising area for future research.

ACKNOWLEDGMENT

The work described in this paper was partially conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion") and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

APPENDIX I
COMPARISON OF METHODS FOR OBSERVING ANGLE AND DISTANCES TO A LIGHT SOURCE

Learning to Measure Angles. The robot is first trained to accurately compute the angle of the light from the centre of the imitator. A number of methods were evaluated including using a *light compass* [19] shown in figure 12, or computing the angle by using *vector summation* of the inputs to each of the light sensors [18] (described in the section on 'Following' in the main text).

However both of these methods were not accurate and suffered from incorrect readings especially when none of the robot's sensors were directly facing the light. A new method, which we call *environmental sampling*, was grounded

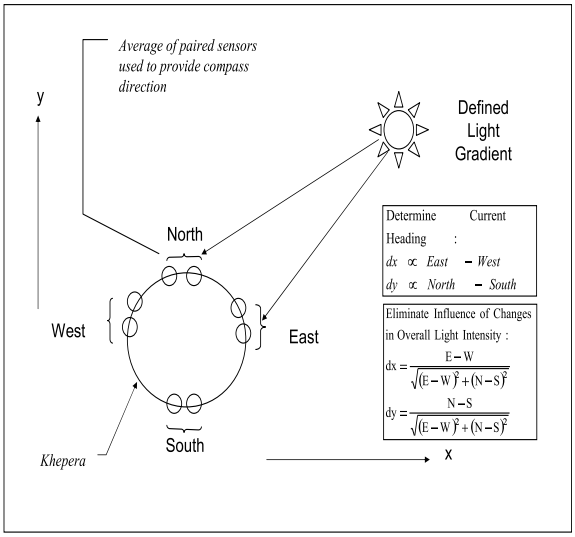


Fig. 12. *Differential Light Compass [19]. The angle of the robots sensor arrays are used to differentiate the light direction into cartesian coordinates. For the Khepera robot the average of each of the sensors nearest the appropriate compass direction is used. The arc Tangent of the resulting values for dx and dy are used to yield the angle of the light in relation to the robot.*

in sensory experience and is to some extent nearer to a biological solution: the robot is allowed to learn about light angles simply by observing them. As the Khepera is a circular robot it rotates in the presence of the stationary model. It detects when the rotation is complete by polling its wheel encoders and stopping when the appropriate value has been exceeded (during the rotation it reads its light sensors every 200ms. A robot rotating at 8mm/s would typically poll it sensors 65 times). As the speed of the rotation is constant the time interval between readings can thus be converted to an angle. Each of the sensor readings are then normalised. This has two effects, firstly that of making distant readings of angle equivalent to closer readings, and secondly allowing these values to be loaded directly as weights into a neural network (a counter-propagation network [25]). This is a fully connected feed-forward three layer network. The first layer takes the normalised input of the 8 light sensors, the number of middle layer neurons is set to the number of times the robot was able to poll its sensors and the final layer used to output the conversion of these values to angles. Using this technique has a number of advantages. Firstly that the network can be built as the environment is observed, secondly there are no additional training steps i.e. there is no further training of the neural network, thirdly the size of the network is directly related to the internal rotation speed, sensor modality and sensor polling time of this particular robot and finally that the method is partially resilient to sensor failure. A comparison of *environmental sampling* and the *differential light compass* is shown in figure 14.

There are some biological observations which may show similar (though not equivalent) mechanisms in animals. For example young bees appear to record the image of their hive from many angles and positions around it: they fly in and out of the hive varying their circular flight path each time [26].

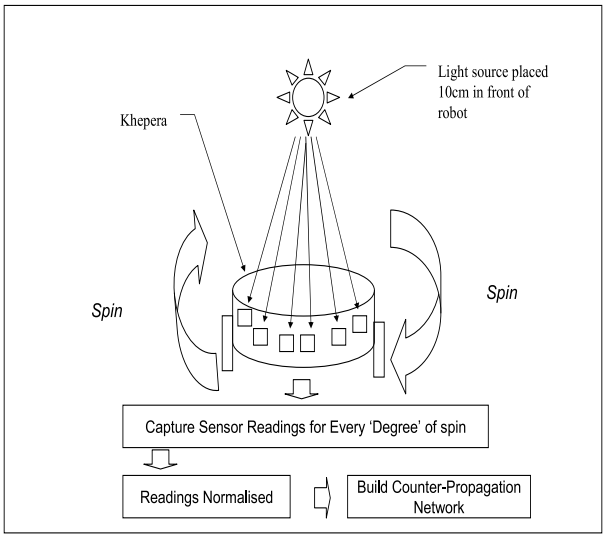


Fig. 13. *Environmental Sampling. As the robot turns it polls it sensors every 200ms. The history of sensor values are then used to form a trained counter-propagation network [25] which subsequently acts as a generalising lookup table.*

Learning to Measure Distance. For distance measurements various mechanisms were also assessed. A first approach was to use *triangulation*, exploiting the fact that accurate angle measurement was now possible. The approach measured the light angle from the model, moved the imitator a fixed distance and then read the new angle. This allows the computation of the original distance using the two angles and the travelled distance. However this mechanism was unreliable for two reasons, firstly that, over small movement distances (which minimised errors in the odometry readings from the wheel encoders), the derived angle would be small and tiny errors in the angle measurement would result in an amplified error in the distance computation, secondly if the model was moving, the measurements/movement combination of the imitator could never be fast enough to resolve the position of the model accurately. An alternative method based on *environmental sampling* was used for the angle computation, the light sensors being summed as

vectors as the robot turned. As follows:

$$(\theta, r) = \Sigma_{i=1}^{n}(LightSensorAngle(i), LightSensorReading(i)),$$

where the summation is carried out in polar coordinates, with the resultant magnitude r held in a lookup table indexed by robot turn angle and distance from the model.

This exploited the fact that sensors directly facing the light would have a larger effect on the vector magnitude than those further away. The robot was trained by rotating at increasing 1cm distances from the light source. The vector magnitude was then held in a lookup table indexed by angle and distance. Using this method gave a reasonable distance accuracy up to about 25 cm from the robot at an angle between

approximately 30° to 150° in front of the robot. However, outside these parameters the distance accuracy was very poor. Following these procedures the robot can compute both angle and distance without further training.

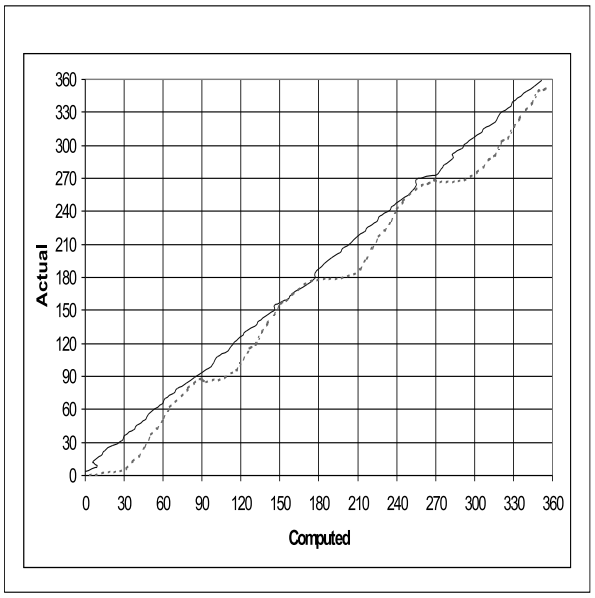


Fig. 14. *Comparison of Environmental Sampling and Differential Light Compass. The y-axis shows the actual angle of the light source with respect to the centre of the Khepera robot. The x-axis shows the results of computing the angle. The dotted line shows the results from using the differential light compass. The continuous line shows the results from using the environmental sampling method described in the text.*

REFERENCES

[1] K. Dautenhahn, "Trying to imitate – a step towards releasing robots from social isolation," in *Proc. From Perception to Action Conference, Lausanne, Switzerland*, P. Gaussier and J.-D. Nicoud, Eds. IEEE Computer Society Press, 1994, pp. 290–301.

[2] R. W. Byrne, *The Thinking Ape: Evolutionary Origins of Intelligence*. Oxford University Press, 1995.

[3] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in Cognitive Sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[4] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Towards robot cultures? - learning to imitate in a robotic arm test-bed with dissimilar embodied agents," *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, vol. 5, no. 1, pp. 3–44, 2004.

[5] C. L. Nehaniv and K. Dautenhahn, "The Correspondence Problem," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 41–61.

[6] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, "Teaching robots by moulding behavior and scaffolding the environment," University of Hertfordshire, Hatfield, Hertfordshire, UK, Technical Report 432, September 2005.

[7] T. R. Zentall, "Imitation in animals: Evidence, function and mechanisms," *Cybernetics and Systems*, vol. 32, no. 1-2, pp. 53–96, 2001.

[8] N. E. Miller and J. Dollard, *Social Learning and Imitation*. Yale University Press, 1941.

[9] B. G. Galef and C. M. Heyes, Eds., *Social Learning in Animals: The Roots of Culture*. Academic Press, 1996.

[10] J. Noble and P. M. Todd, "Imitation or something simpler? modeling simple mechanisms for social information processing," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. L. Nehaniv, Eds. MIT Press, 2002, pp. 423–439.

[11] A. Billard and K. Dautenhahn, "Grounding communication in situated, social robots," in *Proc. TIMR*, 1997.

[12] G. Hayes and J. Demiris, "A robot controller using learning by imitation," in *Proc. Int. Symp. Intelligent Robotic Systems, Grenoble*, 1994, pp. 198–204.

[13] P. Gaussier, S. Moga, J. P. Banquet, and M. Quoy, "From perception-action loops to imitation processes: A bottom-up approach of learning by imitation," in *Socially Intelligent Agents*. AAAI Press, Technical report FS-97-02, 1997, pp. 49–54.

[14] Y. Kuniyoshi, M. Inaba, and H. Inoue, "Learning by watching: Extracting reusable task knowledge from visual observations of human performance," *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 799–822, November 1994.

[15] D. C. Bentivegna and C. G. Atkeson, "Learning how to behave from observing others," in *Proc. Simulation of Adaptive Behaviour -Workshop on Motor Control in Humans and Robots: on the interplay of real brains and artificial devices, Edinburgh, UK, August, 2002*, 2002.

[16] B. Scassellati, "Imitation and mechanisms of joint attention: A developmental structure for building social skills," in *Computation for Metaphors, Analogy and Agents*, C. L. Nehaniv, Ed. Springer Lecture Notes in Artificial Intelligence, Volume 1562, 1999, pp. 176–195.

[17] C. L. Nehaniv and K. Dautenhahn, "Like me? - measures of correspondence and imitation," *Cybernetics and Systems*, vol. 32, no. 1-2, pp. 11–51, 2001.

[18] R. C. Arkin, *Behavior-based robotics*. Cambridge, Mass., London, England: MIT Press, 1998.

[19] U. Nehmzow, "Animal and robot navigation," in *The Biology and Technology of Intelligent Autonomous Agents*, L. Steels, Ed. Springer Verlag, 1993.

[20] J. Wit, "Vector pursuit path tracking for autonomous vehicles," Ph.D. dissertation, University of Florida, 2000.

[21] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, "An experimental comparison of imitation paradigms used in social robotics," in *Proc. IEEE Robot and Human Interactive Communication (ROMAN '04)*. IEEE Press, September 2004, pp. 691–696.

[22] J. Nadel, A. Revel, P. Andry, and P. Gaussier, "Toward communication:first imitations in infants, low-functioning children with autism and robots," *Interaction Studies 5(1)*, vol. 5, pp. 45–74, 2004.

[23] J. Saunders, C. L. Nehaniv, and K. Dautenhahn, "Teaching robots by moulding behavior and scaffolding the environment," 2005, (Submitted) also available as University of Hertfordshire Computer Science technical report number 432, September 2005.

[24] V. Gallese, L. Fadiga, L. Fogassi, and G. Rizzolatti, "Action recognition in the premotor cortex," *Brain*, vol. 119, pp. 593–609, 1996.

[25] R. Hecht-Nielson, "Applications of counterpropagation networks," *Neural Networks*, vol. 1, pp. 131–139, 1988.

[26] R. R. Murphy, *Introduction to AI Robotics*. The MIT Press, Cambridge, 2000, p.76.

Joe Saunders received the M.Sc. degree in artificial intelligence from the Department of Computer Science, University of Hertfordshire, Hertfordshire, U.K., in 2003. He is currently pursuing a Ph.D. degree at the Adaptive Systems Research Group, Computer Science Department, University of Hertfordshire, Hertfordshire, U.K. He previously directed large software development projects for a number of international investment banks. His research interests include social robotics and imitation in artificial systems.

Chrystopher L. Nehaniv received his Ph.D. degree in Mathematics from the University of California at Berkeley in 1992, and is currently Professor of Mathematical and Evolutionary Computer Sciences at the University of Hertfordshire, in Hatfield, England. He is a founding member of the IEEE Working Group on Artificial Life and Complex Adaptive Systems, and serves as Associate Editor for the journals *BioSystems: Journal of Biological and Information Processing Sciences* and *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, as well as Director of the U.K. Engineering and Physical Science Research Council and Interactive Network on Evolvability in Biological and Software Systems.

Kerstin Dautenhahn received her Ph.D. degree from the Biological Cybernetics Department of the University of Bielefeld, Bielefeld, Germany, in 1993. She is Professor of Artificial Intelligence in the School of Computer Science and coordinator of the Adaptive Systems Research Group at the University of Hertfordshire in England. She has published more than 100 research articles on social robotics, robot learning, human-robot interaction and assistive technology. Prof. Dautenhahn edited several books and frequently organises international research workshops and conferences. She is involved in several European robotics projects and is Editor in Chief of the journal *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems.*

Teaching Robots by Moulding Behavior and Scaffolding the Environment

Joe Saunders*
Adaptive Systems Research Group
University of Hertfordshire
Hatfield, AL10 9AB
United Kingdom
J.2.Saunders@herts.ac.uk

Chrystopher L. Nehaniv
Adaptive Systems Research Group
University of Hertfordshire
Hatfield, AL10 9AB
United Kingdom
C.L.Nehaniv@herts.ac.uk

Kerstin Dautenhahn
Adaptive Systems Research Group
University of Hertfordshire
Hatfield, AL10 9AB
United Kingdom
K.Dautenhahn@herts.ac.uk

ABSTRACT

Programming robots to carry out useful tasks is both a complex and non-trivial exercise. A simple and intuitive method to allow humans to train and shape robot behaviour is clearly a key goal in making this task easier. This paper describes an approach to this problem based on studies of social animals where two teaching strategies are applied to allow a human teacher to train a robot by *moulding* its actions within a carefully *scaffolded* environment. Within these enviroments sets of competences can be built by building state/action memory maps of the robot's interaction within that environment. These memory maps are then polled using a k-nearest neighbour based algorithm to provide a generalised competence. We take a novel approach in building the memory models by allowing the human teacher to construct them in a hierarchical manner. This mechanism allows a human trainer to build and extend an action-selection mechanism into which new skills can be added to the robot's repertoire of existing competencies. These techniques are implemented on physical Khepera miniature robots and validated on a variety of tasks.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: [robotics]; I.2.6 [**Artificial Intelligence**]: [learning]; I.2.m [**Artificial Intelligence**]: [miscellaneous - imitation, programming by demonstration]

General Terms

performance

Keywords

Social Robotics, Imitation, Teaching, Memory-based learning, Scaffolding, Zone of Proximal Development

*corresponding author

1. INTRODUCTION

Imagine a scenario where your brand new domestic robot has just been delivered. The factory have pre-programmed it to carry out some useful tasks around the home e.g. collecting cutlery, cups and plates and placing them in a dishwasher or tidying up by picking up clothes left on the floor and placing them in a washing basket. You unpack the robot, press the "on" button, and the robot efficiently carries out these tasks whilst being safe to both you and itself. Later however you find that although the robot performs to the manufacturer's specifications there are some tasks which it does not carry out. It fails to tidy up the children's toys into the toy cupboard or it fails to recognise that a particular and expensive glass should not be placed in the dishwasher. After a call to the manufacturer you discover that there is another button on the robot marked "learn". When this button is pressed the robot can be taught additional skills. This paper presents research on how such a teaching mechanism might be implemented.

In section 2 we suggest that it is the social dimension of behaviour that holds the key to making robots behave more intelligently [6], an approach inspired from studies of social animals (e.g. apes) and the 'social intelligence hypothesis' [4], which proposes that intelligent behaviour in primates has its origins in dealing with complex social dynamics. We discuss how the social aspects of teaching, learning and imitation are used by some social animals to expand their repertoire of skills. From this work we extract the developmental concepts of "scaffolding" and "putting-through"/"moulding" as mechanisms which may prove useful for robot teaching. Section 3 discusses related work where observation, imitation and direct teaching are used. We conclude this review by outlining the computational techniques that we will use in creating a novel learning architecture which will allow new robot skills to be taught whilst retaining or improving existing skills. Section 4 details the realisation of this architecture on physical Khepera miniature robots. Section 5 gives the experimental validation of the work by showing examples of how behaviours can be created in a robot and how additional skills can then be added to an existing robot skill repertoire. Finally we discuss some of our findings and the possible directions for further research in this area.

2. TEACHING AND IMITATION IN ANIMALS

Moore [13] proposes a six-step hypotheses for the evolution of imitation in nature The process starts with Thorndikian conditioning where existing motor actions are associated and reinforced based on particular environmental conditions. This step is later enhanced by operant (or Skinner) conditioning where novel motor responses are formed based on combinations of existing actions. The next evolutionary step is an implicit reinforcement cycle leading to "skills" where the animal is able to perfect the novel act. The fourth stage introduces the teacher. The teacher essentially guides the pupil by physically "moulding" or "putting-through" the actions of the pupil given particular environmental stimuli. This can be considered as self-imitation by the animal as it repeats the actions that it has experienced. Visual imitation of others is the next evolutionary stage. In this case the animal now only has to see an act to be able to repeat it. The final process is called cross-modal imitation where an animal is able to match features of its body with corresponding features of another animal. For example, human babies touch parts of the faces of their parents and can then locate the same features on their own face. In figure 1 we summarise and segment these stages into prime, taught and imitative sections.

The study presented in this paper bases its mechanisms for robot teaching on the *self-imitation* stage. However each of the earlier stages are also used. For example, we simplify the actions available from the *Thorndikian* stage by considering them to be part of the robot's existing repertoire of motor skills. This existing set of skills over and above basic motor actions are called "primitives". Explicit combinations of primitives can be specified by the teacher. We call these "sequences" but they are equivalent to novel sets of responses available at the *operant conditioning* stage. Skill learning is the essential building block upon which the teacher's directions are built. The *skill reinforcement* stage will therefore form the association between sensed stimuli and action. The fourth *self-imitation* stage is based on moulding or putting-through. This is where the training example is provided by the teacher by *putting* the robot *through* the range of actions required. Our previous work [21] considered aspects of observational/imitative learning at the *imitation* stage.

Evidence for teaching in the animal kingdom comes mainly from studies of primates [4]. However there is also evidence from carnivores including domestic cats, tigers, cheetahs, otters, dolphins, orca whales and some bird species [24]. An example from cheetahs is where the mother rather than killing prey will capture and release the live prey to the cheetah cubs when they are about 3 months old. The behaviour is also selective, only prey species which the cubs are likely to catch are released. It appears that the cubs' experience results in faster learning and a more skilled performance. This was tested with domestic cats whose kittens were brought live mice by their mothers at an early age. By 6 months old the kittens showed superior skills to a test group who had not been exposed to the mice [23].

Compelling evidence of intentional teaching comes from studies of primate behaviour. Fouts *et al.* report on the chimpanzees Washoe and Loulis, Loulis being the adopted infant

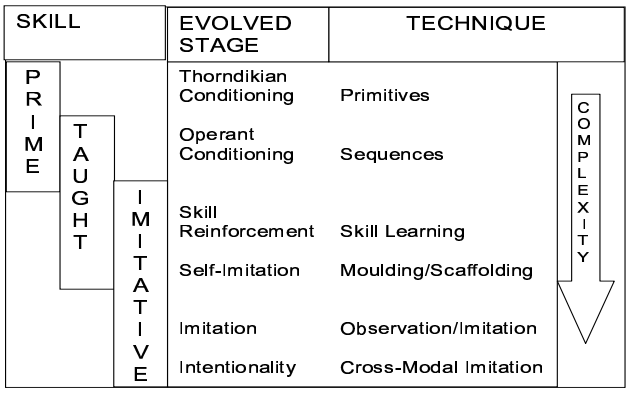


Figure 1: Proposed evolutionary stages with techniques required to implement them. This paper deals with the *taught* skill set.

chimp of the mother Washoe. Washoe had been previously been taught American Sign Language (ASL) however the human carers made no attempt to teach Loulis ASL and did not use ASL in Loulis' presence. However Washoe succeeded in teaching Loulis ASL both by demonstration and by *moulding* of Loulis' hands [18]. Moulding had also been used by the human carers to originally teach Washoe.

Scaffolding is where a physical situation is artificially modified, typically by the mother, to make it much easier for her child to complete the task when the child is at a developmental stage where it could not perform the appropriate acts or sequence its actions correctly. Scaffolding of tasks together with observational learning and moulding have been observed in wild chimpanzees [4]. Cracking nuts with a hammerstone is an especially difficult task for a chimpanzee to learn, taking up to 14 years to perfect in some cases. A number of observations have been recorded where the mother will clean the anvil, reposition the the nut or re-orient the hammerstone to favourable orientations for the infant. Scaffolding is also a familiar concept in human development and is emphasised in Vygotysky's idea of the "zone of proximal development" in his theory of the child in society [27]. Teaching and social interaction allow higher competence levels to be achieved through staged learning and building upon existing skills.

We take inspiration from these examples in social animals to study how *moulding/putting though* and *scaffolding* can be used to good effect in teaching robots new skills and allow existing skills to be modified.

3. RELATED WORK

Even with explicit programming robot control is hard due sensor noise, the non-deterministic state of the environment, the inability to ensure that the robots actions are deterministic and the need for real-time responses. There are generally a number of problems which need to be solved:

i) *how can the human teach the robot?* - what mechanisms can be used to make the robot match the intentions of the teacher? how can the robot learn when the task is complete?

ii) *what techniques can the robot use to learn?* - how can the machine generalise and execute the new task?

iii) *how can the robot incorporate the new experiences into its existing competencies?* - what sort of structure is necessary to ensure that new tasks can co-exist with existing tasks?

iv) *how can it select the right action at the right time?* - given a learned set of competencies which one is appropriate?

Approaches include topics such as programming by demonstration, imitation learning, learning from observation and robot shaping. Typically the observational and imitative approaches attempt to match the behaviour of the demonstrator and so construct an appropriate control policy. Schaal et al. [22] provide an overview where approaches to the problem are classified as follows:

i) *direct policy learning* - where supervised learning is used to learn a control policy directly.

ii) *learning policies from demonstrated trajectories* - this assumes that the task goal is known and uses sample trajectories to learn a control policy

iii) *model based policy learning* - where a predictive model of the control problem is constructed.

All of these approaches face two difficult problems. Firstly, that by observation alone the internal proprioceptive feedback that the teacher experiences cannot be directly experienced by the pupil [20] and secondly, there may be a mismatch between the external and internal sensorimotor spaces of the teacher and pupil - the correspondence problem [14].

In the *direct policy approach* these issues can be avoided by having the pupil experience the same set of actions and sensory states as the teacher with the correspondence problem solved by ensuring that both teacher and pupil have a similar embodiment. This approach has been used by a number of groups including Billard & Dautenhahn [3] and Hayes & Demiris [9]. In both cases a student robot followed a teacher robot and learned to associate imitated actions against perceived environmental state. Saunders *et al.* [20] however have demonstrated that there can be limitations in this approach due to reactive impersistence and teacher interference when using a pure following approach.

In recent work by Nicolescu *et al.* [15] a mobile robot tracks a teacher's movements matching predicted postconditions against the robot's current proprioceptive state. It then builds a hierarchical behaviour-based network based on "Strips" [17] style production rules. This work attempts to provide a natural interface between robot and teacher whilst automatically constructing an appropriate action-selection framework for the robot.

Another way to allow a robot to experience the appropriate sensory state is by allowing the teacher to manipulate the robot directly via a form of tele-operation and record the sensory state of the robot. Although not using a robot this method is closely related to Sammut's [19] "learning-to-fly" application where recordings of control parameters in a flight simulator flown by a number of human subjects were analysed using Quinlan's C4.5 induction algorithm [16]. The algorithm extracted a set of "if-then" control rules. Van Lent [26] also used this approach but provided a user interface which could be marked with goal transition information. This allowed an action-selection architecture to be constructed using "Strips" [17] style production rules. However, in both of these research areas the full "state" of the system (both internal and external) is available to the trainer. This may not be the case when teaching robots.

A long line of research into teaching service robots by observing humans has also been carried out by Dillman [7] where after observation production rules are generated to produce grammatical formalisms held in a knowledge database of actions.

Dorigo and Colombetti [8] use decomposition of tasks by a trainer to "shape" robot behaviour. We take a similar approach however we do not use either evolutionary or reinforcement learning techniques to create or modify robot behaviour.

Learning policies from demonstrated trajectories seems to be appropriate when the goals of the task are known and the task itself is self contained. For example when learning to duplicate human movements [11] or play tennis strokes [10] the goal of the task is already known or programmed into the learning mechanism. It is made explicit by the programmer for the specific (although mechanically complex) task to be solved. It is difficult to see how a new task could be incorporated into the existing learned policy without further explicit programming.

Bentivenga *et al.* [2] use *model based policy learning* to construct a learning framework using a memory-based approach. A humanoid robot learns to play games of "marble maze" and "air hockey" by recording exteroceptive data (ball angle/velocity, board tilt angles) and primitive type (roll ball away from corner, roll ball off wall) from a human demonstrator. The robot is able to select the appropriate primitive by analysing a memory model to find the nearest example to the current state. Parameters for the primitive are constructed using locally weighted regression on points nearest the selected primitive. This technique is also related to loose-perceptual matching methods described in [1].

Memory based learning approaches have a number of technical advantages. Firstly, complex functions can be learned by focusing on sets of less complex local approximations. Secondly, the local approximation for the target query (based on the current sensory state) is based on the training data at the time of the query and not on a pre-built function approximation. This means that additional training instances can be added immediately without the need to rebuild a target function (which would be the case for an inductive or neural network approach).

It is noticeable that many of the example applications described above have the ability to learn complex tasks based

on some form of observation (where observation can be both direct and from post-processing of sensory data). However with the exception of [26, 7, 15] there are few mechanisms which allow another task to be both learned and included into the repertoire of previously learned functions. Our approach is to provide an interface which will both *learn a particular task* and have the ability to *add this task to an existing control mechanism*. This requires a number of steps.

i) a policy needs to be learned based on the sensory state of robot itself. The correspondence between the human teacher and robot also needs to be solved - both of these points we address by the simple process of *moulding* the robot by teleoperation.

ii) the robot needs to be aware of when tasks have a specific goal - we make this an explicit part of the training sequence.

iii) learning must be carried out in real-time and be subsequently modified or enhanced with additional learning experiences - this is made possible by using memory based learning methods.

iv) the new learning experience should not corrupt other previously learned experiences - we allow the construction of a hierarchy of memory models to provide this.

One of the key points in addressing many of the issues described is that a teacher constructs an appropriate learning environment for the robot. We do this while exploiting and extending some of the techniques already used by the practitioners above in a new framework.

4. FRAMEWORK

For this study we have used physical Khepera miniature robots (see figure 2) on a desk in a typical busy academic environment. Khepera's are 5cm diameter non-holonomic robots equipped with eight IR sensors placed at intervals around the base, an arm/gripper and a K213 linear vision system. The IR sensors are capable of detecting both ambient light and short range (10cm) obstacles. The arm/gripper arrangement can detect when an obstacle is within the gripper and also the electrical resistivity of the object grasped. The K213 vision system provides a one dimensional line of 64 grey scale values subtending an angle of 36° from the front of the robot. Commands to control the robot can be sent from a remote PC either via a radio signal or from a directly connected serial cable.

The learning environment we choose is based around the capabilities of the Khepera. To provide a reasonably complex learning environment the Khepera is placed in a walled "room" with various objects of different conductivity and some bar-coded containers.

4.1 Learning Mechanism

We use a memory based "lazy" learning method [12] to allow the robot to learn tasks. This is a simple k-nearest neighbour (kNN) approach where the value of each feature in the robot's state vector (see *Scaffolding* below) is regarded as a point in n-dimensional space, where n is the number of features in the state vector (see table 1). For each chosen task we collect a set of training examples (as described in *Moulding* below) together with their target primitives, each primitive being chosen by the human trainer when moulding the robot's actions. When the task is executed the robot continually computes its current state vector. It then computes the distance from the current state to each of the training examples. The distance between the state vector and the training example being the sum of the distances between the features in each, as follows:

$$distance(X, S) = \sum_{i=1}^{n} W_i \left| \frac{x_i - s_i}{max_i - min_i} \right|$$

Where X is an instance of the training examples and S an instance of the robot's current sensory state. W is a non-negative vector of real numbers used to weight each of the dimensions. This weighting is discussed in the *scaffolding* section below. Setting k to 1 will result in the nearest point in the training examples being used and yield a single primitive as its target function. Where k is greater than 1 the algorithm will yield a set of primitives. We choose the most common primitive from the set as the target function. Note that this method will always result in a primitive being chosen. In work to date the k value has always been set to 1. We make use of the Tilburg University Memory Based Learner [5] to provide this functionality. This has the advantage of providing a very efficient tree-based coding structure for the training examples so as to speed up performance.

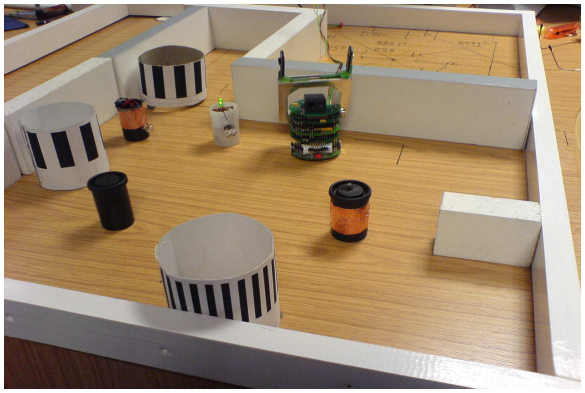


Figure 2: A typical environment showing a Khepera with vision sensor and gripper, objects with different electrical resistance and bar-coded containers.

4.2 Moulding

The concepts of scaffolding and moulding can play an important part in animal learning. They support a form of self-imitation that may be the natural precursor to more complex forms of imitative learning. In our framework we use the idea of moulding or putting-through directly. The human has the ability to control the robot by remotely moving it through a set of pre-defined basic primitives. This set of primitives are basic actions available to the robot (see table 2). The human teacher has no access to the internal state of the robot. By manipulating the robot in this manner we also avoid both the problem of observation by the robot of the human actions and of the correspondence problem between the robot and human. During the robot

Table 1: State Vector Used in experiments

State	Description
Repulsive Force	Vector of IR sensors
Repulsive Angle	Angle of IR Vector
Light Distance	Distance to light
Light Angle	Angle to light
Bars Seen	Number of bars seen by K213
Bar Size	Average bar size seen by K213
Bar.Std.Dev.	Std. Deviation of bar size
Arm Up/Down	Whether the arm is up or down
Gripper Open/Closed	If gripper is open or closed
Arm Up/Down	If arm is up or down
Object in Gripper	If object is in the gripper
Resistivity	Resistivity of object

Table 2: Pre-defined Primitives.

Primitive	Description
Move Forwards	Move Forward 1cm or continuously
Move Backwards	Move Backwards 1cm or continuously
Turn Right	Turn Right by 5° or continuously
Turn Left	Left Left by 5° or continuously
Raise Arm	Raise Arm, if not already raised
Lower Arm	Lower Arm, if not already lowered
Open Gripper	Open gripper if not already open
Close Gripper	Close gripper if not already closed

moulding process a snapshot of the robots proprioceptive and exterioceptive state (see table 1) is recorded together with the directed primitive on each human command to the robot. For each human defined task we can therefore build a memory model of state/primitive combinations.

4.3 Scaffolding

All of the states perceived by the robot are recorded in the state vector however different attributes of this vector are relevant to different tasks. For example, to avoid obstacles the attributes of the IR sensors are of more importance than the position of the gripper, whereas to track an object the perceived orientation of the object is more relevant than the values of the IR sensors. Here we capture a pre-defined set of states some of which are numeric summaries pertinent to the expected applications and realisable by the sensor arrangement of the robot (see table 1).

We use two mechanisms to ensure that the appropriate attributes are chosen. The first is a technical solution originally used in Quinlan's C4.5 Induction algorithm [16]. This is based on computing *information gain* to measure how well a given attribute separates the set of recorded state vectors according to the target primitive. This is defined as follows:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{\mid S_v \mid}{\mid S \mid} Entropy(S_v)$$

where S is the collection of training examples, *Entropy(x)* is a function returning the entropy of x in bits, *Values(A)* is the set of all possible values for a particular state attribute A and S_v is the subset of S for which attribute A has value v. Further explanations of this metric can be found in [16, 12]. The information gain measurement allows particular attributes in the state vector to have greater relevance by using it to weight the appropriate dimensional axes in the kNN algorithm (by setting W_i above). This has the effect of either lengthening or shortening the axes in Euclidean space thus reducing the impact of irrelevant state attributes.

The second mechanism for attribute selection is the human trainer. It is assumed that the trainer already understands the task (from an external viewpoint) that the robot must carry out and therefore is able to construct the training environment appropriately so as to ensure that irrelevant features are removed. This idea allows the technical selection of relevant state features to be enhanced as the other features will now tend to have constant values and therefore a low information gain.

As an example consider training the robot to perform a "wall following" behaviour. The teacher might remove extraneous objects from the training area such as the bar-coded containers. By moving the robot through a number of wall following experiences the set of sensory states recorded will then be primarily based on the IR sensors (which resolve to the repulsive vector/angle attributes). These attributes will then be automatically selected by the extended kNN algorithm based on their higher information gain. As discussed in section 2 above this process of scaffolding or creating favourable conditions for learning would seem a quite natural phenomenon in social animals and is of course fundamental to all forms of human teaching.

4.4 Learning New Tasks

We are now in a position to define the mechanisms available to the human trainer.

The robot can be in one of three modes. The first is *execution mode*, which is its normal mode of operation where its current behaviour is executed. Alternatively the robot can be in *training mode* where the human trainer can mould, scaffold and create new activities for the robot to eventually use in execution mode. An *intermediate mode* is where the trainer can execute one of the set of available competencies. For example by selecting the primitive "Move Forward" in this mode the robot will execute the move forward primitive. This is useful for placing the robot in an appropriate state prior to training.

In "learning" mode the robot can learn new competences at one of three training levels determined by the trainer: *sequence*, *task* and *behaviour*. All three training levels are started by pressing a "start learning" button and terminated by pressing a "stop learning" button. For each new competence (either a behaviour, task or sequence) the trainer provides an appropriate label. When training is complete the label is added to the set of actions available to the trainer and thus can be used immediately for further training sessions. Existing labeled actions can also be modified with additional training episodes as required. In training mode the trainer has the option to execute the selected competence so that the results of the robot's actions can be assessed immediately.

The first competence level is the *sequence*. This is where the robot can be directed through a given sequence of primitives

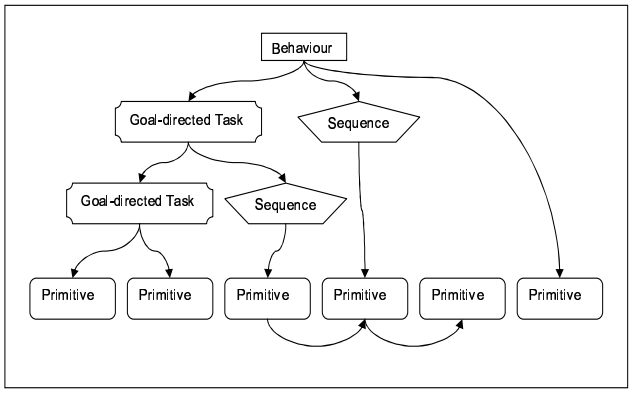


Figure 3: An example of a trained hierarchy of primitives, primitive sequences, learned goal-directed tasks and the final behaviour.

which it records without reference to its state. An example of a sequence might be to lower the arm and close the gripper. This could, for example, be labelled as the 'grab' sequence. The grab sequence would then become part of the available set of competences available for the trainer to use. These new sequences could also then be used in combination with other primitives and other sequences to create further sequences. Note that sequences are entirely deterministic. When requested to perform a sequence the robot will simply execute the recorded list of competences taught by the trainer sequentially. It will make no reference to the environmental state.

The second level for learning is called a *goal-directed task* or simply a *task*. This differs from a sequence in that during training the actions taken by the robot will depend on the environmental state at that time. The trainer now has the opportunity to select not only basic primitives, but sequences and other goal-directed tasks. The tasks are goal directed because the trainer also has the opportunity to inform the robot when the task has completed. This goal state is paired with the robot state and becomes a further training record in the memory model for that particular task. In execution mode the task is iterated until the environmental state is close to a goal state and the task will then terminate.[1] As an example consider an obstacle avoidance behaviour. The trainer would place the robot in an obstacle facing situation, choose the "task" level, label it "Obstacle Avoidance" and press the "start learning" button. The robot can then be moulded into a non-obstacle avoidance situation. The trainer would then signal that the goal state was reached. This training regime could be repeated for many obstacle avoidance situations and thus many obstacle recognition states, appropriate avoidance actions and goal states are recorded into the Obstacle Avoidance memory model.

The final mechanism for learning is a *behaviour*. This allows the trainer to construct the complete behaviour for the robot from the component set of tasks, sequences and primitives. The construction of a behaviour is the same as for a task except that no goal state is required. The behaviour will run continually in execute mode and base its decision of what task, sub-task, sequence or primitive to use based on the current environmental state. With careful training the trainer can now build a hierarchy of tasks, sequences and primitives as required (see figure 3).

[1]There could be instances when a task will never terminate, currently however we do not use a time-out mechanism to avoid the robot continually repeating the same set of actions, but may include this in the future

4.5 Action Selection

The trainer by constructing a hierarchy of tasks, sequences and primitives is now effectively building an action selection architecture for the robot. At the top behavioural level a decision is made based on the robot's current state as to what to execute next (based on the kNN selection). If the selection is a primitive or sequence these will be executed and the next state cycle will begin. Alternatively the selection could be a task. Within the task the robot state selects the next appropriate action, which again could be a primitive, sequence or task. Working down through the hierarchy eventually results in the execution of a primitive. Note that each *task* executed in the hierarchy will only terminate when its goal condition is selected based on the current robot state, thus within the lowest selected task the state will be polled after each executed primitive. This method of action-selection is similar to the extended feed-forward free-flow hierarchy proposed by Tyrrell [25], who demonstrates how hierarchical approaches to action-selection can often exhibit better performance than "Strips" style production rule methods.

5. VALIDATION OF FRAMEWORK

We illustrate the successful functioning of the implemented social learning architecture from using the system on two scaffolded behaviours. The first is simple and illustrates the different ways that a trainer could proceed in training the robot. The second is more complex and shows how a new skill can be added to an existing set of actions. Please note that for reasons of clarity the diagrams only show each unique sequence, task or primitive per memory model. In reality each memory model may have a great many instances of different states for the same sequence, task or primitive.

The first behaviour is called "Scared of Light" and was to train the robot to move forward when a light was off, move backwards when a light was on and avoid bumping into obstacles in all cases (note that the robot had no pre-built competencies other than the basic set of primitives at this stage). Figure 4 shows two different approaches to the task, the first exploits the hierarchy by seperating the behaviour with an "avoid obstacles" sub-task. The second combines both competencies into one behaviour. Both training regimes are successful, however further training episodes may become more difficult with the latter approach.

The second behaviour is called "Tidy Up". This behaviour is a proxy for the household robot described in the introduction to this paper. We provide two containers. One we call the "cupboard", the other we call the "basket". There are a number of objects either plastic or with copper strips. The training regime is much more complex in this instance (see figure 5). This is not only because there is more to teach but also that we need some negative examples. This is im-

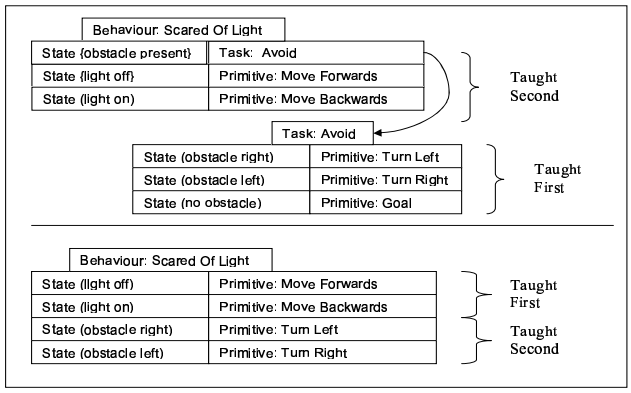


Figure 4: Different teaching styles. The upper part of the diagram shows the avoid task being taught first, followed by scaffolding to recognise when to move forward and backward. In the lower part of the diagram the trainer made no attempt to segment the behaviour. All competencies are added to a single behaviour.

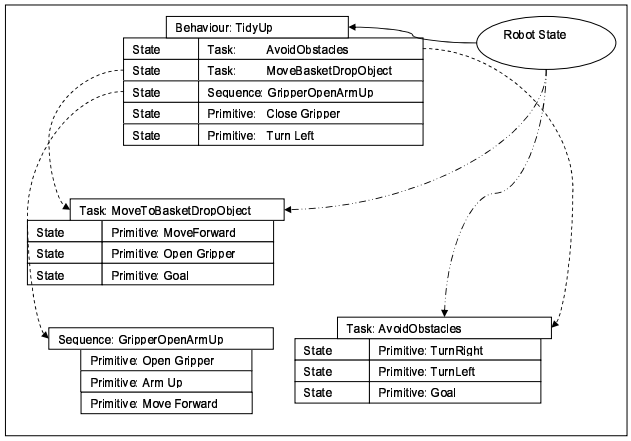


Figure 5: The hierarchy created after successfully training the robot to place plastic containers into the basket (detailed states not shown)

portant to ensure consistent behaviour. For example we had to train the robot to do something sensible if it dropped the object. This situation was scaffolded by initially running the "Tidy Up" behaviour (having already trained the robot to grasp the object) and then removing the object from the gripper. At this point we terminated execution and pressed the learning button. We then selected the "GripperOpenArmUp" sequence and then terminated learning. For the initial "Tidy Up" task seven steps, with up to three scaffolding experiences per task and up to fifteen moulding experiences per scaffold were needed. The robot however executed the behaviour successfully. Figure 7 shows the "Tidy Up" task extended by training the robot to recognise the copper objects and placing them in the "cupboard". The training sequence here involved creating a new task "MoveToCupDropObject", extending the "Tidy Up" task to execute the "MoveToCupDropObject" task if the robot could see the cupboard. Two negative examples were also required. The robot is trained to ignore the cupboard if it has the plastic

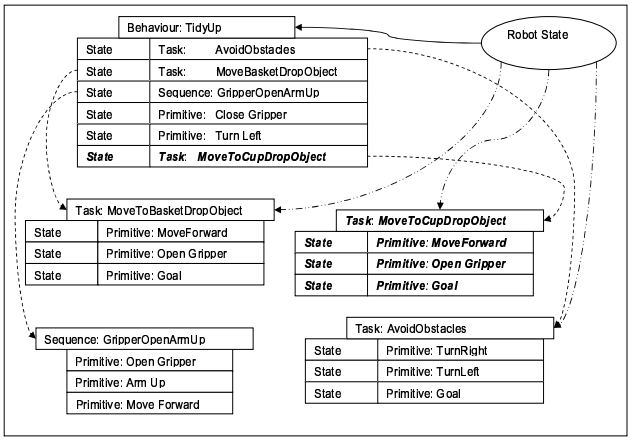


Figure 6: The hierarchy has been extended to allow the robot to succesfully place copper objects in the cupboard whilst still placing plastic objects in the basket (detailed states not shown).

object. Similarly it is trained to ignore the basket if has the copper object.

In some of the training episodes there were indications that suggested that some tasks can be very difficult to demonstrate. For example, the alignment of the robot to successfully pick up a film canister must be precise. If the robot is too close the gripper cannot grasp it, if the robot is slightly misaligned the canister can be knocked over. Demonstrating this ability to the robot as a sub-task proved difficult as the range of possible state attributes was very small in this instance. We think that it may be that certain useful component tasks such as these may be better defined pre-coded as basic primitives i.e. as factory presettings.

6. DISCUSSION

We have described and implemented a robot social learning architecture, inspired from the study of social animals, that allows a human trainer to teach a physical robot without explicit programming. The teaching is based on building up hierarchical sets of reusable competences via interactive scaffolding. Each competence is based on the assumption that experiences captured by the robot as a result of directed human training can be re-applied when the robot experiences a new situation which is similar to those in its set of stored experiences. Thus it "self-imitates", generalising by reproducing its own behaviour in new contexts. The training takes place in real-time and although relatively new the architecture appears to scale from simple to moderately complex tasks successfully. However further experimentation to access performance on tasks of very high complexity will be necessary.

To date we have also obtained limited feedback on the use of the system by non-roboticists where informal tests have indicated that it may not be obvious to a non-technical trainer that a robot may need a developmental program to learn to carry out complex tasks. Although this seems a natural assumption which is made when training other adults, children or animals. This may be simply due to inexperience with

"intelligent" machines or that the robot itself does not "advertise" the fact that it lacks basic skills. Machines up to now have been engineered mostly to work precisely as specified, they are usually not expected to have to be taught or developed in any way.

In our future research we intend to further study these issues and also use the architecture to further investigate how robots could learn from each other.

7. ACKNOWLEDGMENTS

The work described in this paper was partially conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion") and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

8. REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders. An approach for programming robots by demonstration: Generalization across different initial configurations of manipulated objects. In *6th IEEE Int. Symp. Computational Intelligence in Robotics and Automation (CIRA'05)*. IEEE, 2005.

[2] D. C. Bentivegna and C. G. Atkeson. A framework for learning from observation using primitives. In *Proc. RoboCup Int. Symp., Japan*, 2002.

[3] A. Billard and K. Dautenhahn. Experiments in learning by imitation - grounding and use of communication in robotic agents. *Adaptive Behaviour Journal*, 7(3/4), 1999.

[4] R. W. Byrne. *The Thinking Ape: Evolutionary Origins of Intelligence*. Oxford University Press, 1995.

[5] W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. Timbl:tilburg memory-based learner. Technical Report ILK 04-02, Tilburg University, 2004. Available from http://ilk.uvt.nl/.

[6] K. Dautenhahn. Getting to know each other – artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, 16:333–356, 1995.

[7] R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems*, 47:109–116, 2004.

[8] M. Dorigo and M. Colombetti. *Robot Shaping: an experiment in behavior engineering*. MIT Press, 1998.

[9] G. Hayes and J. Demiris. A robot controller using learning by imitation. In *Proc. Int. Symp. Intelligent Robotic Systems, Grenoble*, pages 198–204, 1994.

[10] H.Miyamoto and M.Kawato. A tennis serve and upswing learning robot based on bi-directional theory. *Neural Networks*, 11:1331–1344, 1998.

[11] J.A.Ijspeert, J.Nakanishi, and S.Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *IEEE Int. Conf. Robotics and Automation*, 2002.

[12] T. M. Mitchell. *Machine Learning*. McGraw-Hill International, 1997.

[13] B. R. Moore. *Social Learning in Animals: The Roots of Culture*, chapter The Evolution of Imitative Learning, pages 245–265. Academic Press Inc., 1996.

[14] C. L. Nehaniv and K. Dautenhahn. The Correspondence Problem. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 41–61. MIT Press, 2002.

[15] M. N. Nicolescu and M. M. Matarić. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 31(5):419–430, 2001.

[16] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[17] R.E.Fikes and N.J.Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[18] R.S.Fouts, D.H.Fouts, and T. Cantfort. *Teaching sign language to chimpanzees*, chapter The infant Loulis learns signs from cross fostered chimpanzees, pages 280–92. State University of New York Press, 1989.

[19] C. Sammut, S. Hurst, D. Kedzier, and D. Michie. Learning to fly. In *Proc. Ninth Int. Conf. on Machine Learning*, pages 385–393. Morgan Kaufmann, 1992.

[20] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. An experimental comparison of imitation paradigms used in social robotics. In *Proc. IEEE Robot and Human Interactive Communication (ROMAN '04)*, pages 691–696. IEEE Press, September 2004.

[21] J. Saunders, C. L. Nehaniv, and K. Dautenhahn. An examination of the static to dynamic imitation spectrum. In *Proc. 3rd Int. Symp. on Animals and Artifacts at AISB 2005*, pages 109–118, 2005.

[22] S. Schaal, A.Ijspeert, and A. Billard. *The Neuroscience of Social Interaction*, chapter Computational approaches to motor learning by imitation, pages 199–218. 1431. Oxford University Press, 2004.

[23] T.M.Caro. Predatory behaviour in domestic cat mothers. *Behaviour*, 74:128–47, 1980.

[24] T.M.Caro and M.D.Hauser. Is there teaching in non-human animals? *Quarterly Review of Biology*, 67:151–74, 1992.

[25] T. Tyrrell. Computational Mechanisms for Action Selection, PhD Thesis. Technical report, Centre for Cognitive Science, University of Edinburgh, 1993.

[26] M. van Lent and J. E. Laird. Learning procedural knowledge through observation. In *K-CAP 2001: Proc. Int. Conf. Knowledge Capture*, 2001.

[27] J. V. Wertsch. *Vygotsky and the Social Formation of the Mind*. Harvard University Press, 1985.

Solving the Correspondence Problem in Robotic Imitation across Embodiments: Synchrony, Perception, and Culture in Artifacts

Aris Alissandrakis, Chrystopher L. Nehaniv and Kerstin Dautenhahn

1 The Agent-based Perspective

Imitation is a powerful learning mechanism and a general agent-based approach must be used in order to identify the most interesting and significant problems, rather than the prominent ad hoc approaches in imitation robotics research so far. The traditional approach concentrates in finding an appropriate mechanism for imitation and developing a robot control architecture that identifies salient features in the movements of an (often visually observed) model, and maps them appropriately (via a built-in and usually static method) to motor outputs of the imitator [7, 8]. Model and imitator are usually not interacting with each other, neither do they share and perceive a common context. Effectively this kind of approach limits itself to answering the question of how to imitate for a particular robotic system and its particular imitation task. This has led to many diverse approaches to robot controllers for imitative learning that are difficult to generalize across different contexts and to different robot platforms. In contrast to the above, the agent-based approach for imitation considers the behaviour of an autonomous agent in relation to its environment, including other autonomous agents. The mechanisms underlying imitation are not divorced from the behaviour-in-context, including the social and non-social environments, motivations, relationships among the agents, the agents individual and learning history etc. [4].

Such a perspective helps unfold the full potential of research on imitation and helps in identifying challenging and important research issues. The agent-based perspective has a broader view and includes five central questions in designing experiments on research on imitation: who to imitate, when to imitate, what to imitate, how to imitate and how to evaluate a successful imitation. A systematic investigation of these research questions can show the full potential of imitation from an agent-based perspective. In addition to deciding who, when and what to imitate, an agent must employ the appropriate mechanisms to learn and carry out the necessary imitative actions. The embodiment of the agent and its affordances will play a crucial role, as stated in the *correspondence problem* [12]:

> Given an observed behaviour of the model, which from a given starting state leads the model through a sequence (or hierarchy [or program]) of sub-goals in states, action and/or effects, one must find and execute a sequence of actions using ones own (possibly dissimilar) embodiment, which from a corresponding starting state, leads through corresponding sub-goals - in corresponding states, actions, and/or effects, while possibly responding to corresponding events.

This informal statement[1] of the correspondence problem draws attention to the fact that the agents may not necessarily share the same morphology or may not share access to the same affordances even among members of the same "species". This is true for both biological agents (e.g. differences in height among humans) and artificial agents (e.g. differences in motor and actuator properties). Having similar embodiments and/or affordances is just a special case of the more general problem. In order to study the correspondence problem we developed the ALICE (**A**ction **L**earning via **I**mitation between **C**orresponding **E**mbodiments) generic imitation framework, and implemented it in different simple software testbeds[2].

2 ALICE Overview

The imitative performance of an agent with a dissimilar embodiment to the model will not be successful unless the correspondence problem between the model and the imitator is (at least partially) solved.

To address this in an easy to generalize way, we developed ALICE (**A**ction **L**earning for **I**mitation via **C**orrespondences between **E**mbodiments) as a generic framework for building up correspondences based on *any* generating method for attempts at imitation. This framework is related to statistical string parsing models of social learning from ethology [3] and also the Associative Sequence Learning (ASL) theory from psychology [6].

The ALICE framework (shown in Fig. 1) creates a *correspondence library* that relates the actions, states and effects of the model (that the imitator is being exposed to) to actions (or sequences of actions) that the imitator agent is capable of, depending on its embodiment and/or affordances. These corresponding actions are evaluated according to a *metric* and can be looked up in the library as a partial solution to the correspondence problem when the imitator is next exposed to the same model action, state or effect. It is very important to note that the choice of metric can have extreme qualitative effects on the imitators resulting behaviour [1], and on whether it should be characterized as 'imitation', 'emulation', 'goal emulation', etc. [12].

[1] For a formal statement of the correspondence problem relating to the use of different error metrics and for other applications, see also [9, 10, 11]

[2] These testbeds were implemented using the Swarm agent simulation system (`http:\\wiki.swarm.org`).

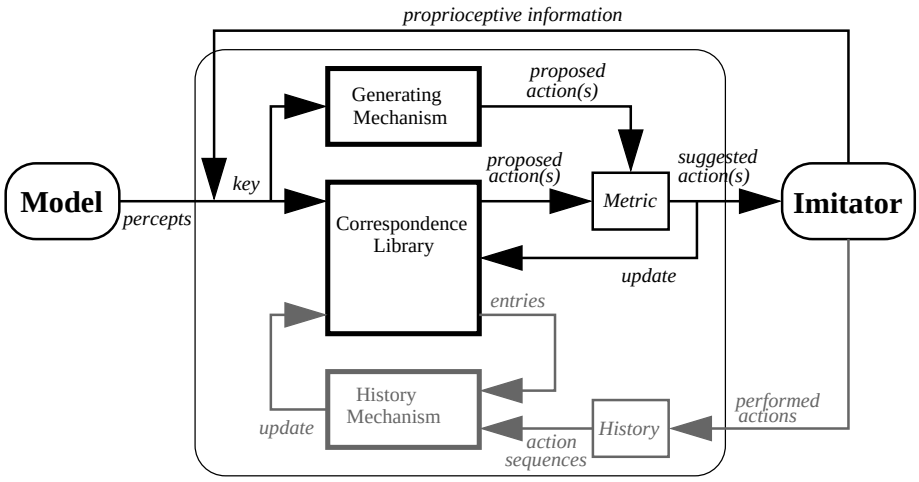


Fig. 1: **The ALICE framework.** The *percepts* of the imitator arising from the model's behavior (actions, states and effects) and *proprioceptive information* (state) of the imitator form a *key* that is used by the *correspondence library* (if it matches any of the existing entry keys at that stage of the library's growth) and the *generating mechanism* to produce a sequence of one or more *proposed action(s)*. These are evaluated using a *metric*, and the correspondence library is updated accordingly with the resulting *suggested action(s)* for the imitator. In parallel (shown in the figure using a gray color), the *history mechanism* can be used to discover any *action sequences* from the *history*, that can improve any of the existing library entries. The history is composed by the sequence of all the actions performed so far by the imitator.

2.1 The Generating Mechanism

The *generating mechanism* is used in the ALICE framework to produce the contents of the correspondence library; it can be *any* algorithm or mechanism that generates actions (or sequences of actions) that are valid (i.e. within the agent's repertoire) and possible in the context of the imitator. Sophisticated applications of ALICE can benefit by replacing, in a modular way, this action generating mechanism with a more sophisticated one, appropriate to the given application. In the ALICE framework, no direct feedback from the model is used, instead the metrics are used to evaluate the imitation attempts.

2.2 The History Mechanism

If only the actions found by the generating mechanism are used to build-up the correspondence library, the performance of the imitator would be directly limited by the choice of the algorithm. Moreover, some of the stored actions, although valid solutions to the correspondence problem, may become invalid in certain contexts. The *history mechanism* helps to overcome these difficulties: The imitator can examine its own history to discover further correspondences without having to modify or improve the generating algorithm used. These correspondences will be *sequences of* actions since, no matter how simplistic, the generating mechanism is required to be able to explore the entire search-space of single actions. An agent's *history* is defined as the list of actions that were performed so far by the agent while imitating the model together with their resulting state and effects. This kind of history provides valuable experience data that can then be used to extract useful mappings to improve and add to the correspondence library created up to that point. This approach can be useful to overcome possible limitations of the generating mechanism [1].

2.3 Building up the correspondence library

When the imitating agent is exposed to each action, state and effect that comprises the model behaviour, the generating mechanism produces a candidate corresponding action. If there is no entry in the correspondence library related to the current action, state and effect of the model, a new entry is created, using these as entry keys with the generated action as the (initial) solution[3].

If instead an entry already exists, the new action is compared to the stored action[4]. If the generated action is worse, according to the metric used, then it is discarded and the existing action from the correspondence library is performed. If on the other hand the new action is better, then it is performed by the agent and the library entry is updated. This could mean that the new action simply replaces the already existing one, or is added as an alternative solution.

[3] More precisely, the contents of the perceptual key depend on the metric the agent is using, for example each of the keys will only contain state(s) and action(s) if a composite state-action metric is used.

[4] There is generally more than one stored corresponding action (or sequence of actions) for each entry, reflecting alternative ways to achieve the same result.

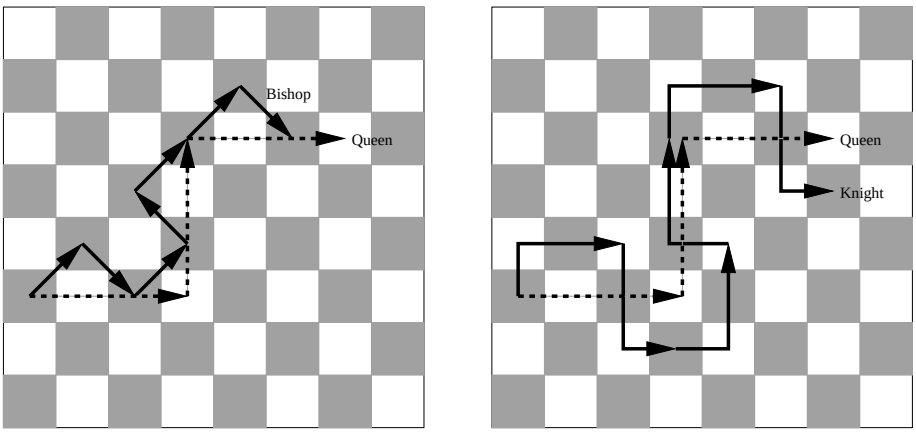


Fig. 2: **Two Chessworld examples.** Two imitator agents (solid paths), a Bishop (left) and a Knight (right) attempt to imitate the movements of the model Queen agent (dotted path).

Over time as the imitating agent is being exposed to the model agent the correspondence library will reflect a partial solution to he correspondence problem that can be used to achieve a satisfactory imitation performance. Effectively ALICE provides a combination of learning and memory to help solve the correspondence problem. There is generalization in that the learned corresponding actions (or sequence of actions) can be reused by the imitator in new situations and contexts.

A more detailed description of the ALICE framework can be found in [2].

3 The Chessworld Testbed

The creation of Chessworld was inspired by the need to implement a shared environment for interacting agents of different embodiments affording different relationships to the world. In the rules of the game of chess, each player controls an army of chess pieces consisting of a variety of different types with different movement rules. We borrow the notion of having different types of chess pieces able to move according to different movement rules, and we treat them as agents with dissimilar embodiments moving on the chequered board. Note that the actual game of chess is not studied. We simply make use of the familiar context of chess in a generic way, to illustrate the correspondence problem in imitation.

The range of possible behaviours by the chess agents is limited to movement-related ones. As a model agent performs a random walk on the board, an imitator observes the sequence of moves used and the relevant displacements achieved and then tries to imitate them, starting from the same starting point. Considering the moves sequentially the agent will try to match them, eventually performing a similar walk on the board. This imitative behaviour is performed

after exposure to a complete model behaviour with no obstacles present, neither static (e.g. walls) nor dynamic (e.g. other moving chess pieces), besides the edges of the board which can obstruct movement.

An action for a given agent is defined as a move from its repertoire, resulting in a relative displacement on the board. For example, a Knight agent can perform move $E2N1$ (hop two squares east and one square north) resulting in a displacement of $(-2, +1)$ relative to its current square.

Addressing what to imitate, the model random walk is segmented into relative displacements on the board by using different granularities. For example, *end-point level granularity* ignores all the intermediate squares visited and emulates the overall goal (i.e. cumulative displacement) of the model agent. In contrast, *path level granularity* not only considers all the squares visited by the model but also the intermediate ones that the chess piece 'slides across' on the chessboard while moving. Between these two extremes, *trajectory level granularity* considers the sequence of relative displacements achieved by the moves of the model during the random walk.

Depending on the embodiment as a particular chess piece, the imitator agent must find a sequence of actions from its repertoire to sequentially achieve each of those displacements. The assessment of how successful a sequence is in achieving a displacement and moving the agent as close as possible to the target square can be evaluated using different simple geometric metrics (Hamming norm, Euclidean distance and infinity norm) that measure the difference between displacements on the chessboard.

3.1 ALICE in Chessworld

The ALICE realization in Chessworld (seen in Fig. 3) corresponds model actions (moves that result in a relative displacement of the chess piece on the board) to actions (or more probably sequences of actions) that can be performed by the imitator. The generating mechanism is a simple greedy algorithm, returning sequences of actions from the imitator agent's repertoire. The list of past moves performed by the imitator is defined as the *history*, from which the agent's history mechanism is looking for sequences of actions that can achieve the same relative displacement as model action entries in the correspondence library. The history mechanism is used in parallel to take advantage of this experiential data, compensating for the generating mechanism not allowing moves that locally might increase the distance, but globally reduce the error, within the generated sequences. The success and character of the imitation observed can be greatly affected by agent embodiment, together with the use of different metrics and sub-goal granularities.

For a more detailed description of Chessworld and the ALICE implementation in this testbed, see [1].

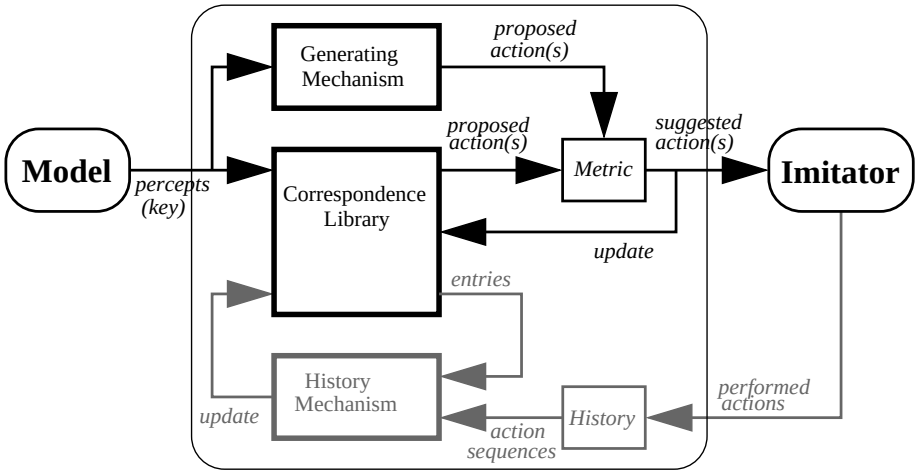


Fig. 3: **The ALICE framework as realized in the Chessworld testbed.** Note that compared to the generic version of the framework (shown in Figure 1), *proprioceptive information* from the imitator is not used here, as the realization of the ALICE framework in Chessworld only considers the *action* aspects of the agent's behavior and not the *state* or the *effects*. Other components are as explained in Fig. 1.

4 The Rabit Testbed

The Rabit (**R**obotic **A**rm em**B**odiment for **I**mitation **T**estbed) environment was created as simple, yet 'rich enough' to allow for several dissimilarly embodied model and imitator agents to be considered. A Rabit agent (see Fig. 4) occupies a two-dimensional workspace and is embodied as a robotic arm that can have any number of rotary joints, each of varying length. Each agent embodiment is described by the vector $L = [\ell_1, \ell_2, \ell_3 \cdots, \ell_n]$, where ℓ_i is the length of the i^{th} joint. There are no complex physics in the workspace and the movement of the arms is simulated using simple forward kinematics but without collision detection or any static restraints (in other words, the arms can bend into each other). Our intention is to demonstrate the features of the imitative mechanism and not to build a faithful simulator.

An *action* of a given agent is defined as a vector describing the change of angle for each of the joints, $A = [\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n]$, where n is the number of its joints. These angles are relative to the previous state of the arm and can only have three possible values, $+10°$ (anti-clockwise), $0°$ or $-10°$ (clockwise).

A *state* of an agent is defined as the absolute angle for each of the joints, $S = [\sigma_1, \sigma_2, \sigma_3, \cdots, \sigma_n]$, where n is the number of its joints. A distinction can be made between the *previous state* and the *current state* (the state of the arm after the current action was executed). As a result of the possible actions, the absolute angle at each joint can be anywhere in the range of $0°$ to $360°$ (modulo

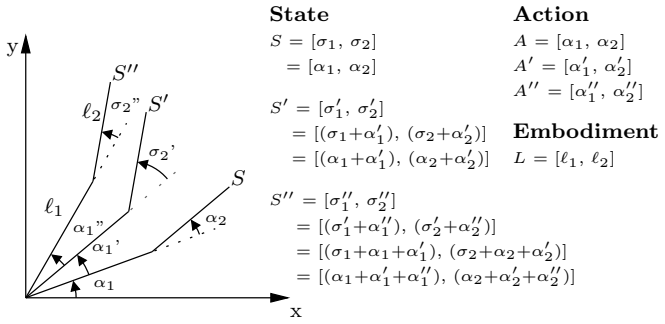


Fig. 4: **Example embodiment of a Rabit agent.** A two-joint robotic arm, with arms of length ℓ_1 and ℓ_2, moving from state S to state S' to state S'', as it sequentially performs actions A, A', and A''. Note that the effects are not shown in this figure.

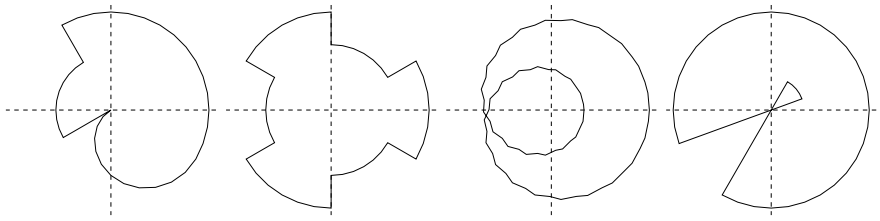


Fig. 5: **Different examples of Rabit behaviours.** Shown are four different effect trails (the agent embodiment is not shown), drawn by the end tip of the each agent manipulator arm. All agents shown have the same embodiment $L = [20, 20, 20]$.

360°) in but only in multiples of 10°.

The end tip of the arm can leave a trail of 'paint' on the workspace, as it moves along the workspace. The *effect* is defined as a directed straight line segment connecting the end tip of the previous and the current states of the arm (approximating the paint trail). The effect is internally implemented as a vector of displacement $E = (x_c - x_p, y_c - y_p)$, where (x_p, y_p) and (x_c, y_c) are the end tip coordinates for the previous and current state respectively.

The model behaviour is broken down as a sequence of actions that move the robotic arm of the agent from the previous state to the current state, while leaving a behind a trail of paint as the effect. The nature of the experimental testbed with the fixed base rotary robotic arms favours circular looping effects and the model behaviours used in the experiments were designed as such (see Fig. 5).

Each complete behaviour (or "pattern") that returns the arm to its initial state observed by the imitator is called an exposure, and the imitator is exposed

to repeated instances of the same behavioural pattern. At the beginning of each new exposure it is possible to reset the imitating agent to the initial state. This resetting is called *synchronization* in our experiments.

4.1 Metrics

The imitating agents can perceive the actions, states and effects of the model agents, and also their own actions, states and effects, and therefore we define several metrics to evaluate the similarity between them. Ideally the metric value should be zero, indicating a perfect match. An example of using the different metrics described below is shown in Fig. 6.

4.1.1 State metric

The state metric calculates the average distance between the various joints of an agent (posed in a particular state) and the corresponding joints of another agent[5] (posed in a different state) as if they were occupying the same workspace. Ideally this distance should be zero when the arms take corresponding poses, but this may not be possible due to embodiment differences. Using forward kinematics, the coordinates of the ends for each joint are found.

$$x_i = \sum_{j=1}^{i-1} x_j + \ell_i \cos(\sum_{j=1}^{i} \sigma_j) \tag{1a}$$

$$y_i = \sum_{j=1}^{i-1} y_j + \ell_i \sin(\sum_{j=1}^{i} \sigma_j) \tag{1b}$$

If both agents have the same number of joints the correspondence between them is straightforward; the Euclidean distance for each pair is calculated, the distances are then all summed and divided by the number of joints to give the metric value.

$$d_i = \sqrt{(x_i^{\text{model}} - x_i^{\text{imitator}})^2 + (y_i^{\text{model}} - y_i^{\text{imitator}})^2} \tag{2}$$

$$\mu^{state} = \frac{1}{n} \sum_{i=1}^{n} d_i \tag{3}$$

If the agents have a different number of joints, then some of the joints of the agent with more are ignored. To find which joint corresponds with which, the ratio of the larger over the smaller number of joints is calculated, and if not integer, is rounded to the nearest one. The i^{th} joint of the agent with the smaller number of joints, will correspond to the $(ratio \times i)^{\text{th}}$ joint of the agent with the larger number of joints. For example if one of the agents has twice the number of joints, only every second joint will be considered.

[5] The state metric can be used not only between different agents, but also to evaluate the similarity between two states of the same agent. This is true for the action and the effect metric as well.

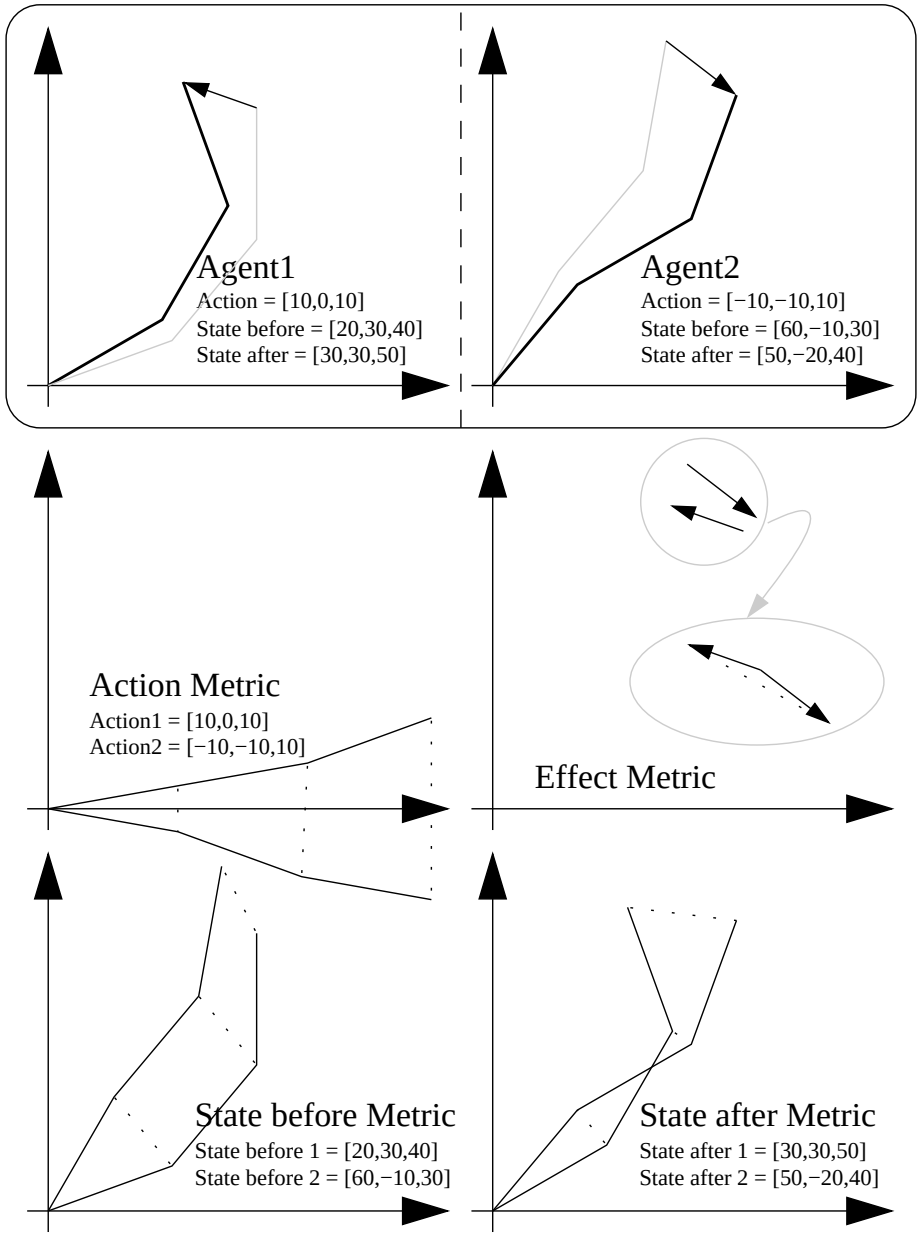


Fig. 6: **An example of using the metrics to compare actions, states (before and after) and effects between two Rabit agents, *Agent1* (top, left) and *Agent2* (top, right).** The figure visualizes the vectors used (depending on the metric) and the distances that are summed and then averaged to give each metric value. Both agents have the same embodiment $L = [20, 20, 20]$.

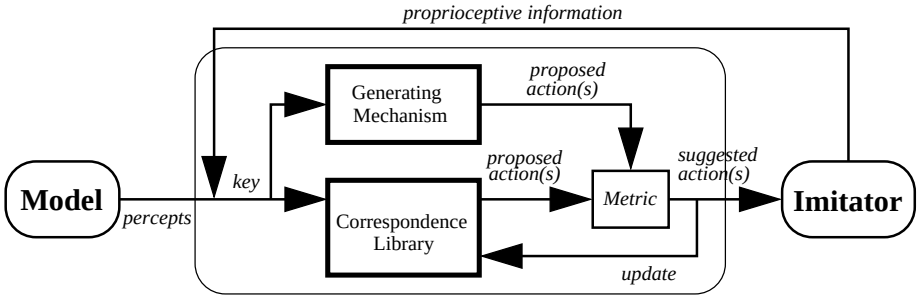


Fig. 7: **The ALICE framework as realized in the Rabit testbed.** Note that compared to the generic version of the framework (shown in Figure 1), the *history mechanism* is not used in parallel to update the correspondence library, as the realization of the ALICE framework in Rabit testbed considers only single actions and not sequences. Other components are as explained in Fig. 1.

4.1.2 Action metric

For the action metric, the same algorithm as the one described above for the state metric is used, but considering the action vectors instead of the state vectors. The value in the case of the state metric represents an absolute position error; for the action metric, it represents the relative error between the changes of the state angles, due to the compared actions.

4.1.3 Effect metric

The effect metric is defined as the Euclidean length of the vector difference between two effects (x_1, y_1) and (x_2, y_2).

$$\mu^{effect} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4}$$

4.2 ALICE implementation

In the Rabit implementation of ALICE, each entry in the correspondence library can use as a key the action/state/effect of the observed model agent and the current state of the imitator, as perceptual and proprioceptive components respectively. The key can be composed of just a single of these aspects (e.g. action only), or a combination (e.g. action, state and the imitator's state for proprioception).

For the generating mechanism, an algorithm that returns single random (yet valid) actions is used. It is possible to replace it with a more complex generating mechanism (i.e. inverse kinematics), but the idea is to have a mechanism that simply returns valid actions from the search space. In order to speed up the

learning, it is possible to generate more than one random action and choose a best one.

It is possible not to require an exact match for the perceptual and/or the proprioceptive components of the trigger key, but a loose one that is "close enough", controlled by a threshold. We call this *loose perceptual matching* and we hypothesized that it should support learning and generalization.

In the current implementation, each entry can store up to three possible corresponding actions that can be seen as possible alternatives[6].

For a more detailed description of Rabit and the ALICE implementation in this testbed, see [2].

5 Experiments on aspects of imitation

Using the robotic arm testbed we conducted various experiments to study the possibility of social transmission of behaviours through heterogeneous agents, the effect of proprioception, loose perceptual matching and synchronization on the imitation learning performance, and also the robustness of the ALICE mechanism when the imitator embodiment changes during the learning process, and also after achieving a successful imitative performance.

5.1 Cultural transmission of behaviours and emergence of 'proto-culture'

Besides being a powerful learning mechanism, imitation broadly construed is required for cultural transmission (e.g. [5]). Transmission of behavioural skills by social learning mechanisms like imitation may also be fundamental in non-human cultures, e.g. in chimpanzees [14], whales and dolphins [13]. The robotic arm testbed makes it possible to study examples of behavioural transmission via imitation, with an imitator agent acting as a model for another imitator. If the original model and the final imitator have the same embodiment but the intermediate imitator a different one, we can look at how the different embodiment and the choice of metrics for the evaluation of a successful imitation attempt can affect the quality of the transmitted behaviour.

The example shown in Fig. 8 shows such a transmission of the original model behaviour via an intermediate agent. Although the intermediary has a different embodiment, the original model and final imitator have the same embodiment, and the model behavioural pattern is transmitted perfectly. This is partially helped by the use of the action metric for evaluation to overcome the dissimilar embodiment of the transmitting agent. This example serves as proof of the concept that by using social learning and imitation, rudimentary cultural transmission with variability is possible among robots, even heterogeneous ones.

[6] Note that the history mechanism which also considers sequences of past imitative attempts when updating the correspondence library entries is not implemented in the Rabit testbed since simple action to action correspondence suffices here. In contrast, corresponding sequences of actions are necessary in Chessworld as most chess pieces are unable to move as far as their model using only a single action.

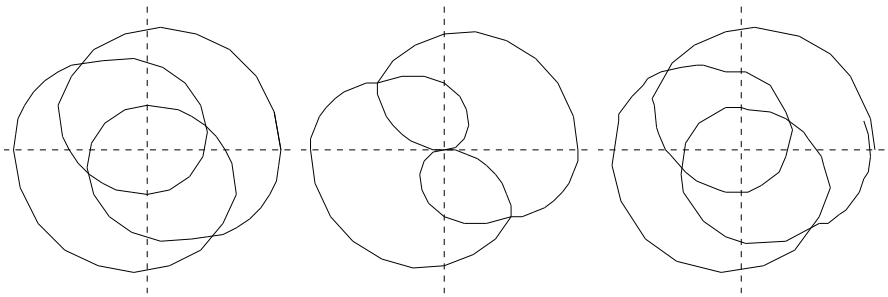


Fig. 8: **An example of social transmission.** The original model ($L = [20, 20, 20]$) is shown to the left. In the middle, an imitator ($L = [30, 30]$) acts also as a model for the imitator on the right ($L = [20, 20, 20]$). Both imitators use the action metric.

The choice of metrics and the particular embodiment of the agents greatly affect the qualitative aspects of imitation, making not every combination suitable for passing on model behaviours, besides crucial aspects of the model behaviours themselves. Note that in Fig. 8, the intermediate agent imitates qualitatively differently, due to its dissimilar embodiment. If the particular embodiment of the intermediate agent greatly distorts the model pattern, then such a transmission might be impossible.

The examples shown in Figs. 9 and 10 illustrate the emergence of 'proto-culture' in a cyclically ordered chain of three and eight imitators with no overall model. The agents imitate only the agent clockwise from them, using the action metric. Initially they move randomly, as the generating mechanism is trying to discover correspondences for the (also random) actions of their model. Over time, they are able to imitate each other's actions and a stable behavioral pattern emerges.

Different runs yield different emergent culturally sustained behaviors. The location and orientation of the emergent pattern is different in each agent's workspace, since the location and orientation are irrelevant to the action metric; they will depend on the state of the agent at the moment that it has solved its correspondence problem. Each agent's state will vary as a result of the agents not synchronizing.

The cultural transmission of skills through a heterogeneous population of robots using the ALICE framework could potentially be applied to the acquisition and transmission of skills in more complex populations of robots, involved in carrying out useful tasks, e.g. on the shop-floor of a factory, with new robots coming and going acquiring behaviors by observation without having to be explicitly programmed and without humans having to develop different control programs for different types of robots that need to perform the same task. Instead, the robots would autonomously create their own programs (using social learning) and correspondence libraries, even as new types of robots with

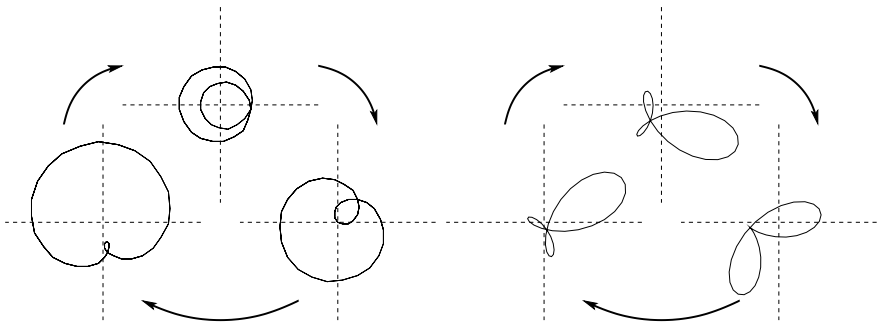


Fig. 9: **Examples of emerging 'proto-culture'.** Two groups of three Rabit agents. All the agents on the left have $L = [20, 20, 20]$ and all the agents on the right have $L = [15, 15, 15, 15]$. Each agent imitates the agent on its left (anti-clockwise) and acts as a model for the agent on its right (clockwise). The emergent behaviour is composed of a single action. $[10, 10, 0]$ left, $[10, -10, -10, -10]$ right.

different embodiments come and go from the population.

5.2 Synchronization

At the end of each exposure of the imitating agent to the model, it is possible to reset the imitator arm to the same initial position, as a result synchronizing the imitation attempt to the model behaviour. We conducted ten experimental runs, each with two imitating agents trying to imitate a model agent, one of them synchronizing with the model by resetting to the initial outstretched initial state after the completion of each exposure, and the other starting each attempt from the final reached state of the previous attempt (ideally the same as the initial state, as all the model patterns are designed as closed loops). Both model and imitator agents had the same embodiment ($L = [20, 20, 20]$) and the metric used was a weighted half-half combination of the action and state metrics. Both imitating agents use proprioception and allow for a 10% margin of looseness for matching the trigger keys (see section 5.4 below). The generating mechanism was creating five random actions to choose from. Each run lasted twenty exposures and the maximum metric value for each exposure was logged. The ratio of the maximum error of the imitating agent that uses synchronization over the maximum error of the agent that does not reset back the start position at the end of each exposure can be seen in the bottom panel of Fig. 11, constantly decreasing and below 1. This indicates that the numerator is minimized faster than the denominator, indicating that it is very difficult for an imitating agent that does not synchronize to reach again states relevant to the model pattern if the initial imitation attempts are not successful. This reduces the chance to update and improve the relevant correspondence library entries as

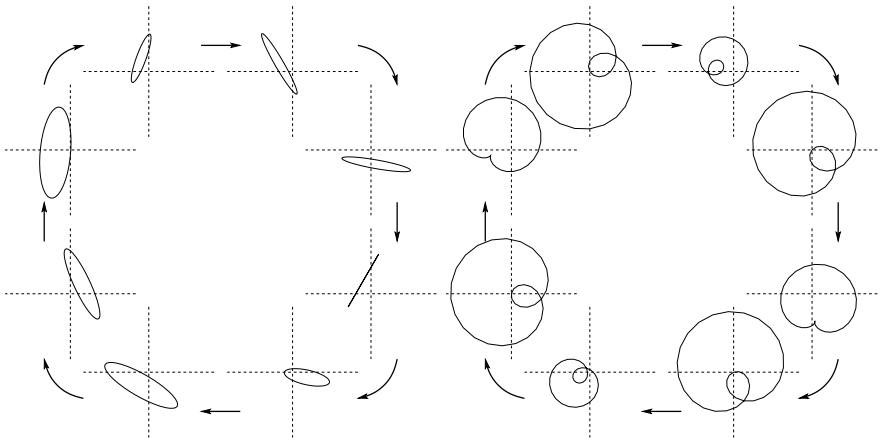


Fig. 10: **More examples of emerging 'proto-culture'.** Two groups of eight Rabit agents. All the agents on the left have $L = [15, 15, 15, 15]$. The agents on the right have alternating embodiments $L = [30, 30]$ and $L = [15, 15, 15, 15]$. Each agent imitates the agent on its left (anti-clockwise) and acts as a model for the agent on its right (clockwise). The emergent behaviour is composed of a single action. $[10, -10, -10, 0]$ left, $[10, 10]$ and $[0, 10, 0, 10]$ right

the agent wanders with no point of reference. If the state space is large enough, it is possible for the agent to get completely lost.

5.3 Proprioceptive matching

The correspondence library entry keys can contain both perceptive (the action, state and effect of the model agent) and proprioceptive (the imitators own state at the time of the observation) data. It is possible to ignore the prioperception and trigger the keys based only on the perception.

We conducted ten experimental runs, each with two imitating agents trying to imitate a model agent, one of them using proprioception, the other not. Both model and imitator agents had the same embodiment ($L = [20, 20, 20]$) and the metric used was a weighted half-half combination of the action and state metrics. Both imitating agents used a loose perceptual matching of 10% (see section 5.4 below) and the generating mechanism was creating five random actions to choose from. Each run lasted twenty exposures and the maximum error metric value for each exposure was logged.

The ratio of the maximum error per exposure of the imitating agent that does not use proprioceptive matching over the maximum error of the imitating agent that does can be seen in Fig. 12 (bottom panel), constantly decreasing and below 1. This indicates that the numerator is minimized faster than the denominator. This indicates that ignoring the proprioceptive component improves the

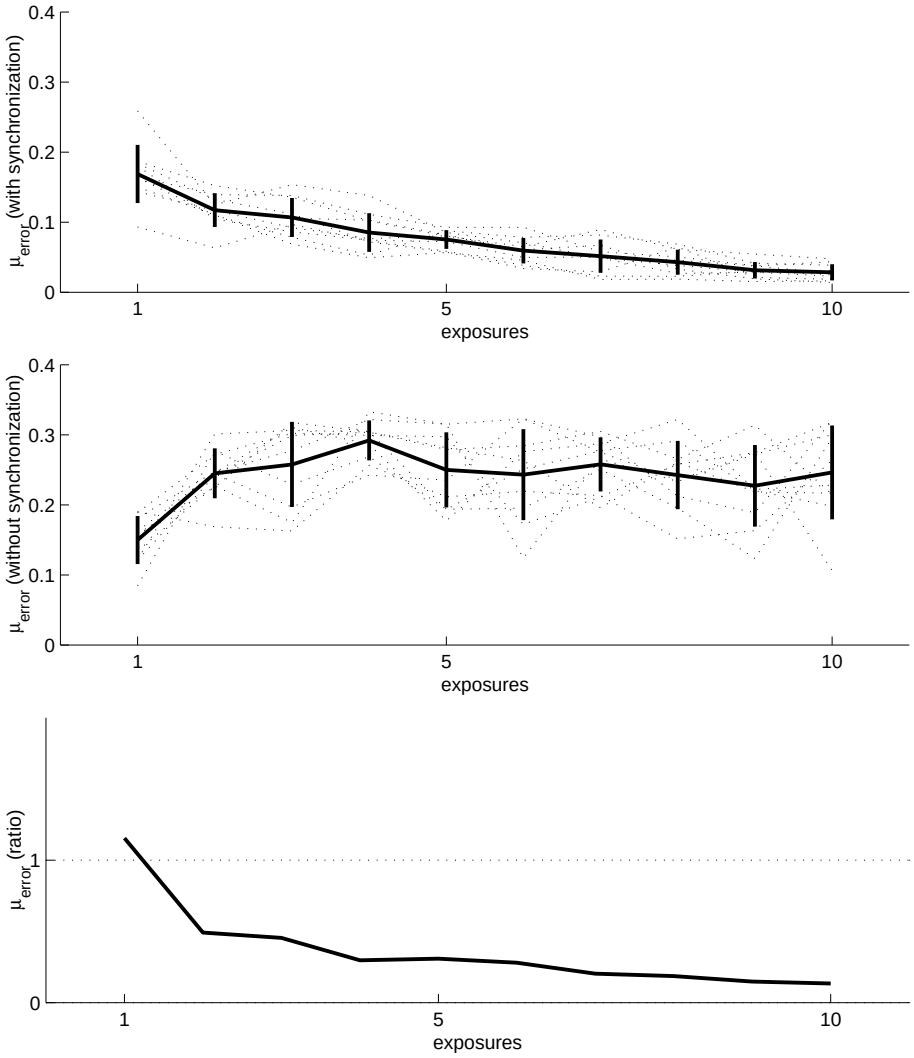


Fig. 11: **Experiments comparing the use of synchronization.** The average maximum error metric value of robotic agents over 10 exposures using synchronization (top panel) vs. not using synchronization (middle panel). The ratio of the maximum error per exposure of the imitating agent using synchronization over the maximum error of the imitating agent that does not use synchronization (bottom panel) indicates a comparative many-fold reduction of error with use of synchronization. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment $L = [20, 20, 20]$ and the imitator agents use a half-half composite of the action and state metrics. Both imitators use proprioception and allow for 10% loose perceptual matching.

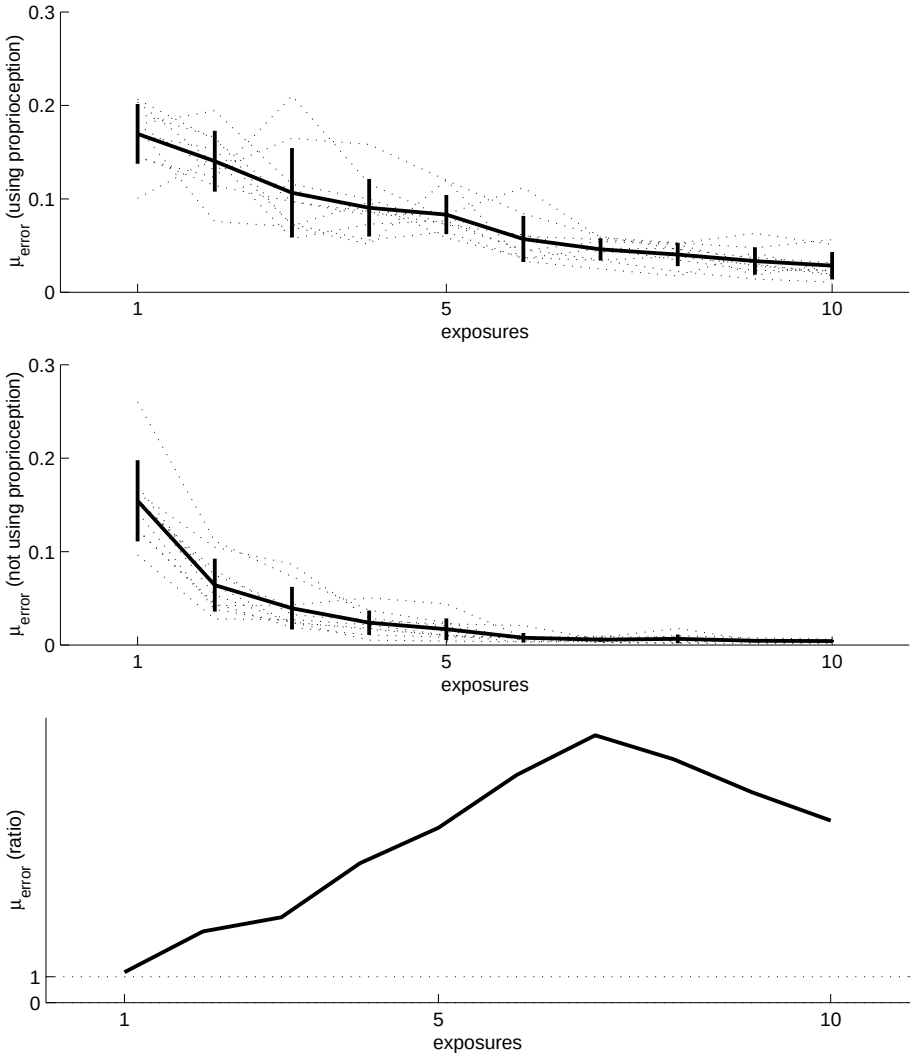


Fig. 12: **Experiments comparing using and not using proprioception.** The maximum error metric value of robotic agents over 10 exposures not using proprioception (top panel) vs. using proprioception (middle panel) when searching through the correspondence library entry keys. The ratio of the maximum error per exposure of the imitating agent not employing proprioception over the maximum error of the imitating agent that does (bottom panel) indicates some comparative reduction of error when not using proprioception. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment $L = [20, 20, 20]$ and the imitator agents use a half-half composite of the action and state metrics. Both imitators synchronize and allow for 10% loose perceptual matching.

performance rate. Ignoring the proprioceptive component of the entry keys will confine the number of entries only to the number of different actions, states and effects that define each model pattern, resulting in a much smaller search space. This reduced number of entries in the correspondence library will have the opportunity to update and improve more often, and explains the performance rate improvement. However given enough time, it is expected that proprioception would allow the imitator to eventually learn much finer control in distinguishing appropriate choices of matching actions depending on its own body state[7].

5.4 Loose perceptual matching

When the ALICE mechanism looks in the correspondence library to find the relevant entry to the currently perceived model actions, states and effects, it is possible not to require an exact match of the entry keys, but one that is close enough, depending on a threshold. We conducted ten experimental runs under the same conditions. Each run consisted of twenty exposures to the model behaviour for two imitating agents, one of them accepting a 10% margin of looseness for the trigger keys and the other one requiring an exact match, both using proprioception. Model and imitator agents have the same embodiment ($L = [20, 20, 20]$) and the metric used was a weighted half-half combination of the action and state metrics. The generating mechanism for the imitating agents was creating five random actions to choose from. The maximum metric value for each exposure was logged and is shown in Fig. 13, using loose matching (top panel) and exact matching (middle panel).

The ratio of the maximum error of the agent that uses loose over the agent that uses exact matching can be seen in the bottom panel of Fig. 13, constantly decreasing and below 1. This indicates that the numerator is minimized faster than the denominator, showing a faster improvement of performance for the imitator agent using loose matching. Examining the middle panel of Fig. 13, there is no obvious performance improvement in this early stage of learning, although the same amount of time is enough to minimize the error for the agent using a loose matching in the top panel. This is mostly due to the large number of entries created in the correspondence library due to the different proprioceptive states that the agent visits during the imitation attempts. The exact match requirement will create a large number with the same perceptive but different proprioceptive part of the keys.

5.5 Changes in the agent embodiment

For each agent, vector L defines its embodiment, the number of arm segments and their lengths. We can define a growth vector G, of same size as L. By adding (or subtracting) these two vectors we get L, a new embodiment with modified joint lengths, simulating the development of the agent. The growth vector can either increase or reduce the length for each of the joints. The number of joints

[7] In this implementation, using proprioception increases the size of the search space by a factor of 36 to the n^{th} power, where n is the number of joints in the imitator.

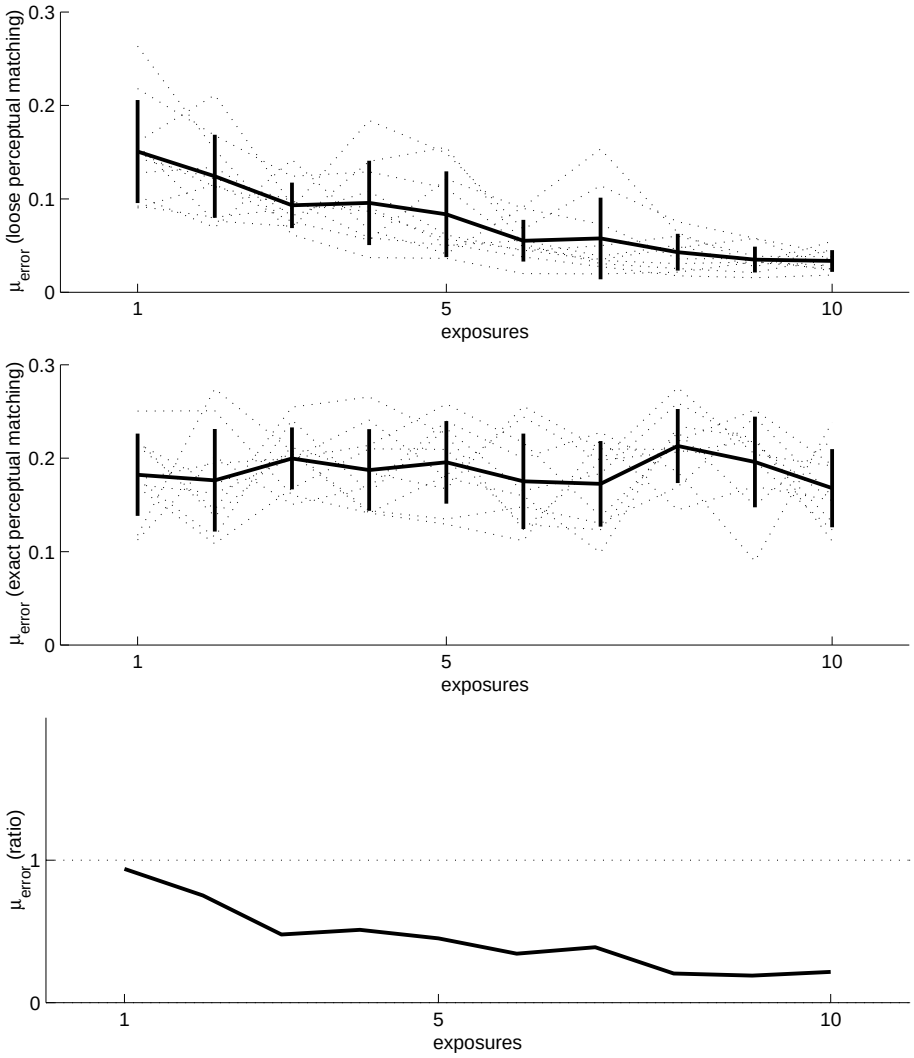


Fig. 13: **Experiments comparing the use of loose perceptual matching.** The average maximum error metric value of robotic agents over 10 exposures using loose matching (top panel) vs. using exact matching (middle panel). The ratio of the maximum error per exposure of the imitating agent using loose matching over the maximum error of the imitating agent that uses exact matching (bottom panel) indicates a comparative many-fold reduction of error with use of loose matching. In each panel, the thicker line shows the average values of all the ten experiments, with the bars indicating the standard deviation. Both model and imitator agents have the same embodiment $L = [20, 20, 20]$ and the imitator agents use a half-half composite of the action and state metrics. Both imitators synchronize and use proprioception.

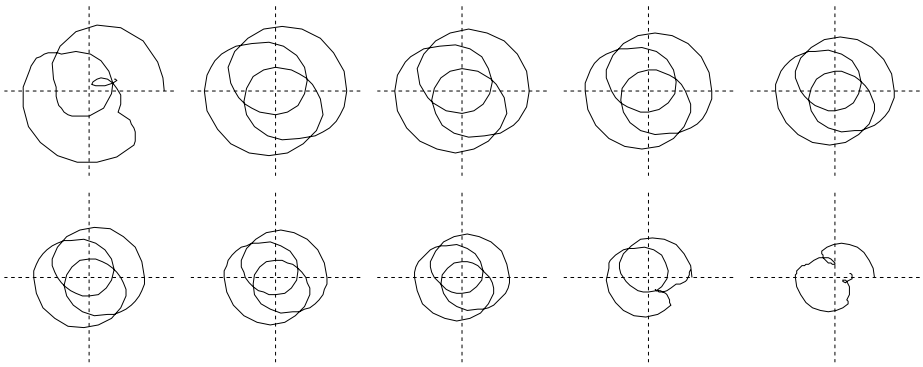


Fig. 14: **Example of an agent imitating with a changing embodiment.** The initial embodiment of the imitator is $L = [20, 20, 20]$ (top left). Shown are the effect trails of the imitator (left to right, top to bottom) for ten consecutive imitation attempts. A growth vector $G = [-1, -1, -1]$ is used.The final embodiment is $L = [10, 10, 10]$ (bottom right).

must remain constant because such a change makes the existing contents so far of the correspondence library invalid[8]. The growth vector can be used to simulate the body development of the imitator agent during the learning process.

Figs. 14 to 16 show examples of imitator agents that try to imitate a model while a growth vector is used after each imitation attempt, modifying their embodiment[9]. In these examples, the growth vecors equally expand or shorten the length of the imitator's arm segments. Although the imitator constantly changes embodiment, the learning process is relatively unaffected, resulting in a robust imitation performance.

The metric used in these examples is the action metric, compensating for the large range of dissimilar embodiments, and the difference in what they afford. The choice of metrics greatly affects the character and quality of the imitation, especially between dissimilar embodiments. For example if the effect metric is used instead of the action metric, very poor results are observed, as the paint strokes created by the shorter joints cannot successfully compensate for the longer strokes achieved by the longer arms of the reference model. Fig. 17 shows an example of the qualitative effect if the state metric is used, instead of the action metric. The growth vector used is $G = [0, -1, 0]$, shortening the imitator's middle arm segment. The action metric is less affected by the embodiment modification, resulting in a "smaller" version of the model's effect trail (shown in gray). In contrast, the imitator using the state metric effectively

[8] A robotic arm with a different number of joints would not be able to perform the stored actions, as they describe the angle changes for each of the existing arm joints when those actions were created.

[9] The model agents preserve a constant embodiment.

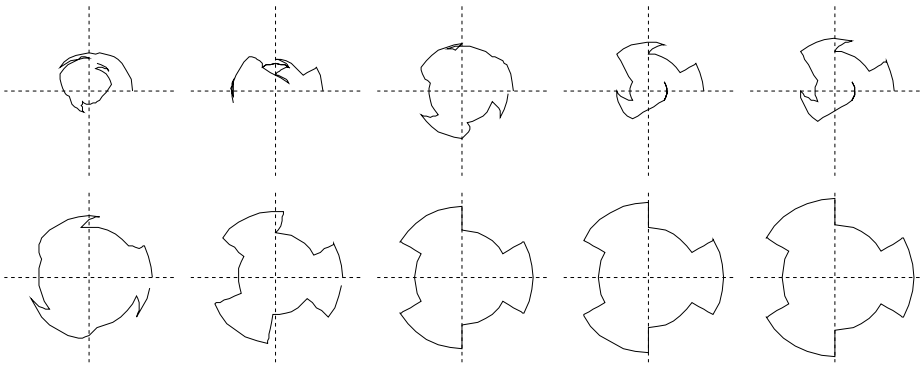


Fig. 15: **Example of an agent imitating with a changing embodiment.** The initial embodiment of the imitator is $L = [10, 10, 10]$ (top left). A growth vector $G = [1, 1, 1]$ is used.The final embodiment is $L = [20, 20, 20]$ (bottom right).

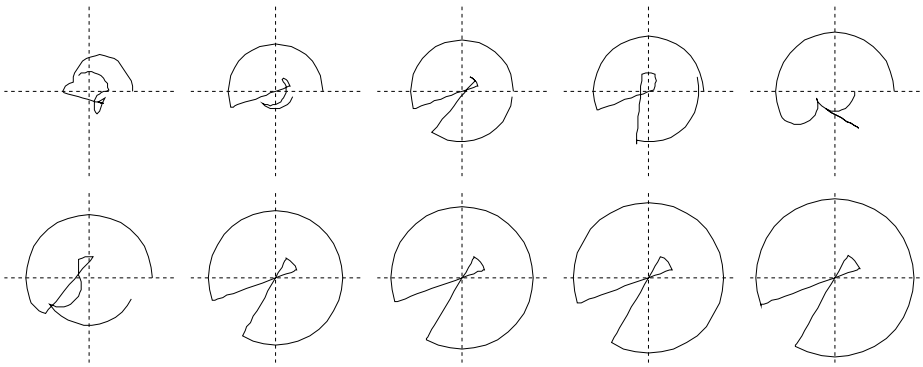


Fig. 16: **Example of an agent imitating with a changing embodiment.** The initial embodiment of the imitator is $L = [10, 10, 10]$ (top left). A growth vector $G = [1, 1, 1]$ is used.The final embodiment is $L = [20, 20, 20]$ (bottom right).

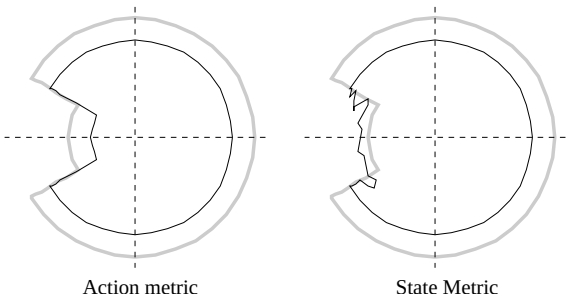


Fig. 17: **Qualitative effect of the metric used by an imitating agent that changes embodiment.** Both imitator agents have the same initial embodiment as the model ($L = [20, 20, 20]$). The figure showns the imitators after ten imitation attempts, having used a growth vector $G = [0, -1, 0]$ after each exposure, with modified embodiments $L = [20, 10, 20]$. The imitator on the left used the action metric, while the imitator on the right used the state metric. The superimposed grey trail shows the model effect pattern for qualitative comparison.

tries to conserve the shape of the pattern by performing actions that acheive similar states.

These examples show that the ALICE mechanism can be robust enough (with a certain tolerance) to compensate for embodiment changes during the initial learning stage (or even later, if the imitator can be again exposed to the model).

6 Conclusions and Discussion

In nature, many organisms' bodies grow and change in the course of their lives. Still the ones that learn socially are able to retain and adapt socially transmitted capabilities despite these changes, whether injurious or natural, to their embodiment. Robots too and other artificial agents that learn socially could benefit from such robustness to embodiment changes. Such a capacity to adapt socially learning despite embodiment change has been demonstrated here via an artificial intelligence learning mechanism framework (ALICE), where the learning is guided by previous experience and evaluation according to given metrics to solve a correspondence problem.

We also showed that loose perceptual matching and synchronization with the demonstrator each resulted in faster learning with lower error rates. Counter-intuitively, in the experiments here use of proprioception in building up a correspondence slowed learning. This is most likely due to the larger state space — there is more to learn if proprioception is employed; however, we hypothesize that further work will show that its employment is ultimately beneficial for longer term learn in more complex scenarios.

The work here demonstrates the principle that artificial social learning mech-

anisms, such as implementations of ALICE, for solving the correspondence problem can be used in populations of robots or agents, to achieve cultural transmission in such populations, even heterogeneous ones consisting of individuals whose embodiments are dissimilar. This may in the future prove useful in the autonomous social learning and adaptation of groups of robots, on factory shop floor or elsewhere, and in social learning interactions in which heterogeneous agents are in involved, e.g. in human-robot interaction. For example, a human might demonstrate a task to a factory robot, which then carries it out, adapting its actions even when its embodiment is perturbed (e.g. by wear-and-tear). Later, when new model robots with different kinds of actuators, degrees of freedom, and so on, come to work in the factory, they acquire skills and task capabilities by learning socially from the robots that are already there. Over generations of robots, cultural knowledge is transmitted, with the robots adapting it to their own changing embodiments.

Acknowledgments

The work described in this paper was partially conducted within the EU Integrated COGNIRON ("The Cognitive Robot Companion") and funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

References

[1] A. Alissandrakis, C. L. Nehaniv, and K. Dautenhahn. Imitation with ALICE: Learning to imitate corresponding actions across dissimilar embodiments. *IEEE Trans. Systems, Man & Cybernetics: Part A*, 32(4):482–496, 2002.

[2] Aris Alissandrakis. *Imitation and Solving the Correspondence Problem for Dissimilar Embodiments – A Generic Framework*. PhD thesis, University of Hertfordshire, 2003.

[3] R. W. Byrne. Imitation without intentionality. using string parsing to copy the organization of behaviour. *Animal Cognition*, 2:63–72, 1999.

[4] K. Dautenhahn and C. L. Nehaniv. An agent-based perspective on imitation. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 1–40. MIT Press, 2002.

[5] Richard Dawkins. *The Selfish Gene*. Oxford University Press, 1976.

[6] C. M. Heyes and E. D. Ray. What is the signtificance of imitation in animals? *Advances in the Study of Behavior*, 29:215–245, 2000.

[7] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observations of human performance. *IEEE Trans. Robot. Automat.*, 10:799–822, November 1994.

[8] Y. Kuniyoshi, H. Inoue, and M. Inaba. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. In *Proc. IEEE International Workshop on Intelligent Robots and Systems IROS '90*, pages 567–574, 1990.

[9] C. L. Nehaniv and K. Dautenhahn. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In John Demiris and Andreas Birk, editors, *Proceedings European Workshop on Learning Robots 1998 (EWLR-7), Edinburgh, 20 July 1998*, pages 64–72, 1998.

[10] C. L. Nehaniv and K. Dautenhahn. Of hummingbirds and helicopters: An algebraic framework for interdisciplinary studies of imitation and its applications. In J. Demiris and A. Birk, editors, *Interdisciplinary Approaches to Robot Learning*, pages 136–161. World Scientific Series in Robotics and Intelligent Systems, 2000.

[11] C. L. Nehaniv and K. Dautenhahn. Like me? - measures of correspondence and imitation. *Cybernetics and Systems*, 32(1-2):11–51, 2001.

[12] C. L. Nehaniv and K. Dautenhahn. The correspondence problem. In K. Dautenhahn and C. L. Nehaniv, editors, *Imitation in Animals and Artifacts*, pages 41–61. MIT Press, 2002.

[13] L. Rendell and H. Whitehead. Culture in whales and dolphins. *Behavioral and Brain Sciences*, 24(2):309–382, 2001.

[14] A. Whiten, J. Goodall, W.C. McGrew, T. Nishida, V. Reynolds, Y. Sugiyama, C.E.G. Tutin, R.W. Wrangham, and C. Boesch. Cultures in chimpanzees. *Nature*, 399:682–685, 1999.

Evaluation of Robot Imitation Attempts: Comparison of the System's and the Human's Perspectives

Aris Alissandrakis, Chrystopher L. Nehaniv, Kerstin Dautenhahn and Joe Saunders
Adaptive Systems Research Group
School of Computer Science
University of Hertfordshire
College Lane
Hatfield, Hertfordshire AL10 9AB
United Kingdom
a.alissandrakis@herts.ac.uk

ABSTRACT

Imitation is a powerful learning tool when humans and robots interact in a social context. A series of experimental runs and a small pilot user study were conducted to evaluate the performance of a system designed for robot imitation. Performance assessments of similarity of imitative behaviours were carried out by machines and by humans: the system was evaluated quantitatively (from a machine-centric perspective) and qualitatively (from a human perspective) in order to study the reconciliation of these views. The experimental results presented here illustrate how the number of *exceptions* can be used as a performance measure by a robotic or software imitator of an object manipulation behaviour. (In this context, exceptions are events when the optimal displacement and/or rotation that minimize the dissimilarity metrics used to generate a corresponding imitative behaviour cannot be directly achieved in the particular context.) Results of the user study giving similarity judgments on imitative behaviours were used to examine how the quantitative measure of the number of exceptions (*from a robot's perspective*) corresponds to the qualitative evaluation of similarity (*from a human's perspective*) for the imitative behaviours generated by the JABBERWOCKY system. Results suggest that there is a good alignment between this quantitive system-centered assessment and the more qualitative human-centered assessment of imitative performance.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.9 [**Artificial Intelligence**]: Robotics; I.2.m [**Artificial Intelligence**]: Miscellaneous—*Imitation, Programming by demonstration*

General Terms

Measurement, Performance

HRI'06, March 2–4, 2006, Salt Lake City, Utah, USA.

Keywords

Imitation and social learning, human-robot interaction, programming by demonstration.

1. INTRODUCTION

Having a robot observe and learn to perform a task from an experienced teacher presents a more flexible and adaptive solution than explicit pre-programming or restrictive hardwiring. Robotics researchers are inspired by imitation and social learning in animals and humans to create controllers for their autonomous robots. The major directions of work have been using such behaviours with humans for adaptive learning [8, 12, 6, 3, 7] and also allowing the robots to be able to engage in richer, more natural social interactions [5, 11, 4].

A fundamental problem when learning *how* to imitate is to create an appropriate (partial) mapping between the actions afforded by particular embodiments to achieve corresponding states and effects by the model and imitator agents – solving a *correspondence problem* [9]. The related problem of *what* to imitate is addressed by the choice of metrics and sub-goal granularity that should be used for generating imitative behaviour, depending on the context [10].

Our work results from the Cogniron project which investigates the development of a robot companion for a domestic scenario. A robotic companion at home could for example acquire knowledge of arranging some household objects on a table from observing its human owner. Acquiring such skills socially requires matching different aspects of the effects that the human actions have on objects in the environment, e.g. matching the relative positioning and orientations of plates and silverware in a dinner setting. Also the various contexts within which a task is replicated might require its generalization to various settings and to other types and shapes of manipulated objects. We examine how the performance of such a robot (able to imitate using a system like JABBERWOCKY described in the next section) could be quantitatively evaluated from the robot's perspective and how this could be compared with a qualitative evaluation from the human's perspective. The results from a series of experimental runs and a pilot user study are presented in this paper and their implications for the design of robots that imitate are discussed.

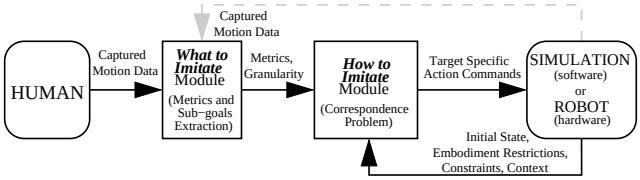


Figure 1: The JABBERWOCKY system architecture. Using data captured from a human and given appropriate metrics and sub-goal granularity, the multi-target system can produce an action command sequence plan that when executed by a software or hardware agent can achieve corresponding actions, states and/or effects. The corresponding actions, states and effects as demonstrated by the imitator can also be captured and used as a demonstration for another imitating agent. Differently embodied and constrained target systems in various contexts are supported.

2. THE JABBERWOCKY SYSTEM

To address the correspondence problem in imitation of humans in social context, we are currently developing JABBERWOCKY, a system that can use captured data from a human demonstrator to generate appropriate action commands to achieve a matching behaviour (see Figure 1). The action commands can be targeted for various robotic simulation (software) and physical (hardware) platforms. These actions allow the imitating agent to achieve corresponding actions, states and/or effects, depending on the given metrics and granularity (relevant to the demonstrated task and context), embodiment restrictions and constraints (imposed by the targeted imitator platform), and possibly different initial configuration of the objects in the environment. A *how to imitate* module uses the metrics and sub-goal granularity provided by a *what to imitate module*, plus the initial state, embodiment restrictions and constrains of the imitator, to generate appropriate action commands that if executed by the targeted imitator platform will result in corresponding actions, states and/or effects.

The current implementation of the JABBERWOCKY system focuses on solving the correspondence problem for the *effects* behavioural aspect (for example, displacement and rotation of 'block' objects on a two-dimensional workspace according to given *effect metrics* - see next section) and generates a *plan* according to which the imitator (which is assumed capable of manipulating the objects on its own) can successfully achieve an imitative behaviour based on a demonstration by a human. For a more detailed description of the JABBERWOCKY system see [1, 2].

3. EFFECT METRICS

Towards a characterization of the *space of effect metrics*, i.e. those that relate to the manipulation of objects (rather than, say, body postures or limb movements), we have explored absolute/relative angle and displacement aspects and focused on overall arrangement and the trajectory of manipulated objects [2]. Focusing on aspects of orientation and displacement of the manipulated objects, two types of effect metrics can be used, *displacement* and *angular*. The

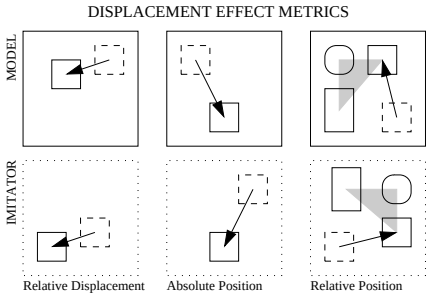


Figure 2: A selection of *displacement* effect metrics. To measure the discrepancy between object displacements, the *relative displacement*, *absolute position* or *relative position* effect metrics can be used. The first row shows three examples of effects demonstrated by the model. The second row shows the way the corresponding object (in a different workspace) needs to be moved (from dashed to solid outline) by an imitator to match the corresponding effects according to each metric. The grey triangles are superimposed to show that for the *relative position* effect metric, the relative final positions of the objects are the same.

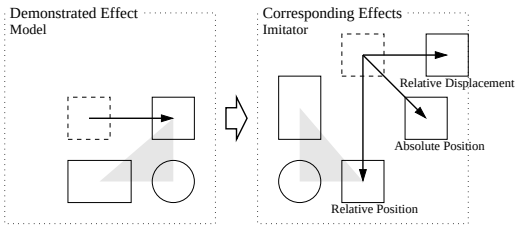


Figure 3: Depending on the *effect metric* used, qualitatively dissimilar imitative behaviours can result from dissimilar object configurations (here position only). The figure illustrates three examples of using different displacement effect metrics.

first type relates an object's movement and position in the workspace (e.g. *relative displacement*, *absolute position* or *relative position* on a table surface, see Figure 2), and the second type to the object's orientation. Using these metrics, one can evaluate the similarity between the *effects* on the environment (object displacement and/or rotation) of the model and the imitator, without considering the *state* or the *actions* of the agents that caused them (cf. [9]).

Some examples of circumstances where each of these metrics can be useful (in the context of setting up a dining table) would be placing a salad bowl or the main plate in the center of the table (absolute position), arranging the forks and knives next to the plates (relative position and orientation) or (having placed a set of plates, silverware and glasses at each seat) repeating the dining arrangement for each person (relative displacement and rotation).

If the objects start from the same positions in the imitator's workspace as in the demonstrator's workspace, all the displacement effect metrics become equivalent (i.e. using any of them, the same trajectories will be generated); similarly

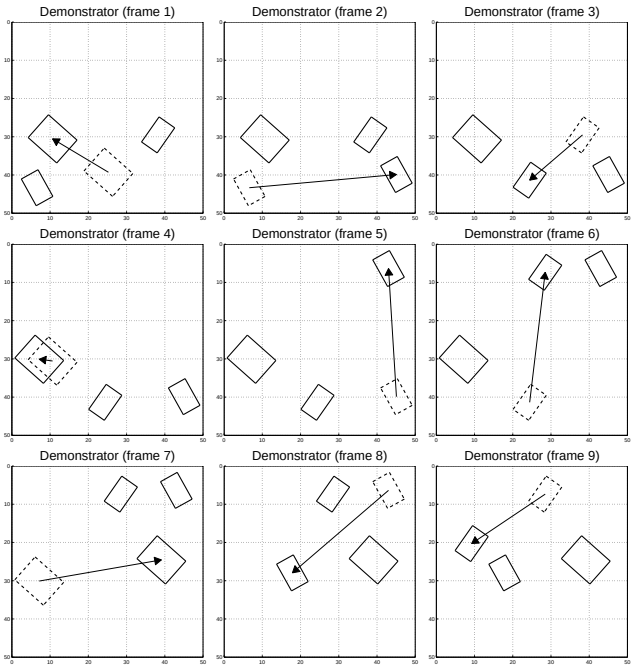


Figure 4: Three objects are manipulated in an example of a demonstration (generated, not captured from a human subject). The figure visualizes the effects of the demonstrated behaviour. At each frame (segmentation according to the sub-goal granularity used), the displacement of an object is shown (arrow) from its previous (dotted outline) to its current position (solid outline).

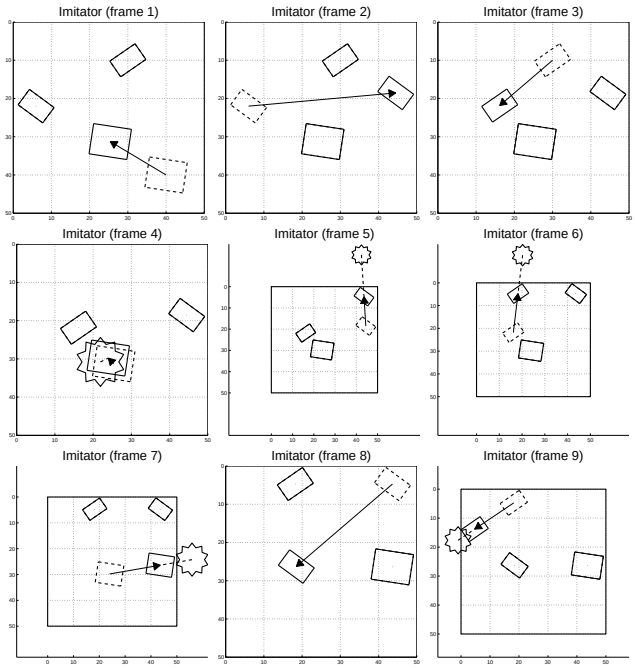


Figure 5: Imitative Behaviour based on the demonstration shown in Figure 4 and with the objects starting from a dissimilar initial configuration (positions and orientations), the JABBERWOCKY system generates a sequence of action commands for the appropriate imitative behaviour (assuming the same objects and workspace). The figure visualizes the resulting corresponding effects using the *relative displacement* effect metric for each of the three objects. In some of the frames, a star indicates an *exception*, i.e. that the displacement that minimizes the metric is not directly achievable because either the object would have to move outside workspace (frames 4, 6, 7 and 9) or is obstructed by another object (frame 4). In those cases, the generated action commands are modified accordingly (modification vector shown as a dotted line) so that the imitative behaviour can be carried out.

if the objects start in the same orientations all the angular effect metrics become equivalent (the objects rotate in the same way as in the demonstration). But if the objects start in a dissimilar initial configuration (positions and/or orientations) to that of the demonstration, the choice of metrics affects qualitatively the character of the resulting imitative behaviour (see Figure 3). See [2] for precise mathematical definitions of these and other effect metrics, as well as for the capacity to use them in generalizing imitative behaviours across different initial configurations.

4. MACHINE-CENTERED ASSESSMENT

Using the JABBERWOCKY system, we conducted a series of experimental runs to examine how the performance of the system using the various effect metrics could be quantitatively measured, from the system's perspective.

Depending on the context (initial object configuration, imitator embodiment restrictions and environment constraints), sometimes the displacement and/or rotation of an object (in order to minimize the metric used) is not directly achievable. In those cases, the system generates action commands that result instead in a modified displacement and/or orientation that can be carried out.[1] We call these events *exceptions*, and we can distinguish between two 'types':

- *type A exceptions* occur when an object, in order for the effect metric to be minimized, has to be moved and/or rotated in such a way that it would drift outside the imitator's workspace.

  The generated displacement and/or rotation action commands will keep the object inside the workspace.

- *type B exceptions* occur when an object, in order for the effect metric to be minimized, would have to be moved and/or rotated in such a way that the next subgoal is obstructed by the other objects in the workspace.

  The generated displacement and/or rotation action commands will either guide the object around the obstacle,

[1] In some cases, the only possible alternative might be for the imitator not to move/rotate the object at all and instead continue with the rest of the imitative behaviour, assuming that the problem would be resolved by the next time the object has to be handled (either because the obstructing objects would have been moved, or because the object would then be required to move/rotate to an unrestricted location).

or allow it to approach as close as possible to the target configuration.[2]

Exceptions occur when the metric used cannot be minimized and the system has to explore alternative solutions when generating action commands for an imitative attempt (given the particular context). [Whenever no type is specified in the text, the total number of exceptions (i.e. totaling both type A and B exceptions) is considered.]

4.1 Methodology

A series of experimental runs were conducted to examine if the number of exceptions could be used as a reasonable measure of the system's performance (from a robot's perspective), with fewer exceptions indicating a "better", more similar according to the effect metrics, generated imitative behaviour. Towards this goal we first experimentally characterize the exception profiles likely to occur with three different metrics. How well the machine-centered performance measures correspond to human judgment of similarity is studied in section 5.

The JABBERWOCKY system is able to generalize across different initial object configurations and effect metrics for a given demonstration. Using the same corresponding objects and workspace for both demonstrator and imitator, we generated 10 random demonstrations (see Figure 4 for an example of such a demonstration)[3], and for each of these, starting from 100 generated random initial object configurations (positions only), we used the JABBERWOCKY system to generate imitative behaviours for three of the displacement effect metrics (*absolute position*, *relative position* and *relative displacement*). Since we were only interested in characterizing the number of exceptions arising in the plans produced with various effect metrics, the generated action commands were not used in any target imitation platforms. For each of these simulated experimental runs, we counted the number of exceptions that occurred (see Figure 5 for an example of a generated imitative behaviour, with the occurring exceptions visually highlighted). In total, we performed 3000 experimental runs (3 effect metrics $\times$ 10 demonstrations $\times$ 100 initial configurations).

Note that we are *not* comparing between metrics to determine which metric is "better". All the metrics are needed and useful in different circumstances [1, 2]. On the contrary, the experiment is used to characterize the different exception profiles that arise in the course of using each of the different metrics.

4.2 Experimental Results

The experimental results are presented as frequency distributions of the number of exceptions in Figures 6 to 8.

In the experimental runs that the *absolute position* effect metric is used, the frequency distribution (Figure 6, left) reflects the fact that there were *no* type A exceptions and only a few type B exceptions. The absence of type A exceptions (Figure 6, center) can be attributed to the nature of the metric used - no corresponding position can be outside the (same as the demonstrator's) imitator's workspace. The low number of type B exceptions (Figure 6, right) can be explained by the fact that after all the objects have been moved once early in the generated imitative behaviour, they then occupy the same positions as in the demonstration, with the remaining imitative behaviour unfolding like an identical replay (which should not cause any exceptions).

When the *relative position* metric is used, a different frequency distribution can be observed (Figure 7, left). Compared to the *absolute position* histogram (Figure 6, center), in this case some type A exceptions do occur (Figure 7, center).

Compared to when the *relative position* metric is used (Figure 7, right), the lower number of type B exceptions (Figure 8, right) when the *relative displacement* metric is used produces another different frequency distribution (Figure 8, left). The frequency distributions for the type A exceptions are very similar when the *relative position* (Figure 7, center) and the *relative displacement* (Figure 8, center) metrics are used. The differentiating factor between the *relative position* and the *relative displacement* is the frequency distributions for the type B exceptions. Using the *relative displacement* metric results mostly in type A exceptions (because e.g. in the demonstration the object position was not as close to the workspace edges as in the imitative behaviour), while using the *relative position* metric results mostly in type B exceptions (because e.g. the other objects are obstructing the path of the object towards the corresponding relative position). The frequency distributions in Figures 8 (left) and 7 (left) are mostly influenced by the type A (Figure 8, right) and type B (Figure 7, right) exceptions, respectively.

Overall, the results indicate that if the *absolute position* metric is used, no type A and a low number of type B exceptions can be expected. If the *relative position* metric is used, a higher number of both types of exceptions can be expected. The profile for the *relative displacement* is similar to the latter in terms of type A exceptions but exhibits fewer type B exceptions. But note that, although this performance measure seems to imply that using the *absolute position* effect metric results in the "best" performance, drawing such a conclusion would miss the point that in general there is no best, task-independent metric. The choice (and combination) of metric(s) to be used depends on the particular demonstrated behaviour and should be automatically determined by the system (together with the sub-goal granularity) from a wide variety of appropriate metrics (see sections 2 and 3).

5. HUMAN-CENTERED ASSESSMENT

The number of exceptions can be used by a robot to self-evaluate its imitation performance, but in most cases some exceptions will unavoidably occur during the generation of the imitative behaviour, given the particular context and the choice of metric. We conducted a small pilot user study to examine how subjects (from a human perspective) would qualitatively evaluate a selection of imitative behaviours generated by the JABBERWOCKY system, compared to the quantitative evaluation (from the system's perspective). The system-centered evaluation was based on the number

[2]These ways of dealing with type A and B exceptions are convenient for this scenario. In more complex scenarios involving interference from outside or self-initiated movement by objects, different strategies might sometimes also work, e.g. waiting for an obstacle to be removed or move away.

[3]The demonstrations were artificially generated instead of captured from a human user, since we required the behaviours to be random, of the same duration, and to be similarly segmented in terms of sub-goal granularity.

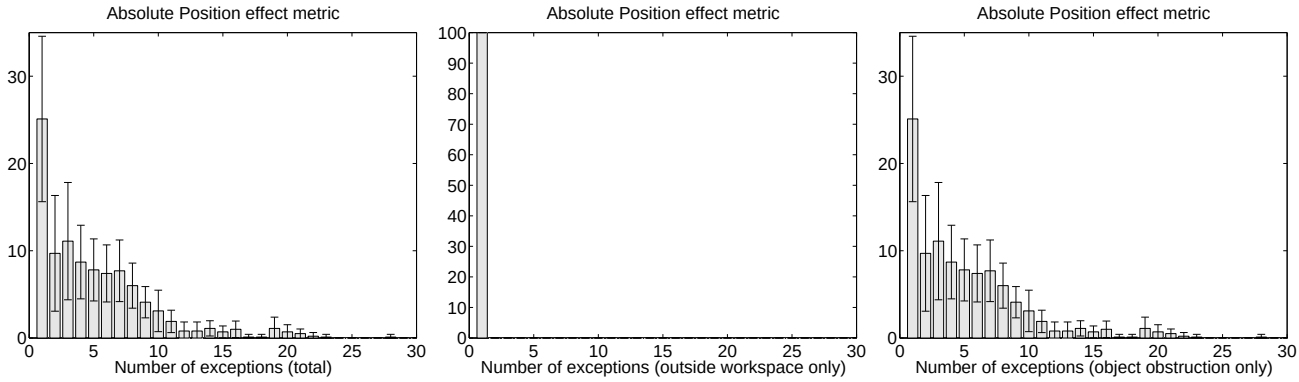


Figure 6: Frequency distributions for the number of exceptions that occurred when the imitative behaviours were generated using the *absolute position* effect metric. The bars show the mean number of experimental runs and the vertical lines indicate the standard deviation (from the 10 different random demonstrations batch runs). For each of those batch runs, 100 experimental runs were conducted, using different random dissimilar initial object configurations. The histogram on the left shows the frequency distribution for the total number of exceptions (at each run), while the histograms center and right show the frequency distributions using only the exceptions that occurred when an object had to move outside the workspace (type A) or was obstructed by other objects (type B), respectively.

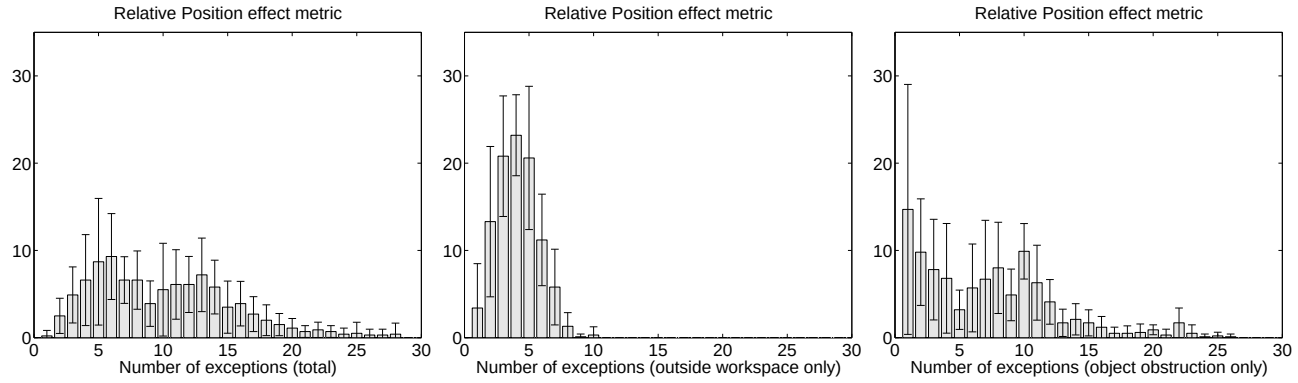


Figure 7: Frequency distributions for the number of exceptions that occurred when the imitative behaviours were generated using the *relative position* effect metric (visualization as in Figure 6).

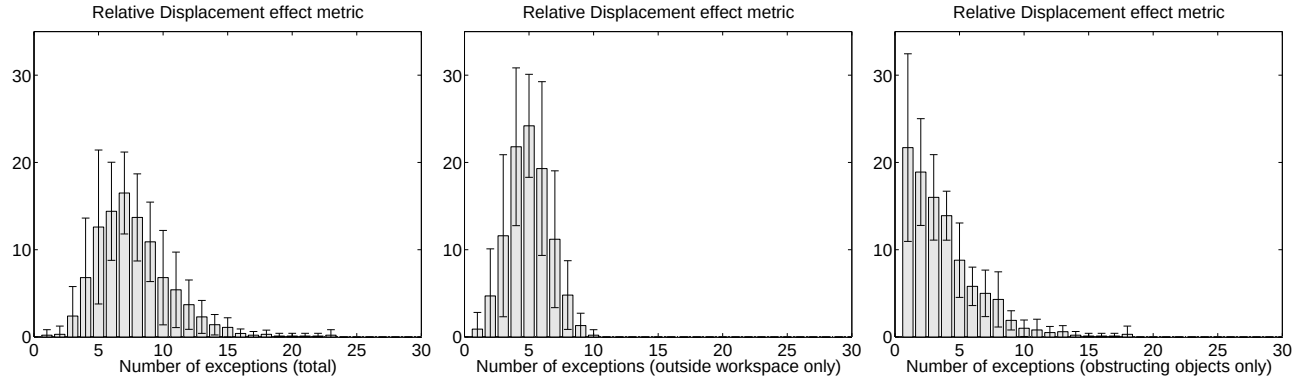


Figure 8: Frequency distributions for the number of exceptions that occurred when the imitative behaviours were generated system using the *relative displacement* effect metric (visualization as in Figure 6).

Sample Characteristics (N = 12)	
Gender	
Female	17%
Male	83%
Age	
< 25	17%
$26 - 35$	58%
> 36	25%

Table 1: Sample Characteristics. All subjects were recruited from University of Hertfordshire and were either Ph.D. students or researchers, with a technological background.

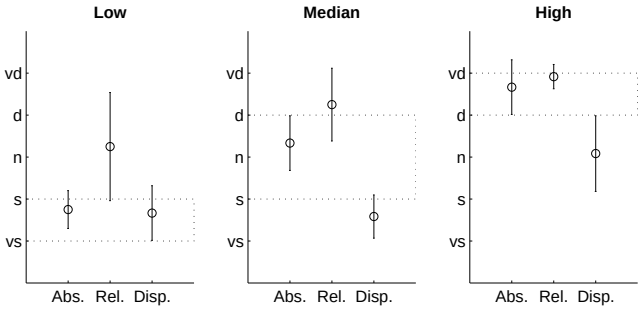


Figure 9: Qualitative human assessment of imitative behaviours. For each of the nine examples in the user study (broken down in terms of number of exceptions). The nine examples are grouped in three subplots in terms of *low*, *median* or *high* number of exceptions, each subplot showing the average answer on the Likert scale used (*vs* = very similar, *s* = similar, *n* = neutral, *d* = dissimilar, *vd* = very dissimilar), for each of the effect metrics used (*Abs.* = absolute position, *Rel.* = relative position, *Disp.* = relative displacement). The circles indicate the average result from the twelve subjects and the vertical lines the standard deviation. The boxed areas in each subplot visualize the hypothesis (see text).

of exceptions (as described in the previous section, with a low number indicating a successful imitation attempt). The human-centered evaluation used a Likert scale.

5.1 User Study Methodology

The subjects (see sample characteristics in Table 1) were shown different imitative behaviour examples (based on the same demonstration) and were asked to evaluate how similar or dissimilar they felt each attempt was to the demonstration. Both the demonstration and the imitative behaviours were given to the subjects as printed visualizations, similar to Figures 4 and 5. The process of generating the imitation attempts was not explained to the subjects. Only the effects and the objects were drawn, not the exceptions. The printouts were in colour with a unique colour identifying each of the three manipulated objects.

For each of three displacement effect metrics (*absolute position*, *relative position*, *relative displacement*), three imitative behaviours with 'low', 'median' and 'high' number of

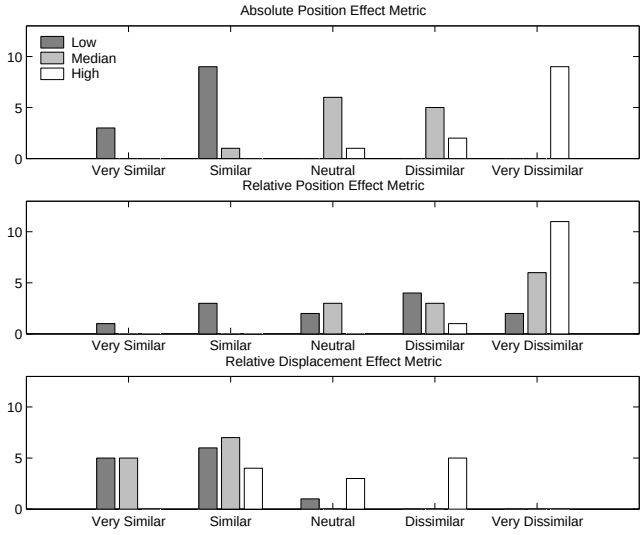


Figure 10: Relationship between exceptions and human-assessed similarity. Frequency distributions for the user similarity judgment on imitative behaviours are shown, broken down in terms of the effect metric used. Grouped in three subplots in terms of *Absolute Position*, *Relative Position* or *Relative Displacement* effect metric, each subplot shows the answer counts on the Likert scale used (*Very Similar* to *Very Dissimilar*), for each of the three number-of-exceptions categories (*low*, *median* and *high*).

exceptions were included (total of nine examples). The particular examples were taken from the experimental results presented in the previous section, with the number of exceptions related to the frequency distributions shown in Figures 6 to 8 ('low' was zero or one, 'median' was the median and 'high' well-above the median).

The subjects were not told that different metrics were used, or that for each metric three cases (according to number of exceptions) were included. They were only told that in each example, although the demonstration is the same for all of them, the initial configuration (positions) of the objects is different. The order of the examples was randomized but the sequence was kept the same for all subjects (without previews of subsequent examples). Following the completion of the evaluations, the subjects were also asked to provide us with some comments on how they made their choices in a brief informal interview.

Using a five-point Likert scale (*very similar*, *similar*, *neutral*, *dissimilar*, *very dissimilar*), our hypothesis was that the subjects would evaluate the imitation examples with:

- low number of exceptions as *similar* to *very similar*,
- median number of exceptions as mostly *neutral* (ranging from *similar* to *dissimilar*) and
- high number of exceptions as *dissimilar* to *very dissimilar*.

5.2 User Study Results

The average values of the answers in qualitative human assessment of similarity of the imitative behaviours from the

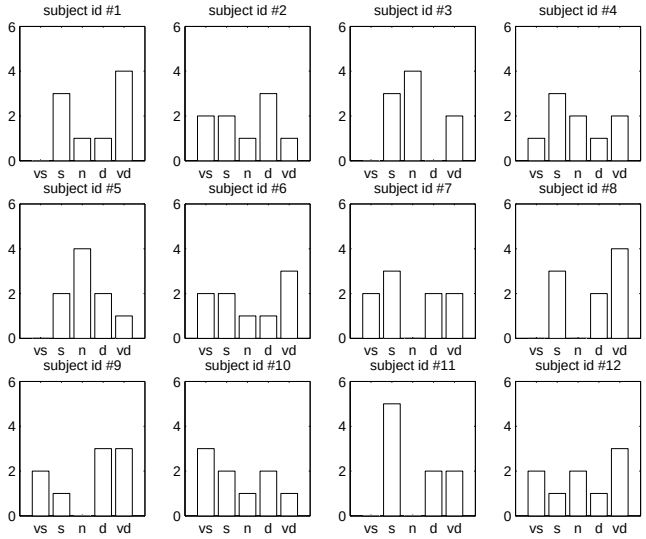


Figure 11: Frequency distribution of the answers for each of the twelve subjects of the user study. (*vs* = very similar, *s* = similar, *n* = neutral, *d* = dissimilar, *vd* = very dissimilar)

user study are shown in Figure 9. Our hypothesis (visualized by the boxed areas in each subplot) is mostly confirmed for the *absolute position* and less for the *relative position* and *relative displacement* effect metric examples. The frequency distributions shown in Figure 10 also indicate that most subjects (overall) evaluated the *low number of exceptions* examples as mostly similar (i.e. the imitation attempts more successful) and the *high number of exceptions* examples as mostly dissimilar to the demonstration. For all three metrics perceived dissimilarity tended to increase with increased number of exceptions. This is an early indication that the performance of a system can be quantitatively evaluated based on the number of exceptions (lower number of exceptions corresponding to a 'better' generated imitating behaviour).

The frequency distribution of the answers for each of the twelve participating subjects are shown in Figure 11.

Although the hypothesis was confirmed overall, differences depending on the effect metric that was used in the individual examples can be identified. Looking at Figure 9, the answers for the *absolute position* examples are a close fit to the hypothesis. In contrast, for the *relative position* examples the answers are predominantly in the range of *neutral* to *very dissimilar*, while for the *relative displacement* examples are mainly in the range of *very similar* to *neutral*. A similar impression can be formed by examining Figure 10.

The interviews at the end of each session provided us with some indications that the subjects while evaluating the imitative behaviours were considering only a single measure, and did not recognize that multiple metrics could be used. The majority said that they focused either on the object displacements (related to the *relative displacement* effect metric) or the object positions (related to the *absolute position* effect metric). A few subjects acknowledged that the arrangement of the objects (related to the *relative position* effect metric) might also be important.

Both the demonstration and imitative behaviour examples used in the user study consisted only of displacement effects, with the objects retaining the same orientation throughout as in their random initial configurations. Many subjects commented that they expected the orientation to be matched (to the one in the demonstration), and in some particular cases this affected their assessment (marking the effects as *similar* instead of *very similar*). Related to these comments, many subjects additionally commented that they found the dissimilar initial positions disorienting, making them unable in some cases to properly evaluate the imitation attempts.

6. DISCUSSION AND FUTURE WORK

The number of *exceptions*, events when the displacements and/or rotations that would satisfy the effect metric(s) used to generate corresponding imitative behaviours cannot be directly achieved in the particular context, can serve as a quantitative performance measure for systems generating imitation attempts. A low number of exceptions would indicate that the system was able to produce an imitative behaviour mostly similar (according to the metrics used) for a given context, while a high number would indicate the contrary. The experimental results presented in section 4.2 (and also similar experimental runs not presented in this paper) seem to indicate that, depending on the different effect metrics used, distinct frequency distributions of exceptions can be identified.

The results from the user study indicate that there is a good alignment between this quantitative performance measure (from the system's perspective) and the way subjects qualitatively evaluate (from a human perspective) the imitation attempts. The choice of effect metrics used to generate the imitative behaviours is shown to affect the subject's assessment of the similarity to the demonstration.

Further user studies (with a bigger sample) need to be conducted, building upon the results of the pilot, examining how humans evaluate the performance of robots capable of learning by imitation in a teaching scenario. The demonstration and the imitation attempts were provided here as printed material, without a meaningful context or any interaction of the subjects with the actual system. The demonstrations in the future studies should be performed by the subjects, allowing them to actively engage with meaningful tasks (that require different types of matching) and to have explicit personal expectations for the generated imitative behaviours. Similarly, the generated imitative behaviours based on the demonstrations (captured from the subjects) should be realized by either (initially) a software platform (with a visualizing animation displayed on a screen) or (eventually) by a hardware robotic platform (arranging corresponding physical objects in a corresponding workspace).

7. ACKNOWLEDGMENTS

The work described here was conducted within the EU Integrated Project COGNIRON ("The Cognitive Robot Companion") and was funded by the European Commission Division FP6-IST Future and Emerging Technologies under Contract FP6-002020.

8. REFERENCES

[1] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders. Achieving corresponding effects on multiple robotic platforms: Imitating using different effect metrics. In *Proc. Third International Symposium on Imitation in Animals and Artifacts – Hatfield, UK, 12-14 April 2005*, pages 10–19. Society for the Study of Artificial Intelligence and Simulation of Behaviour, 2005.

[2] A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, and J. Saunders. An approach for programming robots by demonstration to manipulate objects: Considerations on metrics to achieve corresponding effects. In *Proc. 6th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '05)*, pages 61–66, 2005.

[3] A. Billard, Y. Epars, S. Calinon, G. Cheng, and S. Schaal. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47:2-3, 2004.

[4] C. L. Breazeal, D. Buchsbaum, J. Gray, D. Gatenby, and B. Blumberg. Learning from and about others: Towards using imitation to bootstrap the social understanding of others by robots. *Artificial Life*, 11(1–2):31–62, 2005.

[5] K. Dautenhahn. Getting to know each other – artificial social intelligence for autonomous robots. *Robotics and Autonomous Systems*, 16:333–356, 1995.

[6] Y. Demiris and M. Johnson. Distributed, predictive perception of actions: a biologically inspired robotics architecture for imitation and learning. *Connection Science Journal*, 15(4):231–243, 2003.

[7] R. Dillmann. Teaching and learning of robot tasks via observation of human performance. *Journal of Robotics & Autonomous Systems*, 47(2–3):109–116, 2004.

[8] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observations of human performance. *IEEE Trans. Robot. Automat.*, 10:799–822, November 1994.

[9] C. L. Nehaniv and K. Dautenhahn. Mapping between dissimilar bodies: Affordances and the algebraic foundations of imitation. In J. Demiris and A. Birk, editors, *Proceedings European Workshop on Learning Robots 1998 (EWLR-7), Edinburgh, 20 July 1998*, pages 64–72, 1998.

[10] C. L. Nehaniv and K. Dautenhahn. Like me? - measures of correspondence and imitation. *Cybernetics and Systems*, 32(1-2):11–51, 2001.

[11] M. N. Nicolescu and M. M. Matarić. Learning and interacting in human-robot domains. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 31(5):419–430, 2001.

[12] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences*, 3(6):233–242, 1999.

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.

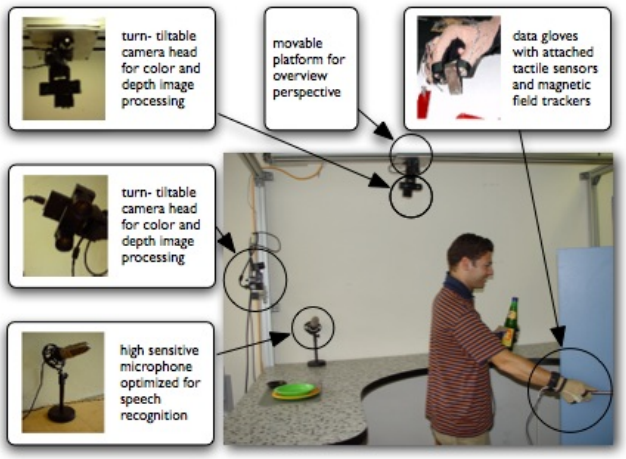


Fig. 1. Training center with dedicated sensors

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.

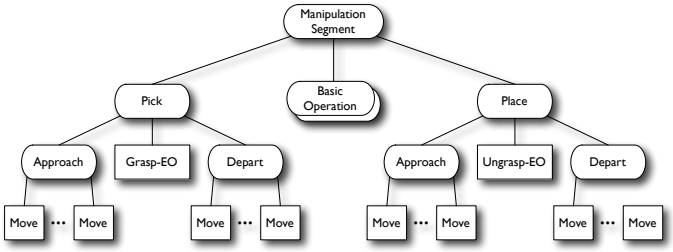


Fig. 2. Hierarchical representation of manipulation segments

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 (pour-

ing 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.

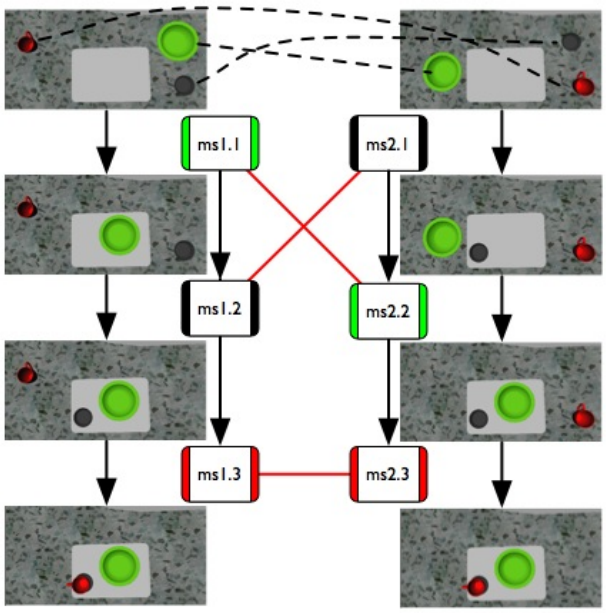


Fig. 3. Recognition of corresponding task features for the first experiment. Object correspondences are drawn with stipled, subtask correspondences with solid lines. The sequential ordering of the subtasks is represented by bold arrows.

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.

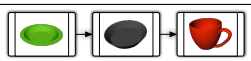
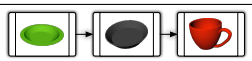
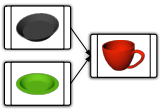
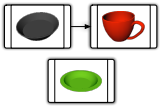
i	P^{D_i}	P_i
1		
2		
3		

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.

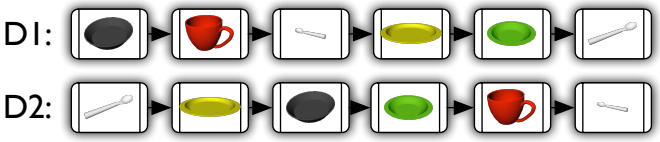


Fig. 5. The two demonstrations performed for the second experiment.

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.

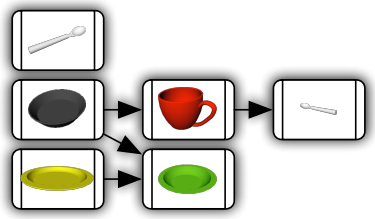


Fig. 6. TPG learned for complex laying the table task in the second experiment. This final TPG is learnt after taking into account only two demonstrations.

[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.

Incremental Learning of Task Sequences with Information-Theoretic Metrics

M. Pardowitz, R. Zöllner and R. Dillmann

Institute of Computer Science and Engineering
Universität Karlsruhe (TH)
Karlsruhe, D-76138, Germany, P.O. Box 6980
Email: {pardow, zoellner, dillmann}@ira.uka.de

Summary. Learning tasks from human demonstration is a core feature for household service robots. Task knowledge should at the same time encode the constraints of a task while leaving as much flexibility for optimized reproduction at execution time. This raises the question, which features of a task are the constraining or relevant ones both for execution of and reasoning over the task knowledge.

In this paper, a system to record and interpret demonstrations of household tasks is presented. A measure for the assessment of information content of task features is introduced. This measure for the relevence of certain features relies both on general background knowledge as well as task-specific knowledge gathered from the user demonstrations. The results of the incremental growth of the task knowledge when more task demonstrations become available is demonstrated within the task of laying a table.

1 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 task knowledge the user has and transform it into knowledge utilizable by a humanoid service robot. 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)*.

PbD-systems should be capable of learning a task from a single demonstration sufficiently for execution while being able to generalize it when further demonstrations of the same task become available. Incremental learning approaches that gradually refine task knowledge 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.

So the task knowledge must explicitly encode those in order to ensure a reliable, faultless and save 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, the incorrect hypotheses conflicting with the new demonstrations can be pruned.

Identifying the corresponding subtasks performed in different demonstrations of the same task is not an easy job. Many features of the task are exposed to noise. Additionally, some features posess higher relevance to a certain task, while being negligible for others. A method for identifying the features that characterize a task and basing the recognition of corresponding subtasks on this selection is presented in this paper.

The next section gives an overview on related work concerning programming by demonstration and task learning from user demonstrations. Section 3 describes the system for the acquisition of task knowledge from a single user demonstration. Section 4 introduces task precedence graphs and proposes a method for learning 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 5 are an important preliminary for this computation. Finally, the methods desscribed in earlier sections are evaluated in section 6.

2 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 [10]. Systems can be discriminated by the learning paradigm applied.

Imitation learning systems are used to re-identify and execute human motions [2], aiming at a high similarity of the robot's movements to the demonstrated trajectories.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 [9, 11], 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 [3, 6]. 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.

3 A System to Generate Hierarchical Task Descriptions

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 of our system.

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. Further details on the hardware used can be found in [4].

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). On the highest level, a sequence of manipulation segments is subsumed under a whole task demonstration (see figure 1). For further details, please refer to [11].

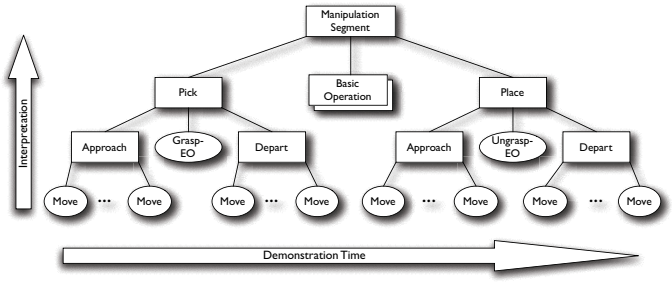


Fig. 1. Hierarchical representation of manipulation segments

On each level in this hierarchical task representation, the changes to the environment are induced from lower levels of the hierarchy. The pre- and post-conditions 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 ("fork", "plate", "table") and internal state descriptors ("opening angle", "oven temperature", "liquid level", etc., depending on the object class).

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.

4 Learning Task Precedence Graphs

This section is concerned with the representation of the sequential features of a task, the task precedence graph (TPG), and how it can be learned incrementally from multiple demonstrations of this task.

When a user performs a task, he performs it's 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. A learning system that builds knowledge about a task has to hypothesize on the underlying sequential task structure. These hypotheses can be represented by task precedence graphs, introduced by [7].

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 $o_1 \rightarrow o_2$.*

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. The learning system can state different equally valid hypotheses, ranging from the most restricting precedence graph $P^D = (\mathbf{N}, \mathbf{R}^D)$ that only allows the execution order demonstrated, to the most liberal TPG that posesses no sequential dependencies at all and allows the robot to choose the sequence of operations without any constraints. [7] states that the set of valid hypotheses is a version space, partially ordered by the subset-predicate on the sets of precedence relations $\mathbf{R}$.

Assuming that, after processing m demonstrations $\{D_1, D_2, \ldots, D_m\}$ of the same task, the system has learned the most restrictive TPG P_m. Under

observation of a new demonstration D_{m+1}, the learned task precedence graph can be adapted in a way that incorporates the new knowledge. [5] suggests that this can be done by further generalising the previous hypothesis to a new one, covering the additional example. In order not to generalize too far, the minimal generalization of P_m that is consistent with D_{m+1} must be chosen. The resulting 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 $P^{D_{m+1}} = (\mathbf{N}, \mathbf{R}^{D_{m+1}})$:

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

With the mechanisms presented in this section it is possible to incrementally learn task precedence graphs from user demonstrations. The process of refining task precedence graphs starts with the most restrictive hypothesis (= the order seen in the first demonstrations) and continues to become more and more general as more demonstrations with different sequences of actions are taken into account.

5 Subtask Similarity Measures Based on Information-Theoretic Metrics

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 recognize the matching manipulation segments, two sets of features are taken into account:

- *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.

The base operation to assess feature conformance is the measure of similarity s_F for two sets of features $\mathbf{A}, \mathbf{B}$ with the portion of common features in both feature sets:

$$s_F(\mathbf{A}, \mathbf{B}) = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A} \cup \mathbf{B}|}. \tag{2}$$

It turned out that this straight-forward approach exposes one major drawback: As the number of features is relatively high and many of the features are

irrelevant to the certain task to be learned, the features carrying the relevant information will be predominated by the irrelevant ones. The solution is to weight each feature f depending of it's relevance to the task to be learned with a weight $w(f)$. This turns equation 2 into

$$s_F(\mathbf{A}, \mathbf{B}) = \frac{\sum_{f \in \mathbf{A} \cap \mathbf{B}} w(f)}{\sum_{f \in \mathbf{A} \cup \mathbf{B}} w(f)}. \tag{3}$$

When choosing a useful weight function $w(f)$ one has to consider two points:

Features of high relevance to the specific task should gain high weights. One way to assess the relevance of a feature is to test its occurrence over several different demonstrations of the same task. Features with a high relevance to the task to be performed have a great probability of occurrence, while features of lower relevance will occurr less frequently. According to Shannon, the information content $I(f)$ of a feature f with probability $p(f)$ is $I(f) = -\log_2 p(f)$. As features with low information content (= high probability of occurrence) to the specific task class $\mathbf{T}$ should be favored, $w(f) = -\log_2(1 - p(f|\mathbf{T}))$ seems a reasonable choice for the weight function.

On the other hand, when a completely new task is learned or only few demonstrations of the same task are available one has to take into account that only little information about the distribution of features in the specific task class is known. The idea is to introduce global background knowledge on the feature distribution. When several demonstrations of different tasks have been observed by the robot and incorporated into the task knowledge base, it can be assumed that the features that uniquely discriminate the task are those that have low occurrence rates accross the whole task knowledge base. Thus, the information content to all other tasks $\overline{\mathbf{T}}$ of feature f is $-\log_2 p(f|\overline{\mathbf{T}})$.

An incremental learning system should be able to cope with both situations. Therefore a combination has to be found that favors the global information content measure when no or few task demonstrations are available, and the task-specific relevance measure as more demonstrations of the specific task become available. The weight function that fulfills those requirements is

$$w(f) = -\left[\alpha \log_2 (1 - p(f|\mathbf{T})) + (1 - \alpha) \log_2 p(f|\overline{\mathbf{T}})\right] \tag{4}$$

with the relative weighting α of the task-specific knowledge as $\alpha = 1 - e^{-k \cdot |\mathbf{T}|}$. In our experiments we chose $k = -\frac{\ln 0.25}{5}$, such that after observing 5 demonstrations of the same task, the proportion α of task-specific to prior-knowledge is $0.75 : 0.25$ and converging towards 1 for more demonstrations. This choice is motivated by [1], stating that the important features of a task can be sufficiently learned with $4 - 5$ demonstrations.

With the weighted proportion of features s_F as in eq. 3 it is possible to compute the similarities s_{pre} and s_{post} of the pre- and postcondition sets of two subtasks M_1 and M_2 and the average of the subtask similarities $\overline{s_{sub}}$of the tasks. They are set to

$$s_{pre} = s_F(Precond(M_1), Precond(M_2))$$
$$s_{post} = s_F(Postcond(M_1), Postcond(M_2))$$
$$\overline{s_{sub}} = \frac{1}{|M_1 \cup M_2|} \sum_{m_1 \in M_1, m_2 \in M_2} s_M(m_1, m_2).$$

The (recursive) overall similarity s_M of the two manipulation segments M_1, M_2 is defined as the simple average of the pre- and postcondition similarity and the average of all subtask similarities: $s_M(M_1, M_2) = \frac{1}{3} \cdot (s_{pre} + s_{post} + \overline{s_{sub}})$. 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 permutation that maximizes the 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} 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 4.

6 Experimental Results

In this section the experiments validating the method for incremental acquisition of task knowledge using heuristic relevance estimates for task features are reported. The task to be learned was chosen from the household domain; it consisted of laying the table for a main dish. The manipulated objects were: a plate, a fork and a mug. The main features that should be learned by the system are the spatial arrangement of the objects (the fork should be placed to the right of the plate and the mug placed on the distant side of the fork) as well as the sequential independence of all subtasks.

Eight different demonstrations of that task were given ($\mathbf{T} = \{D_1, D_2, \cdots, D_8\}$). In the first four of them, the fork, the plate and the mug were manipulated in this order. In the fifth and sixth demonstration the mug was placed first, followed by the fork and the plate. In the final two demonstrations the mug, the plate and the fork were manipulated in this ordering.

After taking into account the first example, the initial task precedence graph $P^{D_1} = P_1$ was correctly obtained, which can be seen in the first column of table 1. After observing a second example, D_2, the relevance estimates for the task similarity measure were continously taken into account. The relevance estimates for several features depending on the number of task demonstrations are depicted in figure 2(a).

One can see that features 1-4 (the Relations *mug ON table*, *fork ALIGNED_EAST mug*, *plate ON table*, *fork ON plate*) increase their relevance estimate as more task demonstrations become available. Meanwhile features

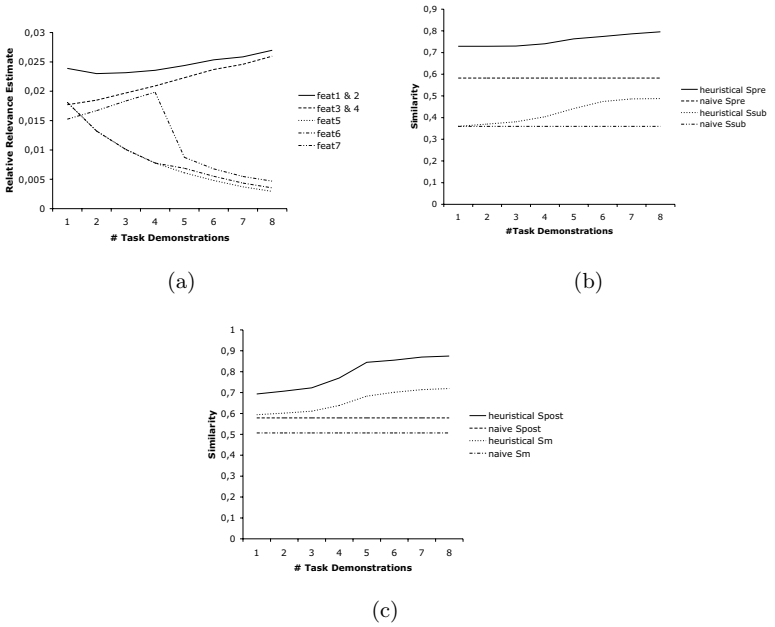


Fig. 2. (a) Relevance estimates for several features, depending on the number of tasks demonstrations taken into account.
(b) & (c) resp.: Precondition-, Subtask-, Postcondition- and Overall-Similarity for the demonstrations D_1 and D_5, calculated according to formula 3 (heuristic similarity) and 2 (naive similarity)

5-7 (the Relations *fork TOUCHES_SOUTH plate*, *mug TOUCHES_NORTH plate* and *mug INTERSECTING fork*), occurring unintentionally in some task demonstrations, receive lesser ratings as they tend to occurr less frequently in all task demonstrations. With the assessment of feature relevance, the pre- and post-condition similarity s_{pre} resp. s_{post} are calculated as well as the overall task similarity measure for all subtasks $\overline{s_{sub}}$, and finally the overall task similarity measure s_M. In figures 2(b) & (c) , the results for the comparision for the task demonstrations D_1 and D_5 are shown in dependence of the number of task demonstrations regarded for the heuristical relevance estimation.

The figures suggest the following: First, the heuristical measures start with a far better similarity than the naive approach. This is due to the second term in formula 4, which favours features that discriminate the new task from all other previously seen tasks. Second, the similarity measure gains higher values with more task-specific data becoming available. With more demostrations of the same task, the increasing weight α favors the first part of equation 4. As the

task feature statistics also become more striking with more demonstrations, this part ensures a task-specific adaption of the relevance measures.

After the system has seen four examples with all the same order of actions, the learned task precedence graph is unchanged. After that, the system observes a demonstration featuring a different execution order. The two different sequences and the recognized subtask permutation are shown in figure 3. The task precedence graph learned after five demonstrations is shown in the second column of table 1. When the last two user demonstrations are observed, the system can eliminate the last unnecessary precedence relation between the fork and the plate and is free to schedule every subtask as it is most convenient during the execution.

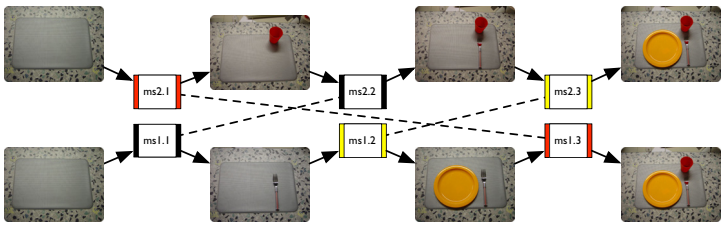
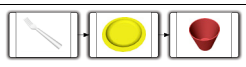
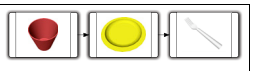
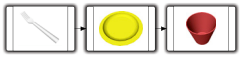
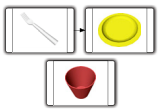
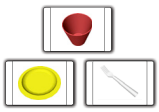


Fig. 3. Recognition of equal subtasks. Subtask correspondences are drawn with stipled lines. The sequential odering of subtasks is represented by bold arrows.

Table 1. Task precedence graphs learned incrementally during the performance of the experiment described in section 6.

i	1-4	5-6	7-8
P^{D_i}			
P_i			

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

Methods based on information-theoretic approaches are presented that estimate the relevance of task features. These heuristics combine pre-existing background knowledge on feature distributions across different tasks with learned knowledge about the task itself.

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.

References

1. Aude Billard, Yann Epars, Sylvain Calinon, Stefan Schaal, and Gordon Cheng. Discovering optimal imitation strategies. *Robotics and Autonomous Systems*, 47(2-3):69–77, 2004.
2. S. Calinon and A. Billard. *Learning of Gestures by Imitation in a Humanoid Robot*, chapter To appear, page In Press. Cambridge University Press, 2005.
3. Jason Chen and Alex Zelinsky. Programming by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robots Research*, 22(5):299–319, May 2003.
4. 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.
5. Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.
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*, pages 241–248, Melbourne, Australia, July 2003.
7. M. Pardowitz, R. Zöllner, and R. Dillmann. Learning sequential constratins of tasks from user demonstrations. In *IEEE-RAS Intl. Conf. on Humanoid Robots (HUMANOIDS), Tsukuba, Japan*, 2005.
8. O. Rogalla. *Abbildung von Benutzerdemonstrationen auf variable Roboterkonfigurationen*. PhD thesis, University Karlsruhe, Departmen of Computer Science, 2002.
9. 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*, pages 1208 – 1213, 2002.
10. Stefan Schaal. Is imitation learning the rout to humanoid robots? In *Trends in Cognitive Sciences*, volume 3, pages 233–242, 1999.
11. 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.

Incremental Learning of Tasks from User Demonstrations, Past Experiences and Vocal Comments

M. Pardowitz, R. Zöllner, S. Knoop and R. Dillmann

Abstract— **Looking at robot assistants sharing the same environment with humans, it became obvious that they have to interact with humans and like humans also learn or at least adapt to individual human needs. So learning as a core feature for future robot assistants must follow the same paradigms like human learning. One of them is learning by demonstration, were the robot is supposed to observe the execution of a task, acquire task knowledge and reproduce it.**

In this paper, a system to record, interpret and reason over demonstrations of household tasks is presented. The focus is on the model based representation of manipulation tasks, which serves as a basis for reasoning over the acquired task knowledge. The aim of the reasoning is to condense and interconnect the knowledge.

A measure for the assessment of information content of task features is introduced. This measure for the relevance of certain features relies both on general background knowledge as well as task-specific knowledge gathered from the user demonstrations. Beside the autonomous information estimation of features, speech comments during the execution, pointing out the relevance of features are considered as well. The results of the incremental growth of the task knowledge when more task demonstrations become available and their fusion with relevance information gained from speech comments is demonstrated within the task of laying a table.

I. Introduction

During the last years, humanoid robotics became a major trend in the robotics community. Generally, the service robot design is dominated in all facets by the fact that in the future more and more robots are supposed to act in close cooperation with humans. In order to achieve a high acceptance by humans, besides of the humanoid appearance these robots have to behave like humans, in order to be accepted by humans. One of the most important properties of robot assistants is learning, since it enables robots to cope with everyday situations and natural (household ore office) environments in a way intuitive for humans. This means that through the learning process on the one hand new skills and tasks have to be acquired and on the other hand the knowledge of the system has to be adapted to new contexts and situations.

Robot assistants coping with these demands must be equipped with innate skills and should learn live-long from their users. Supposedly, these are no robot experts and require a system that adapts itself to their individual needs. For meeting these requirements, a new paradigm for teaching robots is defined to solve the problems of skill and task transfer from human (user) to robot, as a special way of knowledge transfer between man and machine.

The only approaches that satisfy the latter condition are programming systems that automatically acquire relevant information for task execution by observation and multimodal interaction. Obviously, systems providing such functionality require:

1) powerful sensor systems to gather as much information as possible by observing human behavior or processing explicit instructions like commands or comments,
2) a methodology to transform observed information for a specific task to a robot-independent and flexible knowledge structure, and
3) actuator systems using this knowledge structure to generate actions that will solve the acquired task in a specific target environment.

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. This allows the robot system to execute them in a sequence that can be optimized with respect to speed, energy consumption or path length. In order to learn these aspects from multiple demonstrations, common subtasks of different demonstrations of the same task have to be identified. This needs an explicit measurement of subtask similarity, that in turn relies on the similarity of features of each subtask. Different features have different importance attached. Here, an approach is proposed that allows to estimate the relevance of a feature, based on background knowledge, its occurrence probability in all demonstrations of a certain task and vocal comments the user can give while demonstrating the task to be learned by the system.

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 and the general representation of task knowledge, called macro operators. Section IV introduces task precedence graphs, the representation for the sequential structure of a task and 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 V are an important preliminary for this computation. Heuristical relevance estimates for task features based on the current task demonstrations, past experiences and spoken user comments are used to compute the similarity of tasks and subtasks and will be discussed in section VI section. Finally, the methods desscribed in earlier sections are evaluated in section VII.

II. Teaching Robot Assistants: An Overview

A classification of methods for teaching robot assistants in an intuitive way might be done according the type of knowledge which is supposed to be transferred between human and robot. Concerning performative actions, especially manipulation and navigation tasks are addressed. Here the knowledge needed for characterizing the actions can be categorized by the included symbolic and semantic information. The resulting classification, shown in figure 1, contains three classes of learning or teaching methods, namely: skill learning, task learning and interactive learning.

A. Skill Learning

For the term ”skill”, different definitions exist in literature. In this paper, it will be used as follows: A skill denotes an action (i.e. manipulation or navigation), which contains a close sensor-actuator coupling. In terms of representation, it denotes the smallest symbolic entity which is used for describing a task.

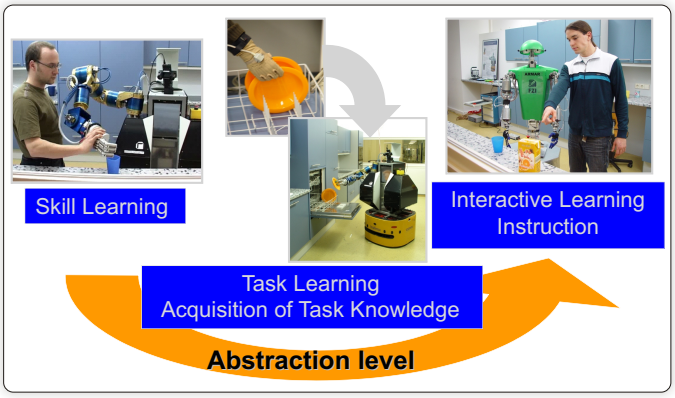


Fig. 1. Classes of skill transfer through interaction.

Examples for skills are grasps or moves of manipulated objects with constraints like ”move along a table surface” etc. The classical ”peg in hole” problem can be modeled as a skill, for example using a force controller which minimizes a cost function or a task consisting of a sequence of skills like ”find the hole”, ”set perpendicular position” and ”insert”.

Due to the close relation between sensors and actors the process of skill learning is mainly done by direct programming using the robot system and if necessary some special control devices. Skill learning means to find the transfer function R, which satisfy the equation $U(t) = R(X(t),\ Z(t))$. The model of the skill transfer process is visualized in figure 2

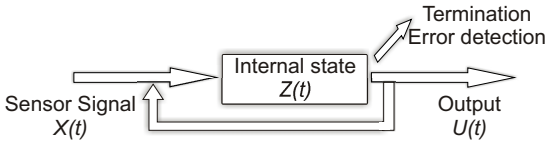


Fig. 2. Process of skill transfer.

As learning methods for skill acquisition often statistical methods like Neuronal Networks or Hidden Markov Models are used, in order to generalize a set of skill examples in terms of sensory input. Given a model of a skill for example through demonstration, reinforcement techniques can be applied to refine the model. Here a lot of background knowledge in form of an evaluation function which guides the optimization process has to be included into the system.

However, teaching skills to robot assistants is time-intensive and affords a lot of knowledge about the

system. Therefore, looking to robot assistants were the user of the systems is no expert, it seems more feasible that the majority of the needed skills are pre-learned. The system would have a set of innate skills and will only have to learn a few new skills during its "living time."

B. Task Learning

From the semantic point of view, a task denotes a more complex action type than a skill. From the representation and model side a task is usually seen as a sequence of skills, and that should serve as a definition for the following sections. In terms of knowledge representation a task denotes a more abstract view of actions, were the control mechanisms for actuators and sensors are enclosed in modules represented by symbols.

Learning of tasks differs from skill learning, since, apart from control strategies of sensors and actuators, the goal and subgoal of actions have to be considered. As in this case the action depends on the situation as well as on the related action domain, the learning methods used rely on a vast knowledge base. Further analytical learning seems to be more appropriate for coping with symbolic environment descriptions used to represent states and effects of skills within a task.

For modeling a task T as a sequence of skills (or primitives) S_i, the context including environmental information or constraints E_i and internal states of the robot system Z_i are introduced as pre- and post-conditions of the skills. Formally, a task can be described as:

$$T = \{(Pre - Cond(E_i, Z_i))S_i(\textit{Effect}(E_i, Z_i))\} \quad (1)$$
$$\textit{Effect} = Post - Cond \backslash Pre - Cond$$

were $\textit{Effect}()$ denotes the effect of a skill on the environment and the internal robot states. Consequently, the goal and subgoals of a task are represented by the effect of the task or the subsequence of the task.

Noticeable in the presented task model is the fact that tasks are state-, but not time-dependent like skills, and therefore enable a higher generalization in terms of learning. Although the sensory-motor skills in this representation are encapsulated in symbols, learning and planning task execution for robot assistants becomes very complex due to the fact that state descriptions of environments (including humans) and internal states are very complex, and decision making is a not always well-defined.

C. Interactive Learning

The process of interactive learning has the highest abstraction level because of the information exchange through speech and gestures. Through these interaction channels various situation-dependent symbols are used to refer to objects in the environment, or to select actions to be performed. Information, like control strategies on the sensory-motor level, can not be transferred any more, but rather can global parameters, which are used for triggering and adapting existing skills and tasks.

The difference between task learning and interactive learning on the one hand consists of the complexity of actions that can be learned and on the other hand the use of dialog during the learning process with explicit system demands. Beside communication issues, one of the main problems in developing robot assistants with such capabilities denotes the understanding and handling of "common sense" knowledge. In this context the problem of finding the same view of situations and communicating these between robot and human is still a huge research area.

For learning however, a state driven representation of actions which can either be skills, tasks or behaviors is required as well as a vast decision making module, which is able to disambiguate gathered information by initiating adequate dialogs. This process relies on a complex knowledge base including skill and task models, environment descriptions, and human robot interaction mechanisms. Learning in this context denotes finding and memorizing sequences of actions or important parameters for action execution or interaction. Associating semantic information to objects or actions for expanding space and action concepts is a further goal of interactive learning.

D. Related Work on Programming by Demonstration

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].

In the following some of the recent works on different learning paradigms in this field are named, in order to show the diversity of the approaches.

Imitation learning systems are typically used to re-identify and execute human motions [3], [4], [5], [6], aiming at a high similarity of the robot's movements to the demonstrated trajectories. 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 [7], [8], [9], usually require much less or even only a single user demonstration to generate executable task descriptions. Other works [10] use interaction for teaching new task, relaying on a

hierarchical task model. These systems analyze 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 [11].

Sequential analysis of tasks is presented in [12], [13]. 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.

The above emphasized categories are examples of some main direction of research, but surely there is an overlap between them.

III. System architecture and task model

The starting point for acquiring new task knowledge is a Programming by Demonstration (*PbD*) system developed for many years at our lab. This section gives a brief overview of the segmentation process and the manipulation task classes which can be learned by the system in order to describe the gathered information and the assumptions made during this process.

The main idea is to separate the acquisition of task knowledge from the execution of the task on a specific robot in order to achieve a robust task description from a vast external sensor system installed in a so called training center. The sensor environment uses two data gloves with integrated tactile sensors for gathering the finger joint angles and the forces applied on the fingers and palm, magnetic trackers for tracking the hand positions and a headset to record spoken information. For a detailed description of the demonstration environment refer to [14].

For ensuring a representation of robot invariant task knowledge the observed demonstration will be transformed to an abstract strips-like tree description called macro operator (MO) relaying on basic skills (primitives) called elementary operators (EO). An EO denotes an action that abstracts a sensory motor coupling like a grasp or a linear move etc. The EO's build the hardware abstraction layer and have to be implemented on the robot system depending on the available sensors and actors.

A. Classes of Manipulation Tasks

Considering the household and office domain under the hypothesis that the user causes all of the changes in the environment (closed world assumption) through manipulation actions, a goal-related classification of manipulation tasks can be performed. Hereby the manipulation actions can generally be viewed including dual hand and fine manipulations. Furthermore, the goal of manipulation tasks is closely related to the functional view of actions and, besides Cartesian criteria, consequently incorporates a semantic aspect, which is very important for interaction and communication with humans.

According to the functional role a manipulation action aims for, the following classes of manipulation tasks can be distinguished (Figure 3):

Transport operations are one of the widely executed action classes of robots in the role of servants or assistants. Tasks like *Pick & Place or Fetch & Carry* denoting the transport of objects are part of almost all manipulations. The change of the Cartesian position of the manipulated object serves as a formal distinguishing criterion of this kind of task. Consequently, for modeling transport actions, the trajectory of the manipulated object has to be considered and modeled as well. In terms of teaching transport actions to robots the acquisition and interpretation of the performed trajectory is therefore crucial.

Device handling. Another class of manipulation tasks deals with changing the internal state of objects like *opening a drawer, pushing a button etc.* Actions of this class are typically applied when using devices, but many other objects also have internal states that can be changed (e.g. a bottle can be open or closed, filled or empty etc.). Every task changing an internal state of an object belongs to this class. In terms of modeling device handling tasks, transition actions leading to an internal state change have to be modeled. Additionally, the object models need to integrate an adequate internal state description. Teaching this kind of task requires observation routines able to detect internal state changes by continuously tracking relevant parameters.

Tool handling. The most distinctive feature for actions belonging to the class of tool handling is the interaction modality between two objects, typically a tool and some workpiece. Hereby the grasped object interacts with the environment or another grasped object in the case of dual arm manipulations. Like the class of device handling tasks, this action type does not only include manipulations using a tool, but rather all actions containing an interaction between objects. Examples for such tasks are *pouring a glass of water, screwing* etc.

The interaction modality is related to the functional role of objects used or the correlation between the roles of all objects included in the manipulation, respectively. The model of tool handling actions thus should contain a model of the interaction. According to different modalities of interaction, considering contact, movements etc., a diversity of handling methods has to be modeled. In terms of teaching robots the observation and interpretation of the actions can be done using parameters corresponding to the functional role and the movements of the involved objects.

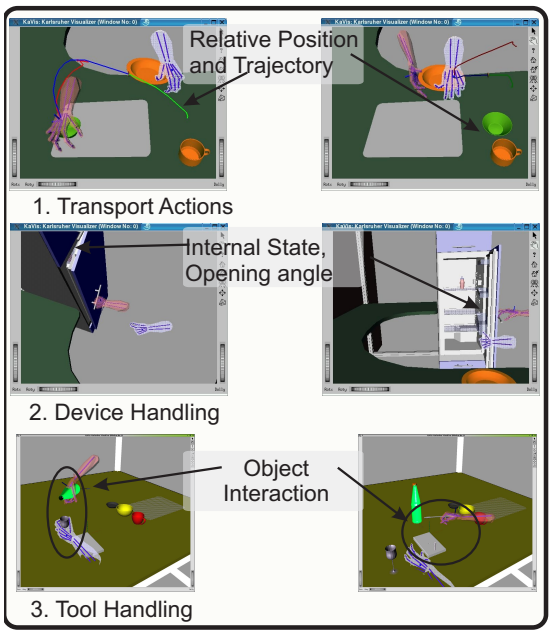


Fig. 3. Examples for the manipulation classes distinguished in a PbD system.

These three distinguished classes are obviously not disjunct, since for example a transport action is almost always part of the other two classes. However identifying the actions as part of these classes eases the teaching and interaction process with robot assistants, since the inherent semantics behind these classes correlates with human descriptions of manipulation tasks.

B. Segmentation of the Demonstration

The phases of the PbD process, described in [15], are a segmentation of the sensor data followed by an analysis and interpretation step for identifying a sequence of EO's which is abstracted and generalized to a macro operator (MO) in a further step.
In order to process the above manipulation classes the segmentation of the sensor data is done in three Phases:

- **Grasp segmentation:**
  Here a stable segmentation of grasp and release actions is performed.
- **Trajectory segmentation:**
  This phase extracts for each grasp and release segment an approach and depart trajectory and fragments the whole hand trajectories in elementary move operations like linear, or circular segments.
- **Statistic segmentation:**
  For detecting object relations during grasp phases, statistic parameters are used in order to generate hypothesis on possible interactions using object-role-dependent probabilities ([16]).

The segmentation of the user demonstration uses a lot of background knowledge about the manipulation process and the environment, respectively the objects used in the demonstrated manipulation. In table I the parameters used for the segmentation are listed. The table shows the dependence of the parameters from the manipulation class. For simple transport actions, in addition to the grasp type, only the trajectory of the objects is needed. Looking at device handling tasks, where the manipulation changes the internal state of the manipulated object (i.e for a "'open a door"' action the state "'door closed"' changes to "'door open"') more information about the object has to be included in the process. The most complex manipulation in terms of detection is a tool handling since in this case the interaction between the manipulated objects determines the goal of the action. In this case more information about the object types and their functional roles is needed in order to detect and describe this kind of manipulations.

TABLE I
PARAMETERS FOR BASIC SKILL SEGMENTATION

Basic Skill	Parameter
Transport:	
Grasp:	Hand:
static	TCP velocity, joint angles
dynamic	+ forces
Move types	TCP trajectory analysis
Device handling:	+ Object model (Type)
Open doors, drawers	Move-axis, handle, state
Push/rotate buttons	Move-axis, handle, state
Tool handling:	+ Functional Role
Screwing	"'Screw-able"'
Pouring	"'pour in /out"'

The result of the segmentation step is a list of key points which are indicating possible starting points of EO's. For generating an action (EO) sequence for each fragment a search for an instantiation of EO's is made. The search is triggered by the information about the

type of the key points and the probabilities of the key points. The proof for the instantiation of a EO requires an evaluation of several EO conditions stored in the EO's. E.g. for instantiating a static grasp a neuronal net is used in order to classify the grasp and a positive classification will result in an instantiation of a certain static grasp (i.e. circular grasp).

C. Hierarchical Task Representation

Representing manipulation tasks as pure action sequences is not flexible and also not scalable. Therefore a hierarchical representation is introduced in order to generalize an action sequence and to prune EO's to more complex subtasks. Looking only on manipulation tasks the assumption is made that each manipulation consists of a grasp and a release action. To cope with the above specified manipulation classes, pushing or touching an object is interpreted as a grasp. The representation of grasping an object constitutes a "pick" action and consist of three sub-actions: an approach, a grasp type and a dis-approach (Fig. 4). Each of these sub-actions consists of a variable sequence of EO's of certain types (i.e. for the approach and dis-approach the EO's are move types and the grasp will be of type grasp). A "place" action is treated analogously.

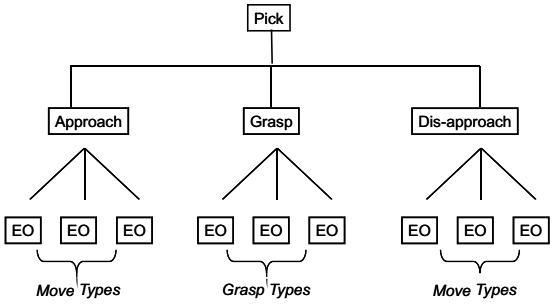


Fig. 4. Hierachical Modell of a Pick Action

Between a pick and a place operation, depending on the manipulation class, several basic manipulation operations can be placed (Fig. 5). E.g. a demonstration of the task "pouring a glass of water" consists of the basic operations: "pick a bottle", "transport the bottle", "pure in", "'transport the bottle"' and "'place the bottle"'. A sequence of basic manipulation operations starting with a pick and ending with a place is abstracted to a *manipulation segment*.

The level of *manipulation segments* denotes a new abstraction level on closed subtasks of manipulation. In this context closed means that a manipulation

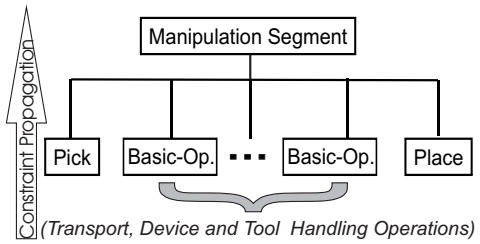


Fig. 5. Representation of a Manipulation Segment

segment ensures that both hands are free and that the environmental state is stable. Furthermore the synchronization of EO's for left and right hands are included in the manipulation segments. Pre and post conditions describing the state of the environment at the beginning and at the end of a manipulation segment are sufficient for their instantiation. The conditions are propagated from the EO level to the manipulation segmentation level and are computed from the environmental changes during the manipulation. In parallel to the propagation of the conditions a generalization in terms of positions and object types and features is done.

A complex demonstration of a manipulation task is represented by a macro operator which is a sequence of manipulation segments. The pre and post conditions of the manipulation segments are propagated to a context and an effect of the macro operator. At execution time a positive evaluation of the context of a macro enables its execution and an error free execution leads to the desired effect in the environment.

IV. Learning of task precedence graphs

This section is concerned with the representation of the sequential features of a task and how it can be learned from multiple user demonstrations of the same 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

$$\mathbf{R}^D = \{(o_i, o_j) | i < j\}, \tag{2}$$

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 $G \succ S$.

The remainder of this section deals with the topic of how a valid hypothesis that complies with all task demonstrations can be learned in an incremental way by using the *more-general*-relation.

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. [17] 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

$$H \succ P_m \wedge H \rightsquigarrow D_{m+1} \wedge (\nexists H' : H' \rightsquigarrow D_{m+1} \wedge H \succ H' \succ P_m) \tag{3}$$

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. 2:

$$\mathbf{R}_{m+1} = \mathbf{R}_m \cap \mathbf{R}^{D_{m+1}} \tag{4}$$

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 2.
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 4.

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.

V. Recognition of Common Subtasks Applying 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 recognize 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.

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

$$s_F(\mathbf{A}, \mathbf{B}) = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A} \cup \mathbf{B}|}. \tag{5}$$

This is the portion of common features in both feature sets.

It turned out that this straight-forward approach exposes one major drawback: As the number of features is relatively high[1] and many of the features are irrelevant to the certain task to be learned, the features carrying the relevant information will be predominated by the irrelevant ones. The solution is to weight each feature f depending on it's relevance to the task to be learned with a weight $w(f)$. This turns equation 5 into

$$s_F(\mathbf{A}, \mathbf{B}) = \frac{\sum_{f \in \mathbf{A} \cap \mathbf{B}} r(f)}{\sum_{f \in \mathbf{A} \cup \mathbf{B}} r(f)}. \tag{6}$$

The weight function $r(f)$ could be influenced by several different aspects like background knowledge, information content of features and vocal elucidations the user gives during the demonstration of a task. This issue is discussed in greater detail in the following section. For now, it is assumed, that a weight function $r(f)$ exists that somehow reflects the importance of a feature.

With the weighted proportion of features s_F as in eq. 6 it is possible to compute the similarities s_{pre} and s_{post} of the pre- and postcondition sets of two subtasks M_1 and M_2 and the average of the subtask similarities $\overline{s_{sub}}$of the tasks. They are set to

$$\begin{aligned} s_{pre} &= s_F(Precond(M_1), Precond(M_2)) \\ s_{post} &= s_F(Postcond(M_1), Postcond(M_2)) \\ \overline{s_{sub}} &= \frac{1}{|M_1 \cup M_2|} \sum_{m_1 \in M_1, m_2 \in M_2} s_M(m_1, m_2). \end{aligned}$$

The (recursive) overall similarity s_M of the two manipulation segments M_1, M_2 is defined as the weighted

[1]About 250-300 feautures for tasks dealing with 3-5 objects.

average of the pre- and postcondition similarity and the average of all subtask similarities:

$$s_M(M_1, M_2) = \alpha_{pre} s_{pre} + \alpha_{post} s_{post} + \alpha_{sub} \overline{s_{sub}}.$$

In the experiments conducted in this paper these 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 permutation that maximizes the 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 IV) 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 IV.

VI. Incremental estimation of feature relevance using vocal comments and past experiences

This section is concerned with the estimation of the relevance of every feature to the task to be learned. Several information channels can be used to guide this weighting process. This section takes into account the physical demonstration and the vocal communication channel. The first provides the representation of actions as described in section III. Here, task specific knowledge (the feature occurrence probabilities over all demonstrations of the same task) as well as background knowledge (the feature occurrence probabilities over all demonstration the robot has seen in his 'lifetime') can be applied.

One way to assess the relevance of a feature is to test its occurrence over several different demonstrations of the same task. Features with a high relevance to the task to be performed have a great probability of occurrence, while features of lower relevance will occurr less frequently. According to Shannon, the information content $I(f)$ of a feature f with probability $p(f)$ is $I(f) = -\log_2 p(f)$. As features with low information content (= high probability of occurrence) to the specific task class $\mathbf{T}$ should be favored, $-\log_2(1 - p(f|\mathbf{T}))$ seems a reasonable choice for the weight function.

On the other hand one has to take into account that when a completely new task is learned or only few demonstrations of the same task are known so far, only little or no information about the distribution of features in the specific task class is known in advance, so the weight function from the last paragraph will not produce reasonable results. The idea is to introduce global background knowledge on the feature distribution. When several demonstrations of different tasks have been observed by the robot and incorporated into the task knowledge base, it can be assumed that the features that uniquely discriminate the task are those that have low occurrence rates accross the whole task knowledge base. Thus, the information content to all other tasks $\overline{\mathbf{T}}$ of feature f is $-\log_2 p(f|\overline{\mathbf{T}})$.

An incremental learning system should be able to cope with both situations: with no or sparse knowledge about the specific task features at the initial learning steps as well as with more task demonstrations coming available in later stages. Therefore a combination has to be found that favors the global information content measure when no or few task demonstrations are available, and the task-specific relevance measure as more demonstrations of the specific task become available. The (partial) weight function for the asessment of feature relevence based solely on the current task and past experiences that fulfills those requirements is

$$w_{demo}(f) = -\left[\alpha \log_2(1 - p(f|\mathbf{T})) + (1-\alpha) \log_2 p(f|\overline{\mathbf{T}})\right] \quad (7)$$

with the relative weighting α of the task-specific knowledge as

$$\alpha = 1 - e^{-k \cdot |\mathbf{T}|}.$$

In our experiments we chose $k = -\frac{\ln 0.25}{5}$, such that after observing 5 demonstrations of the same task, the proportion of task-specific to prior-knowledge is $0.75 : 0.25$ and converging towards 1 for more demonstrations. This choice is motivated by [18], stating that the important features of a task can be sufficiently learned with $4 - 5$ demonstrations. This weight function should be normalized to one, resulting in the relative importance of each feature to the task

$$r_{demo}(f) = \frac{w_{demo}(f)}{\sum_{f' \in \mathbf{F}} w_{demo}(f')}.$$

When, additionally, vocal comments are available that correspond with certain features, the estimate of equation 7 can be further improved. In our system, comments have the form of 'telling the system what is going on'. E.g. when laying the table, the user can tell the system that he/she is putting the fork to the right of the plate on the table. This enables the system to guess that the effect of putting the fork to the right side of the plate is of higher interest to the task than several other features.

Assuming that the evaluation of speech comments resulted in a set $\mathbf{V}$ of features the user reputes important, one can state the vocal part $w_{voc}(f)$ of the weight function as

$$w_{voc}(f) = \begin{cases} 1, & f \in \mathbf{V}, \\ 0, & f \notin \mathbf{V}. \end{cases}$$

This focuses the system on the features the user highlighted during the demonstrations by giving the vocal comments, while ignoring the others. Again, this is transformed into the relative importance function:

$$r_{voc}(f) = \frac{w_{voc}(f)}{\sum_{f' \in \mathbf{F}} w_{voc}(f')}$$

.

Combining these two functions r_{demo} and r_{voc} is a crucial issue. It is unclear a-priori, whether the system should focus on the vocal comments or the actual demonstrations. The first are potentially incomplete as the user comments only parts of the task while he/she rates others as clear and not in need of a comment. On the other hand, the estimate of w_{demo} may be too conservative and unreliable, especially during the first demonstrations of the task, when only little information on the feature distribution of the task class is available.

We propose an approach to this relying on the information content. As we are interested in the features that have high relative weights rather than the ones that are estimated to have low relevance to the task, $-\log(1 - r(f))$ seems to be a good choice (see above). The information contents I_{voc} and I_{demo} respectively for each input modality are calculated as follows:

$$\begin{aligned} I_{voc} &= -\sum_{f \in \mathbf{F}} \log\left(1 - r_{voc}(f)\right), \\ I_{demo} &= -\sum_{f \in \mathbf{F}} \log\left(1 - r_{demo}(f)\right). \end{aligned}$$

Now the final step is to design the overall weight function $r(f)$ as follows:

$$r(f) = I_{voc} \cdot w_{voc}(f) + I_{demo} \cdot w_{demo}(f). \qquad (8)$$

The result in equation 8 presents an assessment of task features' relevance depending on the their occurrence in all demonstrations of the task, vocal comments and explanations given during the tasks demonstrations and background knowledge in form of demonstrations of other tasks already provided to the system. All these different information channels are fused taking into account their information content (the amount by which the system can benefit from that information), providing sound foundations for the measurement of task and subtask similarity and higher level learning, e.g. of task precedence graphs.

VII. Results

This section reports and evaluates the experiments with the system described in the preceeding sections and analyses its results.

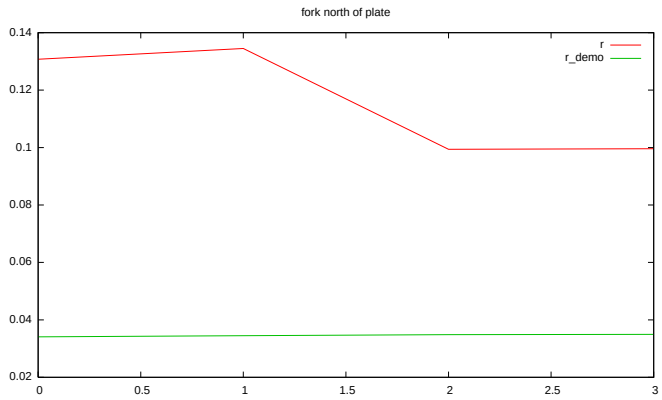
Fig. 6. Learned task of laying the table: final object configuration to be learned

The experiment consisted of teaching an everyday household task from the domain of laying the table. The task for the system to learn was to arrange the objects in a way that can be seen in figure 6: A table should be layed on the silver place mat with a plate, a fork to the right of it and a cup behind the fork (from the users point of view).

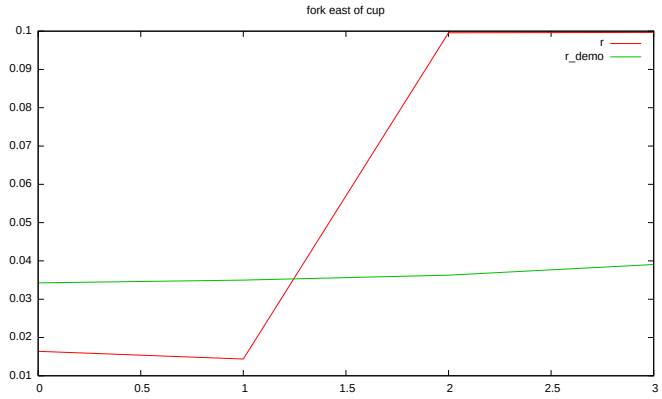
Four different demonstrations of the task were provided to the system. In the first two demonstrations the plate, the fork and the cup were placed in this order. In the third demonstration the first two operations were reversed , while in the last demonstration the cup was placed first, followed by the fork and the plate. The following vocal comments were accompanying the demonstrations: In the first two demonstrations the user told the system "I am putting the plate on the silver place mat, the fork on the silver place mat to the right of the plate and the cup on the mat to the west of the fork". During the third and forth demonstration the user said that he is putting the fork on the mat, the plate to the left of the fork and the cup on the mat in a way, that the fork is to the east of the cup.

The system could extract seven effects of the demonstrations that were highlighted by the vocal comments. These are "plate on place mat", "fork on place mat", "cup on place mat", "fork north of plate" and "cup west of fork" after the first two demonstrations and additionally "plate south of fork" and "fork east of cup" after the third demonstration.

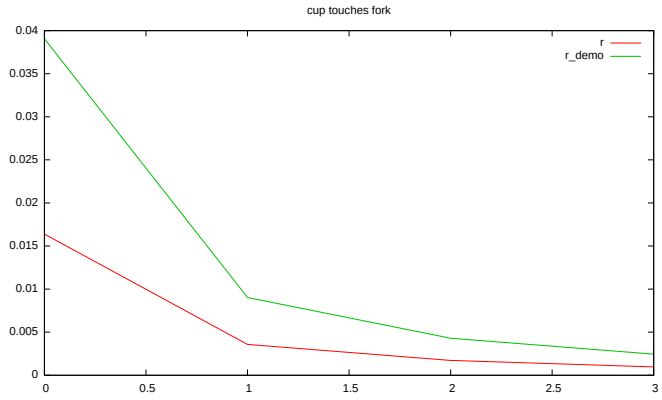
The relative weight function r_{demo}, taking into account only the demonstrations, and the combined relative weight function r over the number of task demonstrations given are plotted for different relations in figure 7. (a) shows the weight function for a very relevant effect: the fork should be put to the north of the plate. One can see that the combined weight function r yields much



(a)



(b)



(c)

Fig. 7. Weight functions $w(f)$ and $w_{demo}(f)$ for certain features over number of task demonstrations

larger results for the relevance estimate than the weight function r_{demo} deduced from the demonstrations alone. The interesting decrease of the relative weight after the third demonstration results from the two additional relations that occurred in the user comments, distributing the overall weight of 1 over seven instead of five effects in the vocal comment weight function r_{voc}. Figure 7 (b) shows the weight functions for the relation "fork east of cup". In the first two demonstrations this effect was

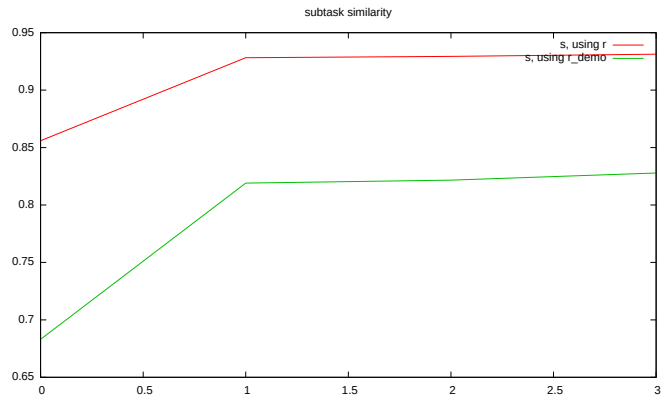


Fig. 8. Subtask similarity measure for the subtask of putting the plate on the place mat over the number of demonstrations.

not mentioned by the user, so the weight is pretty low. As soon as he highlights it in the third demonstration, the relations weight rises. This shows the advantage of incremental learning mechanisms that can adapt their hypotheses stated so far by using additional knowledge in form of demonstrations or comments. Figure 7 (c) shows the decrease of weight for an irrelevant effect, that was only present in the first demonstration by mistake. As it is not stressed by the user comments and does not appear in later demonstrations, its weight decreases with every user demonstration that is given to the system.

The improvement of the weight function applying vocal comments to the subtask similarity measures is shown in figure 8. One can see clearly that the subtask similarity is far greater, leading to more reliable subtask correspondences, which are essential for reidentifying common subtasks during the learning of task precedence graphs.

The first two demonstrations given to the system do not result in a task precedence graph different from the order of subtasks, as no alternative sequences can be observed (see the first row of table II). After the third demonstration has been observed, the system can reason on the sequential independence of the subtasks of putting the plate on the place mat and putting the fork to the right of it (see second row). After the last demonstration of the task is presented, the task precedence graph is completely learned. The system now obtained the knowledge that each of the three subtasks can be performed independently of each other.

VIII. Conclusion

Based on hierarchical, functional and goal oriented task models it is possible to acquire and structure task knowledge from a human demonstration. The presented

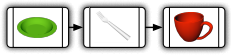
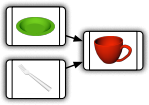
i	P^{D_i}	P_i
1 & 2		
3		
4		

TABLE II

TASK PRECEDENCE GRAPHS LEARNED INCREMENTALLY DURING THE EXPERIMENT.

task models and representations are restricted to manipulation tasks, which contain at least one grasp operation. According to the explanation based theory [19] task knowledge is acquired from a single demonstration. The second part of the paper tackles the problem of structuring the acquired task knowledge trough reasoning. Hereby the concept of task precedence graphs is introduced in order to generalize task knowledge according to the order of subtasks. The crucial point hereby is finding similarity measures for subtasks or actions and since these are based on features the relevance of the features for a certain task have to be estimated. This is done by using the fusion of two channel namely estimating the relevance of features similar to the rule induction using the information theory [20] and deriving relevant information from speech comments. Finally, a brief discussion of the results shows how the task knowledge over lying a table is acquired and processed.

REFERENCES

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

[2] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zoellner, and M. Bordegoni, "Learning robot rehaviour and skills based on human demonstration and advice: The machine learning paradigm," *9th International Symposium of Robotics Research (ISRR'99)*, October 9-12 1999.

[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] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation," *Philosophical transaction of the Royal Society of London, series B*, vol. 358, no. 1431, pp. 537–547, 2003.

[5] A. Alissandrakis, C. Nehaniv, and K. Dautenhahn, "Learning how to do things with imitation," in *AAAI Fall Symposium on Learning How to Do Things*, M. B. . C. Rich, Ed. American Association for Artificial Intelligence, 3-5 November 2000, pp. 1–6. [Online]. Available: citeseer.ist.psu.edu/alissandrakis00learning.html

[6] Y. Demiris and G. Hayes, "Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model." *Imitation in Animals and Artifacts, MIT Press*, vol. 2, pp. 327–361, 2002.

[7] 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.

[8] 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.

[9] R. D. Zöllner, O. Rogalla, R. Dillmann, and J. M. Zöllner, *Extracting Subgoals from Fine Manipulation Tasks.* Springer-Verlag, Berlin Heidelberg 2005, 2004, ch. Advances in Human-Robot Interaction.

[10] C. Breazeal, G. Hoffman, and A. Lockerd, "Teaching and working with robots as a collaboration," pp. 1030–1037, 2004.

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

[12] 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.

[13] 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.

[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] M. Ehrenmann, R. Zöllner, O. Rogalla, and R. Dillmann, "Programming service tasks in household environments by human demonstration," in *Proceedings of the 11th IEEE International Workshop on Robot and Human Interactive Communication (ROMAN)*, Berlin, Germany, 25.-27. September 2002.

[16] 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.

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

[18] 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.

[19] R. M. G. DeJong, *Machine Learning*, 1986, ch. Explanation-based Learning - An alternative view, pp. 145–176.

[20] R. G. P. Smyth, *Knowledge Discovery in Databases*. MIT Press, Cambridge, MA, 1991, ch. Rule induction using information theory, pp. 159–176.