IROS 2003
October 27-31, 2003
Bally Hotel, Las Vegas, USA

Workshop On

Robot Programming Through Demonstration

Half-day Workshop

Organizers:
Aude Billard & Roland Siegwart

Aude.billard@epfl.ch
Roland.siegwart@epfl.ch

Friday 31st of October 2003, 12pm-6pm
Bally's Las Vegas Hotel

Objectives:

Programming by demonstration has become a key research topic in robotics. It impacts both fundamental research and application-oriented studies. Work in that area tackles the development of robust algorithms for motor control, motor learning, gesture recognition and visuo-motor integration. While the field existed for more than twenty years, recent developments, taking inspiration in biological mechanisms of imitation, have brought a new perspective, which this workshop aims at assessing.

Studies of imitation in biological systems stressed a number of key issues that are, also, of concerns to Robotics. These are the need to define *a metric of imitation performance* and the need to determine *a representation common to visual and motor systems*.

The workshop aims at addressing the following key questions:

- What should the robot imitate, i.e. which features of the task should be reproduced?
- Can imitation use known motor learning techniques or does it require the development of new learning and control policies?
- How could the metric drive the choice of learning technique?
- Does imitation speed up skill learning in robots?
- What are the costs of imitation learning?
- How could we define a general metric of imitation performance?
- Are there skills that could not be acquired without demonstration?
- Should gesture recognition and motor learning algorithms be context-specific?
- Can one find a level of representation of movement common to both gesture recognition and motor control?
- How can models of human kinematics, used in gesture recognition, drive the reproduction of the task?

Papers presented at this workshop fall into two categories:

1. research papers (typically 6 to 8 pages)
2. challenge/position statements (typically only 1 or 2 pages)

The papers cover the following topics:

- Programming by Demonstration, Imitation learning
- Task and Skill Learning
- Motor control, Motor learning
- Visuo-motor Integration
- Gesture recognition

Program

TIME	TITLE
12:00	**Welcome and overview** Aude Billard & Roland Siegwart, EPFL (Switzerland)
PART I: Invited Speakers	
12:10	Rüdiger Dillmann ***Teaching and Learning of Robot Tasks via Observation of Human Performance***
12:30	H. Ritter, **Robotics: A data mining perspective**
12:50	Jianwei Zhang ***Self-Valuing Learning and Generalization of Visually Guided Grasping***
13:10	Ales Ude ***Using motion capture techniques to program high degree of freedom humanoid robot motion***
13:30	Jun Nakanishi & Stefan Schaal ***Learning from Demonstration and Adaptation of Biped Locomotion***
13:50	Auke Jan Ijspeert , ***Imitation of Human-Demonstrated Movements with Nonlinear Dynamical Systems in Humanoid Robots***
14:10	Aude Billard, ***Determining what to Imitate in a Manipulation Task***
14:30	**Coffee break (30 min)**
PART 2: Accepted Papers	
15:00	**Leveraging on a Virtual Environment for Robot Programming by Demonstration** Jacopo Aleotti, Stefano Caselli, Monica Reggiani
15:20	**A Posture Sequence Learning System for an Anthropomorphic Robotic Hand** Ignazio Infantino1 Antonio Chella1,2 Haris Džindo1 Irene Macaluso1
15:40	**Motor Representations for Hand Gesture Recognition and Imitation** Manuel Cabido Lopes Jos´e Santos-Victor
16:00	**Improving Robot Programming Flexibility through Physical Human - Robot Interaction** M. Frigola1, J. Poyatos1, A. Casals1 and J. Amat2
16:20	**Learning Issues in a Multi-Modal Robot-Instruction Scenario** J. J. Steil, F. Rothling, R. Haschke, and H. Ritter
16:40	**Towards Integrating Learning by Demonstration and Learning by Instruction in a Multimodal** Robot. Stefan Wermter, Mark Elshaw, Cornelius Weber, Christo Panchev, Harry Erwin
17:00	**Learning From Observation and Practice Using Primitives** Darrin C. Bentivegna, Christopher G. Atkeson, Gordon Cheng
17:20	***Teaching Bayesian Behaviours to Video Game Characters*** Ronan Le Hy Anthony Arrigoni Pierre Bessi`ere Olivier Lebeltel
17:40	**Towards Robot Intermodal Matching Using Spiking Neurons** Emachi Eneje & Yiannis Demiris
18:00	**Wrap-up and goodbye:** Aude Billard & Roland Siegwart, EPFL (Switzerland)

Table of Content:

Teaching and Learning of Robot Tasks via Observation of Human Performance

Rüdiger Dillmann

University of Karlsruhe
Haid-und-Neu-Str. 7, 76131 Karlsruhe, Germany
Email: dillmann@ira.uka.de

Extended abstract:

Programming, cooperation and interaction with upcoming humanoid robots is assumed to be multimodal which means to let the user program the robot simply by speech, gesture or demonstrating the task to be done. The robot observes, interprets and then tries to imitate and to learn the performed user action. A complete system for programming by demonstration is presented which interprets and stores the observed demonstrated actions, segments them into meaningful sequences in a given context including task knowledge. Due to sensor errors and the complexity of the intended interaction, it is usefull, that the system generates queries concerning intention, manipulation and objects. Motion and grasp capture techniques in combination with learning capabilities are combined towards imitation learning and active observational learning strategies.

To achieve the goal of multimodal interaction, an effective representation and interpretation of the world is required. The context in which we focus on this research is spontaneous continuous and adaptive humanoid interaction and cooperation with the robot. As we believe, programming by demonstration and advice (PbD) and imitation learning is a technique that overcomes the drawbacks of classical robot programming approaches and could be a natural way to program humanoid robots. Providing multimodal bi-directional interfaces allows both, modeling of motion and interaction but also setting up a common representation of the world programmed in an intuitive mode. Thus, human motion can be traced, segmented and interpreted. The observed action sequences can be mapped into a generalized representation and finally adapted and executed by a target robot. Full body motion catch techniques combined with statistical learning techniques [Schaal 99] are considered to implement imitation learning of natural human movement by the robot.. If more natural motion behaviour can be achieved a higher acceptance rate for robots in various applications may be the result [Billard et al., 01].

Several PbD systems and approaches for different applications have been proposed in the past. An overview and classification can be found in [Dillmann et al., 99]. Learning of skills, action sequences or operation plans requires modelling of cognitive skills. In fact, the purpose of observing the user is to generate an abstract description of the demonstration, reasoning about the user's intention and modelling the problem solution preferably in an optimal way. Likewise, a given problem has to be generalized by identifying for example significant parameters and attributes being characteristic for a particular demonstration and by deriving data being relevant for the problem concept. Both requires background knowledge about the environment and about the user's behaviour and performance.

The analysis of demonstrations may be based on the observation of dynamic transitions in the workspace manipulated by the user. These dynamic transitions can be described using relational expressions or contact relations [Kuniyoshi 94, Ikeuchi 93, Onda 97]. For generalising a single

demonstration typically explanation based methods are used [Mitchel 86] because they allow an adequate generalisation taken from only one example (One-Shot-Learning).
Because physical demonstration in the real world may be very time-consuming some work addresses virtual demonstrations [Archibald 93]. Augmented reality (AR) techniques and simulation allow some additional degrees of freedom to the user. Hence, in some approaches object poses or trajectories [Takahashi 96, Tung 95, Kang 97] and/or object contexts [Onda 97, Heise 92, Segre 89] are acquired and generalised with the help of such techniques.
In addition direct cooperation between the user and the robot has been investigated. Here, user and robot operate in a common working space. The user may direct the robot with the help of a common vocabulary like speech or gestures [Zhang 99, Voyles 99, Steinhage 98].
For the humanoid robot project experiments following the PbD paradigm have been undertaken in a stationary environment.
The perceptive capabilities of the robot are essential for the humanoid robot to be able to interact within a dayly dynamic environment including humans. An active stereo vision system with steerable cameras and an microphone array integrated into a head combined with a flexible neck is used to identify objects, recognizing people and identifying their focus of attention. The purpose is tracking of people and analyzing their intention and activities. The perceptive and control system steers the robot behaviour. Typical capabilities of the audio- vision system include face recognition, people tracking, lip reading, gesture recognition, identification focus of attention, speech recognition as well as estimation of speaker position [Yang et al 98, Stiefelhagen et al 99, Kroschel et al 95].

Multimodal Man Robot Interaction

To improve intuitive interaction the robot must provide a multimodal user interface which enables the user to communicate via speech, via gesture and via direct physical interaction with the robot system.
Gesture recognition enables the user to control and command the robot in a straightforward way. Additionally, it extends the intuitive speech interface by pointing gestures (e.g. to select the objects to be grasped). This additional information gathered by gesture recognition will be fused with the dialogue context to a multimodal environment context in order to understand focus of interest, what the user desires and what he is talking about. As an extension of this context the interactive environment modelling (the user explains a new environment structure and situations to the robot) is a very important scientific challenge in that field. This yields to dynamic vocabularies and dynamic grammars.
A part of the multimodal environment context is the actual user state configuration. The robot needs all information about the user (head, arms, hands etc.) that is available. Thus, the gesture recognition is needed to complete upper body recognition and later on to the entire human to identify his intention and the context of actual situations.

Adaptation and Learning

Various learning strategies may be applied on all levels of abstraction in the cognitive and perceptive architecture of the humanoid robot:

- Learning of natural type basic movements (human-like movement)
- Learning of coordinated movements (human-like movement)
- Learning of tasks (e.g. assembly, fetch and carry, loading, unloading, opening/closing a door, cleaning etc.)
- Learning of cooperative tasks or task-pattern

Learning and mapping of natural basic movements requires adaptive controls for the whole robot body motion system. Learning of coordinated movements includes learning of all control sets for entire-arm-movements, entire-body-movements etc.

Learning of manipulative tasks may be based on the programming by demonstration paradigm. The user demonstrates the solution of a problem to the robot. The robot observes the solution and tries to imitate. Basic skills or task solutions may be generated with the help of a training center which can be a simulator to be accessed via a network. Simulation allows to correct the generated robot program. Such a program may consist of an operator tree based on a huge set of basic skill operators.
Learning of cooperative tasks is similar to learning autonomous tasks. The robot is incorporated into the process by being commanded via its multimodal interface. The entire user demonstration including its own actions is recorded and post processed. For activating the patterns or execution of them the specification of competences or roles (who does a specific action in a concrete case ?) between the robot and the user is required.

Programming by demonstration for manipulation tasks

In the following our approach to PbD is shortly discussed. One key issue for the robot is the interpretation of what has been done by the human demonstrator. If the humanoid robot can derive for a given user demonstration a clear and correct hypothesis, it enables the robot system to reuse this observed action in similar or slightly different situations. The PbD process starts with a user task demonstration which is observed by a sensor system. Within a sequence of 7 processing phases the data derived from the human demonstration is converted into a control program:

1. A sensor systems observes the user´s intention, motion, interaction and behaviour. Dynamic changes of object positions and constraints in the environment can be detected. If context information is given as input, the system`s sensor processing and tracing performance can be improved considerably..
2. During the next phase relevant operations and environment state changes can be derived based on the acquired sensor data. This process is called segmentation. Segmentation can be performed online during the observation process or offline based on recorded data. The segmentation processing performance can be improved significantly by user input concerning context and purpose of the observed interaction to encrease the systems hypothesis generation capabilities.
3. Within the interpretation phase the segmented user task demonstration trace is mapped onto a sequence of symbolic descriptions. These symbols contain information about the action (e.g. type of grasp, trajectory type) as well as important data from the sensor traces like forces, used for grasping, etc.
4. Abstraction from the given demonstration for representing the task solution as general as possible is the next processing step. Generalization of the obtained operators may include further advice by the user. Spontaneous and not goal oriented motion and actions may be identified and filtered in this phase. The task knowledge is stored in a form that allows reuse even if the execution constraints and conditions differ slightly from the demonstration.
5. Transfer of the internal symbolic knowledge representation to the target system has to consider physical properties of the robot.. Input to the mapping process is the generated task solution knowledge from the previous processing phase. Background knowledge about the kinematic constraints of the target robot system is matched with the abstract task description.
6. In the following process the mapped action sequence may be tested with the help of a simulator. In this phase the user can access the system and can confirm and evaluate performance and correctness to avoid ambiguous situations during real task execution.
7. During the execution process success and failure information can be used to evaluate and modify the actual task program but also the mapping process.

The PbD technique has been applied to fetch and carry operations of various household objects and to other service tasks. However, the experiments with real household objects yield to learning of different grasp-types for distinct robots. A 2 - arm - 2-hand demonstration system is set up actually to demonstrate simple 2 coordinated hand tasks in the above described environment.

System structure

The system structure is shown in figure 1. It integrates four basic modules. The sensor module is responsible for analysis and presegmentation of data from the observer channels connected with the sensor systems. It's output is a vector describing environment states and user actions. This information is stored in a world model database.
The second module operates on the gained observation vectors and associates sequences of observation vectors to a set of predefined symbols. These parametrized symbols represent elementary action sets. During the interpretation phase the symbols are chunked into hierarchical macro operators after replacing specific task-dependant parameters by variables. The result is stored in a database as generalized execution knowledge.
This knowledge is taken from the execution module which uses specific kinematic robot data for processing. It calculates optimized trajectories for the target system taking into account the actual world model.
Before mapping the generated program to the target system it's correctness is tested by simulation. In case of unforeseen errors or conflicts, the movement of the robot has to be corrected and optimized.
All four components communicate with the user via a graphical user-interface. Additional information can be retrieved or hypotheses can be accepted or rejected. While demonstrating, the user may use gestures for interaction as well.

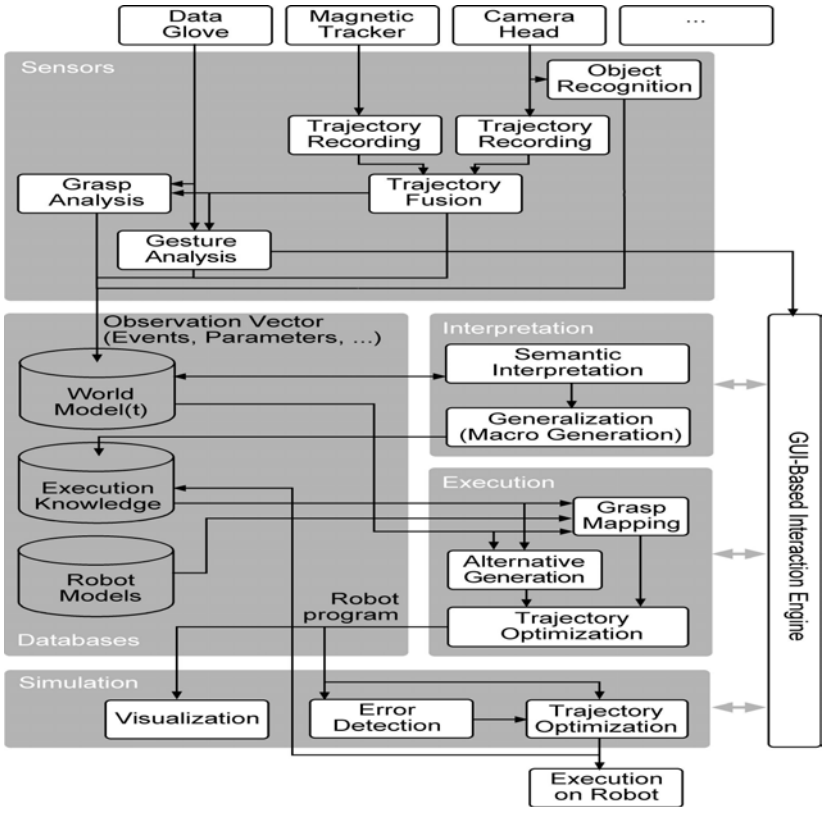


Figure 1: System structure

Future work

For demonstration of various other tasks to the humanoid robot, the multimodal user interface will be enhanced. Several interaction channels will allow intuitive interaction between man and machine:

- Speech recognition will be extended with face detection and eye tracking to determine focus of attention.
- Gesture recognition considering static and dynamic gestures. Gestures will be used for commanding and commenting a demonstration and for interaction in user dialogues.
- Markov representations embedded in a dynamic grammar are planned to describe observed natural motion of humans acquired by motion capture techniques. Cooperation patterns will me modeled to enable the robot to continuously interact with humans in a given context.

Acknowledgement

This work is supported by the Deutsche Forschungsgemeinschaft (DFG) as Collaborative Research Center 588 Humanoid Robots, the DFG project "Programming by Demonstration" and also by the BMBF lead project MORPHA .

Related references

[Archibald et al 93] C. Archibald and E. Petriu. Computational paradigm for creating and executing sensorbased robot-based programming. ISIR, 401-406, Tokyo, Japan, 1993.

[Billard et al., 01], A. Billard and M. J. Mataric, Learning Human Arm Movements by Imitation: Evaluation of a Biologically Inspired Connectionist Architecture Robotics and Autonomous Systems (2001), 145-160

[Cutkosky 89] M. R. Cutkosky. On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks. IEEE Transactions on Robotics and Automation, 5(3), 269-279, 1989

[Dillmann et al. 99] 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. 9th International Symposium of Robotics Research (ISRR '99), 229-238, Snowbird, Utah, USA, October 9-12, 1999

[Ehrenmann et al. 00] M. Ehrenmann, D. Ambela, P. Steinhaus and R. Dillmann. A Comparison of Four Fast Vision Based Object Recognition Methods for Programing by Demonstration Applications, Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA), San Francisco, California, USA, 1862-1867, April 2000

[Ehrenmann et al. 01a] M. Ehrenmann, T. Lütticke and R. Dillmann. Dynamic Gestures as an Input Device for mobile platforms. IEEE International Conference on Robotics and Automation, Seoul, Korea, May 21-26, 2001

[Ehrenmann et al. 99] M. Ehrenmann, P. Steinhaus and R. Dillmann, A Multisensor System for Observation of User Actions in Programing by Demonstration, Proceedings of the IEEE International Conference on Multi Sensor Fusion and Integration (MFI), Taipeh, Taiwan, 153-158, August 1999

[Kang et al 97] S. B. Kang, K. Ikeuchi. Toward automatic robot instruction from perception – mapping human grasps to manipulator grasps. IEEE Transactions on Robotics and Automation, 13(1), Februar 1997

[Mitchel 86] T.M.Mitchell. Explanation-based Generalization - a unifying View. Machine Learning, 1;47-80, 1986.

[Onda et al 97] H. Onda, H. Hirukawa, F. Tomita, T. Suehiro, and K. Takase. Assembly motion teaching system using position/force simulator – generating control program. 10th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS`97), Grenoble, Frankreich, September, 7-11, 1997.

[Schaal 99]. S. Schaal, Is Imitation Learning the Route to Humanoid Robots? Trends in Cognitive Sciences 3 (6) (1999) 233 - 242

[Segre 89] A.M. Segre. Machine Learning of Assembly plans. Kluwer Academinc Publishers, 1989

[Steinhage et al 98] A. Steinhage, T. Bergener. Dynamical Systems for the Behavioral Organization of an Anthropomorphic Mobile Robot. In R. Pfeifer, B. Blumbert, J.A. Meyer and S.W. Wilson

(Editors), From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB 98). MIT Press, August 1998.

[Stiefelhagen et al 99].R.Stiefelhagen, j.Yang, A.Waibel, Modeling People´s Focus of Attention, IEEE Int. Workshop on Modelling People. Sept. 1999, Corfu

[Takahashi et al 96] T. Takahashi. Time normalization and analysis method in robot progrmming from human demonstration data. Proceedings of the IEEE International Conference on Robotics and Automation, 695-700, Atlanta, Georgia, USA, 1993

[Tung et al 95] C.P.Tung, A.C. Kak. Integrating sensing, task planning and execution. IEEE International Conference on Robotics and Automation, 3;2030-2037, 1994

[Voyles 99] R. Voyles and P. Khosla, Gesture-Based Programming: A Preliminary Demonstration. Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, Michigan, 708-713, May 1999

[Yang et al 98].J.Yang, R.Stiefelhagen, U.Meier, A.Waibel, Visual Tracking for Multimodal Human Computer Interaction, Proc. Of CHI´`98, p. 687-694, Hongkong

[Zhang et al 99] Jianwei Zhang, Yorck von Collani, Alois Knoll. Interactive Assembly by a Two-Arm-Robot Agent. Robotics and Autonomous Systems, Elsevier 1999.

[Zöllner et al. 01] R. Zöllner, O. Rogalla and R. Dillmann. Integration of Tactile Sensors in a Programming by Demonstration system. IEEE International Conference on Robotics and Automation, Seoul, Korea, May 21-26, 2001

Robotics: A Data Mining Perspective

Helge Ritter
Neuroinformatics Group
Faculty of Technology
Bielefeld University

As sensor technology matures, modern robots can become equipped with increasingly rich sensor systems. While this fills an important prerequisite to make robots behave more smartly and give them abilities closer to what we observe (and admire) in animals and humans, this very desired progress confronts us with ever increasing dimensionality and complexity of the acquired real-time sensor data stream. At the same time, in the majority of cases the relationship between the sensed data and their causes in the environment is sufficiently complex to prevent the use of readily available, explicit models for their evaluation. Therefore, we will have to continue and intensify efforts to develop adaptive and learning methods that can work on the basis of approximate or even imprecise a-priory assumptions about underlying regularites and relationships between sensor data and the robot environment.

There is an interesting analogy to this situation in an apparently rather different domain. The increasingly "computerized" processing of business and purchasing transactions at all levels, in connection with modern means of acquiring information about customer behavior plus numerous other market variables has endowed companies with an enormously broadened set of "sensors" about the environment in which they struggle for growth or survival. In the market domain, explicit models to transform these "sensory data" into useful information about the state of the environment are even more difficult to derive from firm principles than in robotics, where at least in principle physics should be the ultimate ground[1].

The field of *data or information mining* [2, 5] has evolved as a response to and largely driven by these needs during the last decade. It comprises a broad set of methodologies to "sift" through large amounts of data, typically of a high dimension and in the absence of precise a-priori models, in order to discover useful regularities or structures. With this goal, it has embraced the combined development of computational learning techniques, visualization and database methods in order to build systems that may be viewed as (semi-)intelligent sensory interfaces into complex market environments that can extend our natural cognitive apparatus into domains for which nature so far had no possibility (and need) for evolutionary adaptation.

Consequently, we can discern many areas where robotics and data mining can benefit from each other. A first and obvious level of exchange concerns basic methods: algorithms for basic data preprocessing, such as dimension reduction, clustering, classification, model extraction and time series prediction are of crucial importance to both fields and therefore actively developed by both communities [16]. While there is a correspondingly large overlap of expertise of researchers in both fields, there tend to be characteristic differences that may make it worthwhile to listen to the respective other field. One strong emphasis in the data mining area concerns scalability to extremely large (either in volume or dimensionality or both) data sets [6]. This has spurred much research towards finding approximate variants to many important algorithms in order to circumvent their frequently quadratic or worse scaling in some data size parameter by a more favorable scaling law. Examples are fast approximate techniques for dimension reduction (random projection [11], fast map [4]), or efficient schemes for computing correlation functions on very large data sets [7]. A related and very active field concerns data structures for rapid search. Here, datamining research has triggered a development of increasingly sophisticated data structures and search paradigms that can beat previous methods for nearest neighbor and similarity searches on large and high-dimensional data sets by orders of magnitude [1, 9].

Such developments will heavily gain in importance as soon as we move to the control of advanced robots with many degrees of freedom [14] or extend the memory span of our current (almost memory-

[1] even in robotics much of this ceases to be useful when robots have to cooperate with humans.

less) robots to systems that collect data over significant periods of their entire life-span. Such "life-long" learning may turn out to be the only viable solution to acquire the huge mass of world knowledge that is required for even moderately "intelligent" behavior. Current disk memory technology is no longer a constraining factor to such data-intensive approaches, and to cope with the resulting immense data piles is likely to benefit strongly from techniques developed in datamining and intelligent databases.

Specifically, recent progress in content-based image database indexing and retrieval [13, 15] may have much to offer for robotics. Future intelligent robots should be endowed with some kind of episodic long-term memory to accumulate visual and other sensory experiences. Using raw sensor images for this purpose has many attractive features, provided we can solve the task of efficiently indexing into and navigating within large collections of such data [10]. Unlike symbolic scene descriptions, raw sensor images are easy to acquire and collect. Moreover, they do not enforce a prior commitment onto a narrow range of possible future queries but remain "open" to inspection under aspects that may be unforeseable at the time of their acquision. This flexibility may be one of the reasons why also the memory system in our brain offers an apparently visually organized interface to our episodic memory store. In fact, recent progress in semantically organizing large image collections with machine learning techniques for unsupervised category formation [3] and automatic labelling with classifiers previously trained on a variety of visual object domains [17] (so that human keyword assignment becomes dispensable) can be seen as the first promising steps towards the self-organized structuring of a larger body of sensory experience for an artificial cognitive system. Obviously, any progress along these lines is of immediate significance for robotics also, a field which may motivate us to extend such approaches to additional modalities, e.g., collecting and organizing a database of haptic experiences for dextrous object manipulation.

Another area of prime importance in data mining has always been data rendering, in particular visualization, and the development of innovative interfaces to inspect, browse and explore data interactively. Results in these areas are of importance to robotics in at least two ways. First of all, viewing robots themselves as "datamining engines" foraging for information and regularities in the sensory images of their environment, algorithms for efficient data exploration and data summarization can be of direct interest for developing improved exploration abilities for the robot. Secondly, viewing the robot itself as a complex dynamical system, we are faced with the task to provide to the researcher and — at a more distant level — the user with insightful state visualization and browsing capabilities in order to interact with the robot, either at the engineering or at the user level. This can benefit from applying datamining techniques *to the* robot system. For instance, recently we have developed a multimodal state-monitor for a research prototype of a rather large interactive robot system, integrating active stationary and manipulator-based vision systems, a speech-analysis system, a short-term-memory and several further modules for robot arm and manipulator control for situated human-robot cooperation [12]. As an approach, we combined adaptive visualization techniques with the rather recent approach of sonification in order to convey an informative, yet intuitive multimodal display interface [8] for the state and the interactions of about 30 processes operating simultaneously in that system. We found that this interface originally inspired from datamining techniques was very helpful in the final debugging and tuning phases of the complex system.

We therefore think that a data mining perspective of robotics can contribute some fresh and useful views onto quite a number of difficult and important questions that need to be solved to make robots more cognitive. Navigation in data spaces is a central issue in data mining, and navigation in real, but also abstract goal spaces is a major task for robots. We think that this is not just a coincidence of terms, but a deeper-reaching analogy in which natural evolution has already gone an impressive way, manifested in the creation of highly developed nervous systems and brains. In contrast, information mining is a very young late comer, developing navigational capabilities for agents or "bots" in so-far largely artificial data worlds which in the web and in large geodata or image collections start to become fused with more and more aspects of the "real" space that makes up our own world. The development of robots occurs somewhere in between, benefitting already from ideas about processing in the brain and likely to benefit no less from insights and inspiration from mining the increasingly important "material" of information.

References

[1] Stefan Berchtold, Bernhard Ertl, Daniel A. Keim, Hans-Peter Kriegel, and Thomas Seidl. Fast nearest neighbor search in high-dimensional spaces. In *Proc. 14th IEEE Conf. Data Engineering, ICDE*. IEEE Computer Society, 23–27 1998.

[2] Ming-Syan Chen, Jiawei Han, and Philip S. Yu. Data mining: an overview from a database perspective. *IEEE Trans. On Knowledge And Data Engineering*, 8:866–883, 1996.

[3] Yixin Chen, James Z. Wang, and Robert Krovetz. An unsupervised learning approach to content-based image retrieval. In *Proc. IEEE International Symposium on Signal Processing and its Applications*, Paris, France, 2003.

[4] Christos Faloutsos and King-Ip Lin. FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In Michael J. Carey and Donovan A. Schneider, editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174, San Jose, California, 22–25 1995.

[5] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. *From data mining to knowledge discovery: an overview*, pages 1–34. MIT Press, 1996.

[6] Venkatesh Ganti, Johannes Gehrke, and Raghu Ramakrishnan. Mining very large databases. *Computer*, 32(8):38–45, 1999.

[7] Alexander G. Gray and Andrew W. Moore. 'n-body' problems in statistical learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 521–527. MIT Press, 2001.

[8] Thomas Hermann, Christian Niehus, and Helge Ritter. Interactive visualization and sonification for monitoring complex processes. In *Proc. of the Int. Conf. on Auditory Display*, pages 247–250, Boston MA USA, 2003.

[9] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theor. Comput.*, pages 604–613, 1998.

[10] T. Kämpfe, T. Käster, M. Pfeiffer, H. Ritter, and G. Sagerer. INDI – Intelligent Database Navigation by Interactive and Intuitive Content-Based Image Retrieval. In *IEEE 2002 International Conference on Image Processing*, volume III, pages 921–924, Rochester, USA, 2002.

[11] S. Kaski. Dimensionality reduction by random mapping. In *Proc. Int. Joint Conf. on Neural Networks*, volume 1, pages 413–418, 1998.

[12] P. McGuire, J. Fritsch, J. J. Steil, F. Röthling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter. Multi-modal human-machine communication for instructing robot grasping tasks. In *Proc. IROS*, pages 1082–1089. IEEE, 2002.

[13] Y. Rui, T. Huang, and S. Chang. Image retrieval: current techniques, promising directions and open issues. *Journal of Visual Communication and Image Representation*, 10(4):39–62, April 1999.

[14] S. Schaal, C.G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparameteric statistics for real-time robot learning. *Applied Intelligence*, 17:49–60, 2002.

[15] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:1349–1380, 2000.

[16] Padhraic Smyth. *Data Mining At The Interface Of Computer Science And Statistics*, pages 35–61. Kluwer, 2001.

[17] James Z. Wang, Jia Li, and Gio Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23:947–963, 2001.

Self-Valuing Learning and Generalization of Visually Guided Grasping

Jianwei Zhang and Bernd Rössler

Technical Aspects of Multimodal Systems (TAMS), Department of Computer Science,
University of Hamburg, D-22527 Hamburg, Germany
{*zhang,roessler*}*@informatik.uni-hamburg.de*

Abstract

We present a self-valuing learning technique which is capable of learning how to grasp unfamiliar objects and generalize the learned abilities. The learning system consists of two components which distinguish between local and global quality criteria for grasp points. The local criteria are not object-specific while the global criteria cover physical properties of each object. In this case we present a generalization method of the learning parameters based on a tree distance model for the medial axis transformations. The system is self-valuing, i.e. it rates its actions by evaluating sensory information and the usage of image processing techniques. An experimental setup consisting of a PUMA-260 manipulator equipped with a hand-camera and a force/torque sensor, was used to test this scheme. The system has shown the ability to grasp a wide range of objects and to apply pre-learned knowledge to new objects.

1 Introduction

In a wide range of robotic systems grasping is a basic skill that is crucial to manipulation tasks and interaction with the environment. In most industrial applications the problem of grasping is solved via *teaching-by-doing* or static programs. However, when thinking of recent research fields, e.g. service robots or humanoids, aspects of sensor-based grasping will play a very important role. New techniques must be developed for the robots to operate in uncharted and unknown territories. They should consider elements of human learning abilities, e.g. the human potential of generalization, when constructing a robotic grasping system.

2 Related Research

A lot of work has been done in the field of robot grasping. [1] gives a brief overview of the field over the last two decades. The commonly used analytical approaches try to compute optimal grips according to special heuristics (e.g. [2] and [3]). In these cases either a fully specified model of the object and its mass distribution is known or the center of area of the object, extracted via image processing, is used to "approximate" the real center of gravity. The first case is very difficult to obtain via external sensors and without any a priori knowledge. A complete 3D representation of the object via image processing needs to be gained and additionally features like the material of the object need to be examined. However, a hidden internal inhomogeneous mass distribution can never be found with such an approach. The latter case of using the center of the object's area is certainly only an approximation. This would work if the center of gravity coincides with the object's center of area, but would not work in case of inhomogeneity, too.
Among several attempts to handle the problem of learning how to grasp, [4] presents a system that learns how to grasp objects with a parallel-jaw gripper. Two main subproblems are learned: to choose grasping points and to predict the quality of a given grasp. The disadvantage of this system is that only *local* criteria are used to store grasping configurations. Without *global* criteria it is for example impossible to learn how to grasp an object whose center of gravity does not coincide with the center of its image area. Without self-valuing learning techniques it is impossible to handle the real physical properties of an object. [5] presents a learning system for visual guided grasping, constructed of two learners. This system is not self-valuing, i.e. the optimal grasp point has to be given initially to the learner. Therefore, the two learners are not generalizable to new objects either. In [6] a system is developed that performs skillful grasping with a dextrous robot hand by emulating some infants' growth processes.

3 Learning Scheme

To construct a robotic learning system, it is useful to investigate elements of human learning abilities. In recent work, of course, the human hand with its five fingers is considered, which is more complex than a simplified robot gripper. Therefore, in this paper an approach is suggested that is based on our idea of human learning abilities when grasping an object with two fingers.

3.1 Local and Global Quality Criteria

Our work is based on our idea that when intending to grasp an unfamiliar object with two fingers, one can

mainly consider two kinds of quality criteria on how to choose optimal grasp points. These two kinds are further referred to as *local quality criteria* and *global quality criteria* or *local/global criteria* for short. Together they form the basis for the underlying learning system design.

Local Quality Criteria: The local quality criteria are mostly independent of a special shape and therefore of global aspects like the distribution of an object's mass. Therefore, they are valid for any kind of object. Local quality criteria are considered first when one decides to grasp an unfamiliar object. Such a criterion is for example the sliding friction between an object and the fingers of the gripper.

Global Quality Criteria: Global quality criteria, by contrast to the local ones, are closely connected to a special object and therefore seldom significant for other objects. They are applied after the local criteria to find the optimal grasp point. These criteria consider aspects like the distribution of mass of an object, e.g. the torque at the gripper when grasping an object.

Technically speaking, the local criteria define a virtual axis on which a possibly good grasp point may be found with the help of the global criteria. For example, the grasp configuration computed in Fig. 1(b) could be determined from the search axis proposed by grasp point 3 in Fig. 1(a). In the learning process these criteria are repeatedly considered one after the other for a finite number of steps until a good grasp point is found. The number of steps varies with the skill of the learner and the structure of the object. For a familiar type of object, the global and local criteria are considered in only one step. In fact, since the same local criteria can be applied to any kind of object, as mentioned above, they are fully learned prior to the global criteria.

3.2 Optimality Conditions

A grasp point is considered to be optimal, if the quality criteria are met in the following fashion:

- the fingers can cover the object at this grasp point, and
- no sliding friction occurs between the fingers and the object.

It is considered to be optimal according to the global quality criteria if:

- no torque occurs between the fingers grasping the object, and
- the grasp is stable, i.e. the object does not slip between or out of the fingers.

Some sample grasp configurations are shown in Fig. 1.

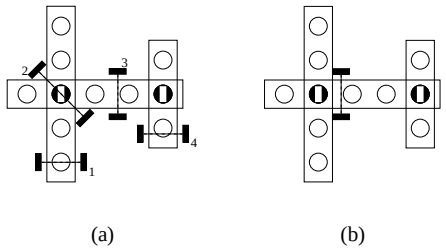


Figure 1: *Optimal grasp points in terms of the local and global quality criteria. a): some example grasp configurations which are optimal according to the local quality criteria. b): the grasp point is optimal according to both criteria.*

3.3 Higher Level Criteria

An additional and *higher level criterion* for human grasps is the role of the grip, i.e. its importance for carrying out further operations, e.g. grasping a cup at its bail in order to drink something or a sledge at its handle to drive a nail into a wall[1]. Other higher level criteria are for example the material or surface of an object. To consider these criteria additional sensors or sophisticated image processing techniques need to be integrated, which is beyond the scope of this work. Our objective is to emulate the abilities of somebody who just wants to learn to get hold of an object as good as possible.

4 Two-Learner System

The quality criteria mentioned above suggest a system consisting of two learners, one for the local and the other for the global quality criteria. The states for the first learner only consist of the local features $s = (f_{l_1}, \ldots, f_{l_m})$. The learner tries to map them to actions consisting of a rotational component $a = \phi$. The second learner tries to map states of global features $s = (f_{g_1}, \ldots, f_{g_n})$ to actions of translational components: $a = (x, y)$. Because the local criteria are mainly covered by the relative orientation of the gripper, the responsible learner is called *orientation learner*. The global criteria are affected by the position of the grasp point in the object and therefore the proper learner is further referred to as *position learner*. These two learners operate right after each other (Algorithm 1). The key element for a good performance of such a learning system is the choice of adequate features which reflect the local and global quality criteria. The choice depends mainly on the kind of gripper and other available sensors used in the system. The local and global features are shown in Fig. 2. The first component of the state vector of the orientation learner is length L of the grasp-line. With this feature, good grasp

[1]For this higher level criteria, aspects of optimality like reducing torque must possibly be shelved.

Algorithm 1 Algorithm for learning an optimal grasp point

choose an initial grasp point configuration
$steps \leftarrow 0$
repeat
  $steps \leftarrow steps + 1$
  repeat
    learn with the orientation learner
  until [the grasp point is optimal according to orientation OR number of episodes exceeds a given value]
  repeat
    learn with the position learner
  until [the grasp point is optimal according to the position in the object OR number of episodes exceeds a given value]
until [the optimal grasp point is found OR $steps > steps_{max}$]

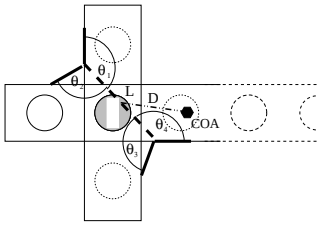


Figure 2: *State coding for the learners. The orientation learner uses length L and angles $\Theta_1, \ldots, \Theta_4$ while the position learner integrates the distance D between the center of the grasp-line and the center of area of the object's image.*

points are distinguished from those which are inadequate because the gripper cannot cover the object. The remaining features are the corresponding angles $\Theta_1, \ldots, \Theta_4$ between the grasp-line and the flanking straight line segments gained by a simple contour tracking process. The features for the position learner are the distance D between the center of the grasp-line and the center of area of the object's image and the torque T around the normal vector $\vec{n}$ of the gripper. Due to the learner separation the local criteria need not be learned for every new object. The orientation learner is a universal learner, which means that the same learner can be used for every object. E.g., this learner can learn to grasp objects at opposite parallel or concave edges.

In principle, a two-learner system design was first introduced in [5]. The new aspect of our work is a different usage of the two learners. As described above this design was chosen in respect to the local and global criteria and their generalization properties.

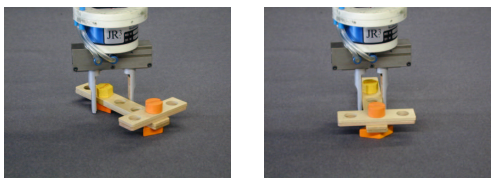
4.1 Self-Valuation

The presented system is self-valuing, a method to gain an estimation for the learning algorithms. Self-valuation is done via a force/torque sensor and the hand-camera at the gripper of the robot. It is important to mention that no optimal grasp point is pre-known and the system finds its own grasp points taking into account the quality criteria.

Orientation Learner. The best estimation of a good grasp, determined by the orientation learner, is obtained by the second optimality condition, i.e. no sliding friction at the fingers of the gripper. When an object slips between or out of the fingers at the moment of closing the gripper, the selected grasp configuration was not optimal according to the local criteria. Some existing systems (e.g. [2]) try to determine the friction occurring within the gripper analytically, i.e. by computing the friction cone via geometrical features. Here, several grasp configurations are tried out with the real robot that values the success or failure of the performed grasps - like humans who do not analytically compute their optimal grips, but learn by success and failure. Because the gripper used in this work does not slip like human fingers over the object's surface, sliding friction appears either as a rotation or as a displacement of the object itself. The valuation of the orientation learner is basically gained by image processing. A penalty for self valuation is computed as follows:

$$P = \begin{cases} -(\Theta_{\text{diff}} + \mathcal{D}_{\text{diff}}) & \text{if grip was successful} \\ -\mathcal{P}_{\text{const}} & \text{otherwise} \end{cases}$$

where Θ_{diff} is the angle between the initial and the least inertia axis after the performed grasp, D_{diff} the displacement of the center of area and P_{const} a high constant penalty which is greater than the highest estimated changes in orientation and position together. Fig. 3 shows a grasp configuration which results in a rotation of the object itself. If a grasp has totally failed and therefore



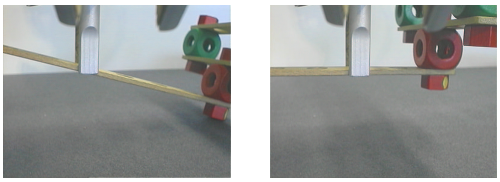
(a) (b)

Figure 3: *Sliding friction of the fingers result in rotation of the object.*

the first optimality condition cannot be met[2], a predefined penalty is given.

Position Learner. While the valuation technique for the orientation learner is primarily based on processing images from the camera sensors, the self-valuation of the position learner is primarily gained via a force/torque sensor. The two points for optimality of the global quality

[2]This occurs either when the orientation of the gripper does not permit to cover the object or the object slips out of the fingers while closing them.



(a) (b)

Figure 4: *Grips that are suboptimal according to the quality criteria.*

criteria, mentioned in Section 3.2, are taken into account for self-valuation in the following fashion:

Stable Grasp: A grip is unstable, for example, if the gripper is not strong enough to fix the object at a given position. Such a situation is shown in Fig. 4(a). This lift-up movement of the manipulator results in forces shown in Fig. 5. Nearly during the whole lift-up movement, the force in the direction of the approach vector of the gripper is approximately constant. At the moment when the object loses contact with the table (in this example at $4s$) the force rises to a higher value. These profiles can be analyzed and used for valuation of the learner, e.g. this situation is valued with a predefined high penalty to express that such grips are not desired.

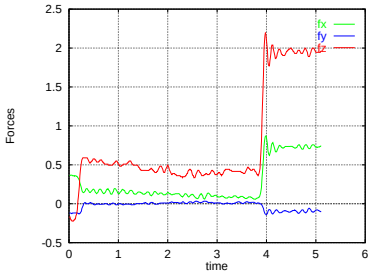


Figure 5: *Force profiles when the grip is not stable as shown in Fig. 4(a).*

When an object slips out of the fingers of the gripper the force in the direction of the approach vector of the gripper suddenly reduces to zero and the grasp can be considered a failure. Such a grasp is totally undesirable. Therefore, a constant high penalty is given to prevent the system from effecting such grips in the future.

Reducing torque: The goal of the position learner is to reduce torque within the fingers of the gripper. Fig. 4(b) shows an example of a grip that produces high torque. The torque profiles are shown in Fig. 6. Immediately after the beginning of the lift-up process, the torque around the normal vector of the gripper rises to a value considerably bigger than zero and stays constant while the object is being held. Torques are computed, and their negative values are directly used in the position learner. Here, no constant penalty is given because a grip with large torque is not necessarily bad, i.e. the system must have the possibility to distinguish between grasp points with different torques and choose the best among them.

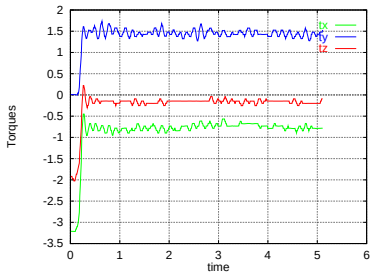


Figure 6: *Torque profiles of the grip in Fig. 4(b).*

5 Implementation of Learning

The learning scheme in our system is based on the *Sarsa* algorithm [7]. The general update formula computes the difference between the current and the next prediction of cumulative reward and updates the action-value function Q by a fraction of this difference as follows:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right]$$

As seen above, our system must handle continuous states and actions, i.e. angles, torques, etc. In such a situation we cannot provide a single value Q for every state-action pair but rather have to use a function approximator. Such a function approximator is of the form $Q_\omega(s, a)$, where $\omega = (\omega(1), \omega(2), \ldots, \omega(n))^T$ is a set of adjustable weights. The update of the current estimate of Q is performed by modifying the weights according to the following rule:

$$\Delta\omega_t = \alpha \left[r_{t+1} + \gamma Q_{t+1} - Q_t \right] \nabla_{\omega_t} Q_t \quad (1)$$

A general advantage of function approximators is that they are able to *generalize*. The system is able to estimate the expected return of state-action pairs that were never visited before. Although a function approximator can deal with continuous state and action spaces, it may not be able to accurately represent Q for the entire state and action space due to its finite resources.

We employ the B-spline function approximator [8] for the Q-function which is a natural generalization of coarse coding to continuously-valued features.

5.1 Approximating the Action-Value Function

In the following we define $\vec{x}$ as the concatenation of the current state $s = (s_1, \ldots, s_l)$ and the taken action $a =$

$(a_1, \ldots, a_m)$, that is: $\vec{x} = (s_1, \ldots, s_l, a_1, \ldots, a_m)$. The output for the B-spline function approximator which is the prediction of Q is computed by:

$$Q(\vec{x}) = \frac{\sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \left(c_{i_1,\ldots,i_n} \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j) \right)}{\sum_{i_1=1}^{l_1} \cdots \sum_{i_n=1}^{l_n} \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j)} = \sum_{i_1=1}^{l_1} \cdots \sum_{i_m=1}^{l_n} \left(c_{i_1,\ldots,i_n} \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j) \right)$$

where:
– x_j: the jth input ($j = 1, \ldots, n$),
– k_j: the order of the B-splines used for x_j,
– N_{i_j,k_j}^{j}: the ith B-spline of x_j,
– $i_j = 1, \ldots, l_j$: the index of the B-spline of x_j,
– l denotes the number of B-splines and
– $c_{i_1,i_2,\ldots,i_n}$: the *control vertices*[3].

This is called a *general NUBS hypersurface* , which possesses the following properties:

– If the B-splines of order $k_1, k_2, \ldots, k_n$ are employed to cover the spaces of the input variables $x_1, x_2, \ldots, x_n$, it can be guaranteed that the output variable y is $(k_j - 2)$ times continuously differentiable with respect to the input variables $x_j, j = 1, \ldots, n$.
– If the input space is partitioned fine enough and at the correct positions, the interpolation with the B-spline hypersurface can reach a given precision. Because the introduced weights ω of Q here correspond to the control vertices $c_{i_1,i_2,\ldots,i_n}$ of the B-spline function approximator, the gradient of Q with respect ω from Eqn. (1) is:

$$\nabla_\omega Q = \nabla_{c_{i_1,i_2,\ldots,i_n}} Q$$

Now the learning update from Equation (1) turns into the following formula:

$$\Delta c_{i_1,i_2,\ldots,i_n} = \alpha \left[r_{t+1} + \gamma Q_{t+1} - Q_t \right] \prod_{j=1}^{n} N_{i_j,k_j}^{j}(x_j)$$

Based on this, the control vertices are updated online after each grasping trial of the system.

5.2 Accumulating Trails

A practical problem that arises is that the system will learn a path from an initial state, i.e initial grasping configuration, up to a final state, i.e. a successful grip. To overcome this side effect in systems where the goal state itself is the most important outcome and not the path to the goal state, we propose an easy approach to increase the performance of such a learning system, called *accumulating trails*. When a learning system learns a type of path from an initial state to a final state, i.e. by applying a set of actions $a_0, \ldots a_n$ to s and its successors, it is sometimes possible to get to the same goal state if applying a set of actions $a'_0 \ldots a'_m$ to the state s and its successors, where $m < n$. That is to say, that one would reach the goal state $n - m$ steps earlier.

Let ψ denote the function applying an action a to a state s, denoted $\psi : \mathcal{A} \to (\mathcal{S} \to \mathcal{S})$, where $\mathcal{A}, \mathcal{S}$ are the total sets of actions and states, respectively. The outcome of this function, applying it to an action, is a function on the state space $\mathcal{S}$ called *action execution function*.

Using the definition above, each learning episode can be considered as a composition of functions $(A : \mathcal{S} \to \mathcal{S})$

$$A(s) = \psi(a_n) \circ \psi(a_{n-1}) \circ \cdots \circ \psi(a_0)(s)$$

where s is the starting state of the episode and a_i is the action applied in time step i. This function composition is further referred to as *sequence*.

A sequence B of action executions $\psi(b_m) \circ \cdots \circ \psi(b_0)$ is called a *sub-sequence* of sequence $A = \psi(a_n) \circ \cdots \circ \psi(a_0)$, if $A(s) = B(s)$:

$$\psi(a_n) \circ \cdots \circ \psi(a_0)(s) = \psi(b_m) \circ \cdots \circ \psi(b_0)(s), \; m \leq n$$

where s is the starting state. Then, sequence A is called *substitutable* through B. The sub-sequence B always produces the same resulting state as the sequence A. That means, if we start in state s it makes no difference whether we "follow" sequence B or sequence A. The state at the end of the sequence is always the same. If a sequence A is not substitutable through any other sequence B, it is called *final*. When the agent's intention is to reach the goal states as soon as possible, as for example in this work[4], the learning algorithm should converge to a situation of purely final sequences.

An accumulation function on action executions is defined as: $\oplus : (\mathcal{S} \to \mathcal{S}) \times (\mathcal{S} \to \mathcal{S}) \to (\mathcal{S} \to \mathcal{S})$. A sequence $A = \psi(a_n) \circ \cdots \circ \psi(a_0)$ of action executions is *accumulatable*, if

$$\psi(a_n) \oplus \psi(a_{n-1}) \oplus \cdots \oplus \psi(a_0) = B,$$

where B is the subsequence of A. The accumulation function describes how to combine action executions to produce shorter sequences. This function has to be defined according to the learning system one wants to develop. The accumulation is defined for action executions and not solely for actions, because it depends on the states if such an accumulation can be performed. In some situation the accumulation function is defined as follows:

$$\psi(a_k) \oplus \psi(a_l) = \psi(a_k \diamond a_l), \tag{2}$$

where $\diamond$ is a function $\diamond : \mathcal{A} \times \mathcal{A} \to \mathcal{A}$.

In most situations, the accumulation function must include a kind of model of the environment and this is only possible by also taking into account the states rather than only the actions as suggested by Equation (2). The agent

[3] Corresponding to *de Boor points* in CAGD.

[4] It is desirable to find an optimal grasp point as soon as possible.

must "know" in which situation it is possible to accumulate action executions and in which situation it is not. However, for some tasks Equation (2) is an easy and sufficient definition.

As an example, for application within the orientation learner the accumulation function $\diamond$ is defined as:

$$a_k \diamond a_l = \begin{cases} a_k + a_l & \text{if} \quad -90 \leq a_k + a_l \leq 90 \\ a_k + a_l + 180 & \text{if} \quad a_k + a_l < -90 \\ a_k + a_l - 180 & \text{if} \quad a_k + a_l > 90 \end{cases}$$

assuming that the actions of the orientation learner are rotational movements from the interval $[-90, \ldots, 90]$.

6 Generalization

The orientation learner is fully applicable to any kind of object, i.e. it provides a total generalization potential. In other work, where the learning process is not divided into two separate learners, the generalization is only partial. This results in slower learning phases for new objects. Propositions like "grasping at parallel edges is always good" cannot be made by such systems at all. Here, once the orientation learner has learned several grasping situations, it can be used with any kind of new object the robot is faced with.

In comparison, the position learner is more complicated. Because of different shapes of objects the global positions of the learned grasp points cannot be applied to every object. In most cases it is a better choice to initialize the learning parameters by values of a pre-learned similar object than a random initialization.

Three main subproblems must be solved:

1. In which situation can a pre-learned position learner be fully adopted to a new object?
2. When can a pre-learned position learner be used as a basis for a new object?
3. When must a completely new position learner be initiated?

6.1 Tree Distance as a Measure

Assuming a distance measure on the objects to grasp, we can improve our initial learning parameters as follows: Let $\mathcal{L}_s = \{(o_i, l_i) | i = 1 \ldots n\}$ be the set of tuples of n pre-stored objects o_i in tree notation, together with their stored position learners l_i, and $dist(o_i, o_k)$ the distance of the trees according to a distance measure. Then,

1. a pre-learned position learner l' of an object o' can be fully adopted to a new object o, if $\forall (o_i, l_i) \in \mathcal{L}_s \backslash (o', l')$:

   $$dist(o', o) \leq dist(o_i, o) \leq D_{min}; \tag{3}$$

2. a pre-learned position learner l' of an object o' can be used as basis for a new object o , if $\forall (o_i, l_i) \in \mathcal{L}_s \backslash (o', l')$:

   $$D_{max} \geq dist(o', o) \leq dist(o_i, o) > D_{min}; \tag{4}$$

3. a completely new position learner is initiated for a new object o, if $\forall (o_i, l_i) \in \mathcal{L}_s$

   $$dist(o_i, o) > D_{max}, \tag{5}$$

where D_{max} and D_{min} are adequate thresholds for accepting and refusing an object to be equal, respectively.

Simple distances on the objects' pixel data, like the *Hamming Distance*, are susceptible to noisy data and moreover do not consider the object structure. We restrict our observations to object structures that can be represented as a hierarchical relation. A simple tree encoding is used for the objects. These are rooted ordered trees, i.e. there exists an ancestor relation of nodes and the order of sibling nodes matters. In order to obtain a tree out of an object's image pixel data a medial axis transformation [9] is performed. The resulting graph is analyzed by a contour tracking process which transforms the medial axes into the corresponding trees. Fig. 7 gives examples of this process. The nodes of the shown trees contain length information about the corresponding parts of the medial axis.

6.2 Computational Results

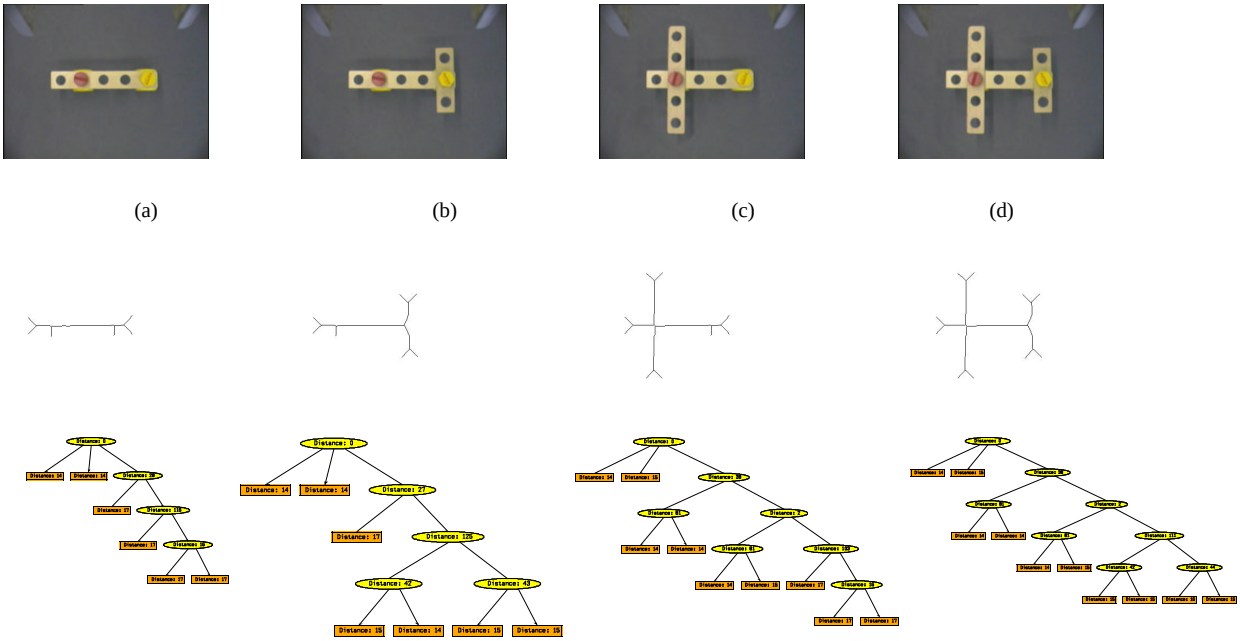
Distance models on rooted ordered trees based on editing operations are proposed in [10, 11]. We used the latter for our experiments. For instance, we learn to grasp the object shown in Fig. 7(a) without any a priori knowledge. Accordingly, a tree comparison with the object in Fig. 7(b) leads to Eqn. (4). For the object in Fig. 7(c) no object of sufficient similarity is stored in the database which leads to Eqn. (5). Finally, Eqn. (4) again suggests to grasp the object in Fig. 7(d) using the parameters of the object in Fig. 7(c).

In this manner one can determine the "most similar" object out of the set of pre-learned aggregates for a new object. If the similarity is strong enough, i.e. Eqn. (3) is met, the objects are treated the same and the same position learner is used.

7 Experiments and Results

The physical set-up of this system consists of the following components:

Main actuator: One 6-DOF PUMA-260 manipulator is installed overhead in a stationary assembly cell. On the wrist of the manipulator, a robot gripper with integrated force/torque sensor and "self-viewing" hand-eye system (local sensors) is mounted (Fig. 8). The robot is controlled by RCCL (*Robot Control C Library*).



(a) (b) (c) (d)

Figure 7: *Different objects, their medial axes and the resulting tree coding. At each node the length of the branch leading to that node is stored.*

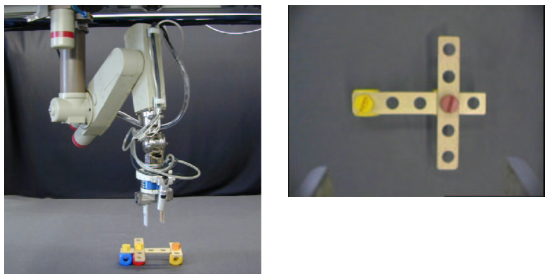


Figure 8: *The setup and a sample view of the hand-camera.*

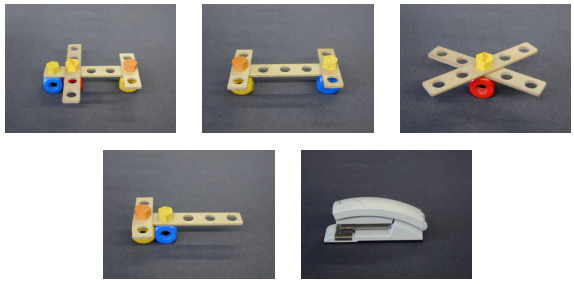
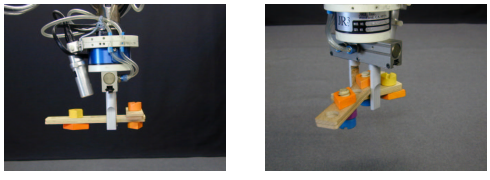


Figure 9: *Sample objects.*

Objects: Most kinds of objects are constructed from *Baufix* elements, wooden toys for children containing parts like screws, ledges and cubes. Therefore, these objects are also referred to as *aggregates*. An advantage of these parts is that one can very quickly construct several aggregates that can be tested with the system.

To get a uniform and matchable view of the objects the system learns to grasp, the manipulator initially moves itself over the object so that the x-axis of the camera's coordinate system appears parallel to the axis of least inertia of the object and the center of area in the right side of the image. The center of the object's bounding box coincides with the center of the image. An additional tool-transformation is performed, so that the camera is moved towards the working surface. Several objects were used to test the performance of the whole system. Some of them are shown in Fig. 9. The robot has found a good and stable grasp point for each object that fulfills the optimality conditions given above, most times near the object's center of gravity. Two special results of a grasping operation are shown in Fig. 10. In Fig. 10(a) the manipulator has grasped the object at a point different from the center of area but near the center of mass of the object. Fig. 10(b) shows a successful grasp at a convex edge of a different object. To show the generalization ability of



(a) (b)

Figure 10: *Successfully performed grasping operations.*

the orientation learner, it was first applied to a new object for a defined number of epoches. Thereafter, the same learner was used on a different object to show that the average steps until the goal state decrease much faster. The result is shown in Fig. 11. In the second part of the experiment the orientation learner did not start at a value of three where the initially performed orientation learner ended. This is due to the fact that in the first cycle a simple ledge was used and the learner still did not converge while in the second cycle a more complex object was used. However, one can see that in the second cycle the orientation learner was quicker. Only new states that do not occur on the simple ledge have to be learned additionally.

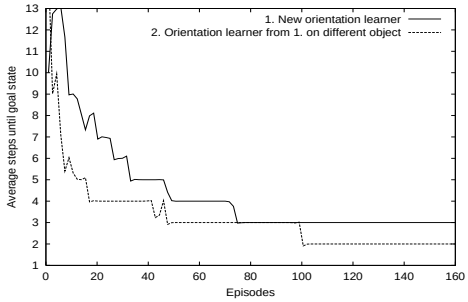


Figure 11: *Generalization of the orientation learner.*

8 Discussion and Ongoing Work

We presented a self-valuing learning system that is capable of grasping various kinds of objects. Our system consists of two learners based on local and global quality criteria. While the orientation learner is applicable to arbitrary objects and therefore fully generalizes between them, the position learner is mostly dependent on a special object and its physical properties. The generalization of the position learner is accomplished by a tree distance model on the shape of the object. The system shows the ability to grasp several kinds of objects and to generalize the learned faculties to new ones.

In our ongoing work, the system will be extended to handle grips in 3D. Therefore the tree coding for generalization has to be extended to represent the 3D-structure of an object. With additional sensors, e.g. a stereo camera vision system, the robot should examine the objects and place the grips from different orientations in space. We are also adapting the presented system to a multi-fingered robot hand. With such a hand a single grasp point is much more complex than with a parallel-jaw gripper. In this case a grasp point no longer consists of only two contact points on the object's surface, e.g. the three finger hand, as shown in Fig. 12, can perform a full 7 point form closure grasp. Furthermore, the possible actions of the learners are more challenging with a multi-fingered hand. The different fingers can move independently to a certain extent and apply different forces. However, the basic principle of two learners, based on local and global criteria, and the self-valuing approach can be maintained.

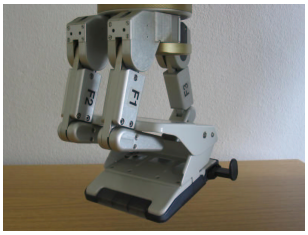


Figure 12: *Sample grip with a multi-fingered hand.*

References

[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000, pp. 348–353.

[2] G. Smith, E. Lee, K. Goldberg, K. Boehringer, and J. Craig, "Computing parallel-jaw grips," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999, pp. 1897–1903.

[3] P. J. Sanz A. Morales, G. Recatalá and Ángel P. del Pobil, "Heuristic vision-based computation of planar antipodal grasps on unknown objects," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2001, pp. 583–588.

[4] I. Kamon, T. Flash, and S. Edelman, "Learning to grasp using visual information," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996, pp. 2470–2476.

[5] J. Zhang, G. Brinkschröder, and A. Knoll, "Visuelles reinforcement-lernen zur feinpositionierung eines roboterarms "uber kompakte zustandskodierung," in *Tagungsband Autonome Mobile Robotersysteme (im Druck)*, München, 1999.

[6] J. Coelho, J. Piater, and R. Grupen, "Developing haptic and visual perceptual categories for reaching and grasping with a humanoid robot," 2000.

[7] Satinder P. Singh and Richard S. Sutton, "Reinforcement learning with replacing eligibility traces," *Machine Learning*, vol. 22, no. 1–3, pp. 123–158, 1996.

[8] J. Zhang and A. Knoll, "Constructing fuzzy controllers with B-spline models - principles and applications," *International Journal of Intelligent Systems*, vol. 13, no. 2/3, pp. 257–285, Februar/March 1998.

[9] H. Blum, "A transformation for extracting new descriptors of shape," in *Models for the Perception of Speech and Visual Form*, W. Wathen-Dunn, Ed., Cambridge, MA, 1967, pp. 362–380, M.I.T. Press.

[10] K. C. Tai, "The tree-to-tree correction problem," *J. Assoc. Comput. Mach.*, vol. 26, no. 3, pp. 422–433, 1979.

[11] L. Wang T. Jiang and K. Zhang, "Alignment of Trees - An Alternative to Tree Edit," *Theoretical Computer Science*, vol. 143, no. 1, pp. 137–148, 1995.

Using Motion Capture Techniques to Program High Degree of Freedom Humanoid Robot Movements

Aleš Ude

ATR Computational Neuroscience Laboratories
Dept. of Humanoid Robotics and Comput. Neurosc.
2-2-2 Hikaridai, Seika-cho, Soraku-gun
Kyoto 619-0288, Japan

Jožef Stefan Institute
Dept. of Automatics, Biocybernetics and Robotics
Jamova 39, 1000 Ljubljana
Slovenia

The formulation and optimization of joint trajectories for humanoid robots is quite different from this same task for standard robots because of the complexity of humanoid robots' kinematics and dynamics. Movements for such robots are difficult to generate due to the large number of joints, coupling between joints, and redundancies. In addition, people that interact with humanoid robots expect that they move like humans. Our approach to the programming of such movements is to exploit the similarities between humanoid robots and humans to generate appropriate robot trajectories. We take advantage of the nature of the tasks humanoid robots are asked to perform, which typically involve making human-like motion.

Our approach involves capturing full body motions of a human performer using a suitable measurement device. In particular, we used an optical tracking device, which provides the 3D location of identified active markers that are currently in view. We have also experimented with goniometer devices strapped to the performer, and magnetic systems that provide marker orientation as well as location. Our goniometer-based system SenSuit from Sarcos is worn like an exoskeleton and can measure motion directly in the joint space. The main disadvantage of such a system is that it is designed for a specific robot and comes at a specific size, so that only people of proper height can wear it. For example, when a professional dancer came to our institute to teach our humanoid robot an Okinawan folk dance, it turned out that she couldn't put on the SenSuit because she was too short. Magnetic and marker-based optical systems are comparable. Magnetic sensors can measure both position and orientation and can handle occlusions better than optical systems. They are, however, sensitive to metallic objects and noisier than optical systems. The techniques presented in this talk can be partially extended to these different types of motion capture systems. We are also extending our techniques to vision systems where there are no markers, but individual pixels must be matched and accounted for.

We break up the problem into three parts: 1) identifying a kinematic model of the human being observed, 2) estimating the joint angle trajectories of the particular motion to be imitated, and 3) transforming the motion so that it is appropriate for the kinematics of the robot. Our contributions include the development of an automatic approach to identify a kinematic model of a human, which is useful also for the perception of human motion from video. The proposed technique involves measuring marker positions at a zero configuration and over a repertoire of motions exercising all relevant degrees of freedom. No manual measurement of the performer's limb lengths is necessary. The structure of the generated kinematic model is kept the same as the kinematic structure of the humanoid robot under consideration, but the joint axis positions are scaled to the lengths of the human performer by optimization. We exploited the sparseness of the Jacobian matrices to solve the resulting optimization problems efficiently.

Our approach to the formulation and optimization of joint trajectories for humanoid robots is based on B-spline wavelets. We demonstrated that B-spline wavelets are suitable for the representation of humanoid robots' trajectories at different resolution levels and showed how to resolve the resulting large-scale optimization problems to compute such trajectories. The ability to treat large-scale optimization problems that need to be solved to generate optimal full-body motions and to automatically infer the appropriate resolution level draws a distinction between our approach and other approaches proposed for human motion capture in the computer graphics literature. We were able to process different motion sequences and to generate both computer animations and humanoid robot motions using these approaches.

Finally, a robust optimization framework for human body tracking and motion recovery from video will be presented in this talk. The developed system is resistant to occlusions and demonstrates that it is possible to treat different problems arising in human motion analysis in a unified manner without using many decision thresholds. The implemented system requires only a standard CCD camera and no special markers on the body, but it is at the moment limited to simpler movements than marker-based systems.

Learning from Demonstration and Adaptation of Biped Locomotion with Dynamical Movement Primitives

Jun Nakanishi*[1], Jun Morimoto[1], Gen Endo[1,2], Gordon Cheng[1], Stefan Schaal[1,3] and Mitsuo Kawato[1]

[1]ATR Computational Neuroscience Laboratories, Kyoto 619–0288, Japan
[2]Sony Intelligent Dynamics Laboratory, Tokyo 141–0001, Japan
[3]University of Southern California, Los Angeles, CA 90089-2520, USA

***Abstract*— In this paper, we report on our research for learning biped locomotion from human demonstration. Our ultimate goal is to establish a design principle of a controller in order to achieve natural human-like locomotion. We suggest dynamical movement primitives as a CPG of a biped robot, an approach we have previously proposed for learning and encoding complex human movements. Demonstrated trajectories are learned through the movement primitives by locally weighted regression, and the frequency of the learned trajectories is adjusted automatically by a novel frequency adaptation algorithm based on phase resetting and entrainment of oscillators. Numerical simulations demonstrate the effectiveness of the proposed locomotion controller.**

I. INTRODUCTION

There has been a growing interest in biped locomotion with the recent development of humanoid robots. Many of existing successful walking algorithms use the zero moment point (ZMP) criterion [21] for motion generation with off-line planning [9], [20] and on-line balance compensation [5], [8], [23]. These ZMP methods have been shown to be effective to guarantee point-wise stability of biped locomotion. However, they require precise modelling of robot dynamics and high-gain trajectory tracking control, and the generated patterns result in a typical "bent-knee" posture to avoid singularities. From the viewpoint of energy efficiency, such walking patterns are not desirable since torque must be continuously applied to the knee joint to maintain a bent-knee posture. The previous ZMP approaches have primarily focused on stability during walking rather than natural human-like motion which exploits passive dynamics of the body.

In contrast to off-line trajectory planning, biologically-inspired control approaches based on central pattern generators (CPGs)[1] with neural oscillators have been drawing much attention for rhythmic motion generation. As a CPG, a neural oscillator proposed by Matsuoka [12] is widely used, which models the firing rate of two mutually inhibiting neurons described in a set of differential equations. This model is used in robotic applications to achieve designated tasks involving rhythmic motion which requires interactions between the system and the environment. Examples include biped locomotion [4], [19], quadruped locomotion [3], juggling [13], drumming [11], and playing with a slinky toy [22]. Neural oscillators have desirable properties such as adaptation to the environment through entrainment. However, it is difficult to design robust controllers with coupled oscillators, and to manually tune all open parameters to achieve a desired behavior.

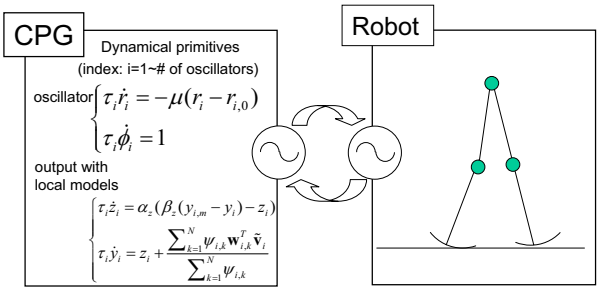


Fig. 1. Proposed control strategy: CPG with dynamical movement primitives and the robot.

In this paper, we suggest an approach to learning biped locomotion from human demonstration and its adaptation through coupling between the pattern generator and the mechanical system. Motivated by human's capability of learning and imitating a demonstrated movement by others, imitation learning has been explored as an efficient method for motor learning in robots to accomplish the desired movement [16], [17]. Previously, Ijspeert, Nakanishi and Schaal have proposed a method to encode complex discrete and rhythmic multijoint movements through imitation learning as movement primitives [6], [7]. Kinematic movement plans are described in a set of nonlinear differential equations with well-defined attractor dynamics, and demonstrated trajectories are learned using

*Email: jun@atr.co.jp

[1]The term CPG is widely used, but a distributed pattern generator (DPG) may be more appropriate, since, in many robotic applications, a distributed architecture which consists of coupled oscillators is generally used for pattern generators. In this paper, we shall use the classic term CPG although we consider much more a DPG-like distributed architecture.

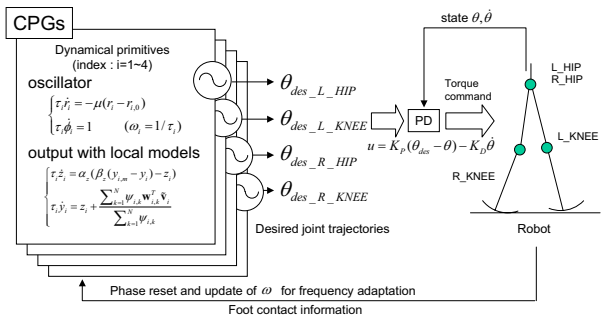


Fig. 2. Robot controller with dynamical movement primitives.

locally weighted regression. In this paper, we present the idea of using rhythmic movement primitives [7] as a CPG to achieve natural human-like walking in biped robots. Figure 1 depicts a conceptual architecture of the proposed control system. The dynamical movement primitive [7] has various desirable properties which are beneficial for biped locomotion—for example, it can learn a demonstrated trajectory rapidly, and it is easy to re-scale the learned rhythmic movement in terms of amplitude, frequency and offset of the patterns. In this work, we also propose an adaptation algorithm for the frequency of walking based on phase resetting [10] and entrainment between the phase oscillator and mechanical system using feedback from the environment. We present numerical simulations to demonstrate the effectiveness of the proposed control strategy.

II. Learning Biped Locomotion from Human Demonstration

A. Rhythmic Dynamical Movement Primitives

We briefly review the rhythmic dynamical movement primitives proposed in [7], which we will use as a CPG for biped locomotion in this paper. Consider the following limit cycle oscillator characterized in terms of an amplitude r and a phase ϕ as a canonical dynamical system which generates basic rhythmic patterns:

$$\tau\dot{\phi} = 1 \tag{1}$$

$$\tau\dot{r} = -\mu(r - r_0) \tag{2}$$

where τ is a temporal scaling factor, r_0 determines the desired (relative) amplitude, and μ is a positive constant. Note that the phase dynamics (1) can be written as

$$\dot{\phi} = \omega \tag{3}$$

where $\omega \stackrel{\text{def}}{=} 1/\tau$ is the natural frequency. This rhythmic canonical system is designed to provide an amplitude signal $\tilde{\mathbf{v}} = [r\cos\phi, r\sin\phi]^T$ and phase variable $\mathrm{mod}(\phi, 2\pi)$ to the the following second order dynamical system (z, y), where the output y is used as the desired trajectory for the robot.

$$\tau\dot{z} = \alpha_z(\beta_z(y_m - y) - z) \tag{4}$$

$$\tau\dot{y} = z + f(\tilde{\mathbf{v}}, \phi) \tag{5}$$

where α and β are time constants, y_m is an offset of the output trajectory. f is a nonlinear function approximator using locally linear models [15] of the form

$$f(\tilde{\mathbf{v}}, \phi) = \frac{\sum_{k=1}^{N} \Psi_k \mathbf{w}_k^T \tilde{\mathbf{v}}}{\sum_{i=k}^{N} \Psi_k}. \tag{6}$$

where $\mathbf{w}_k$ is the parameter vector of the k-th local model which will be determined by locally weighted learning [15] from a demonstrated trajectory y_{demo} (see Section II-C.2). Each local model is weighted by a Gaussian kernel function

$$\Psi_k = \exp(-h_k(\mathrm{mod}(\phi, 2\pi) - c_k)^2) \tag{7}$$

where c_k is the center of the k-th linear model, and h_k characterizes its width. A final prediction is calculated by the weighted average of the predictions of the individual models. As demonstrated in [7], the amplitude, frequency and offset of the learned rhythmic patterns can be easily modified by scaling the parameters r_0, $\omega(= 1/\tau)$ and y_m individually.

B. Rhythmic Dynamical Movement Primitives as a CPG

We use the rhythmic dynamical movement primitives introduced above as a CPG. Figure 2 illustrates the proposed control architecture in this paper. Each joint is equipped with a movement primitive which generates the desired joint trajectory θ_{des}. We define the index and the corresponding name of the joint as Left hip (i=1, L_HIP), and Left knee (i=2, L_KNEE), Right hip (i=3, R_HIP), and Right knee (i=4, R_KNEE). In this setting, each degree of freedom (DOF) has its own oscillator, however, different allocation of oscillators can be considered, e.g., a unique oscillator for the whole CPG or one oscillator for each leg. We will address this design issue in our future work. A low-gain PD controller is used for each joint to track the desired trajectory which is the output of the movement primitive, and ground contact information is fed back to the CPG in order to reset the phase and adjust the natural frequency of the oscillators. At heel contact, the phase of all the oscillators is reset to $\phi = 0$ for the stance leg and to $\phi = \pi$ for the swing leg respectively at the same time. Thus, the phase difference between the oscillators for the left leg and the right leg is kept π rad. The update law for the frequency adaptation of locomotion will be discussed in Section III-B in detail.

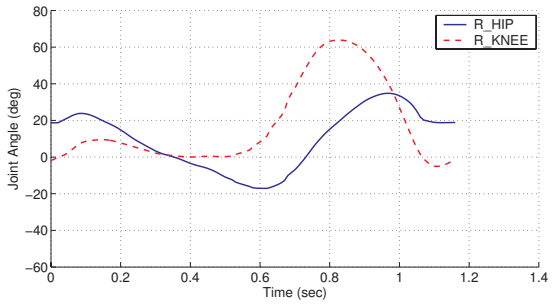


Fig. 3. Extracted one period of joint trajectory data of human walking presented in [2] as used for learning in this paper (R_HIP and R_KNEE).

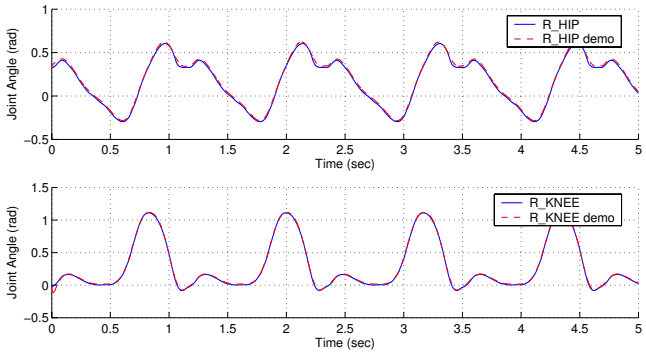


Fig. 4. Learning result of human's walking trajectories using dynamical movement primitives. The learned trajectories (output of the dynamical primitives) nearly coincide the demonstrated trajectories.

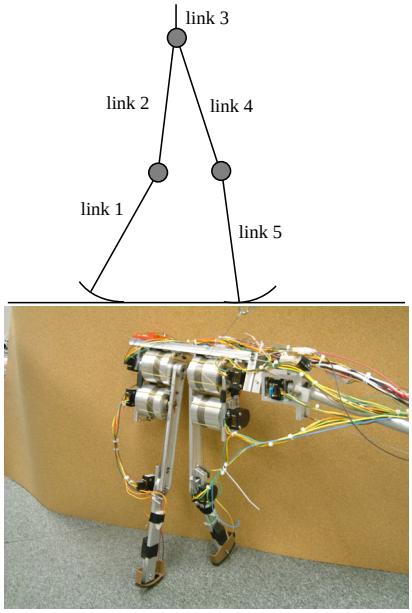


Fig. 5. Top: Five-link model of the robot. Bottom: physical system whose dynamics are simulated in the numerical studies.

TABLE I

PHYSICAL PARAMETERS OF THE ROBOT MODEL

	link1	link2	link3	link4	link5
mass [kg]	0.05	0.43	1.0	0.43	0.05
length [m]	0.2	0.2	0.01	0.2	0.2
inertia ($\times 10^{-4}$ [kg·m])	1.75	4.29	4.33	4.29	1.75

C. Learning from Human Demonstration

1) Human's Walking Pattern: As a demonstrated trajectory, we use the recorded joint data of human walking in the book [2] (29-year-old male, 173cm, 83.5kg, right hip and knee). In the future, we plan to measure human walking under various conditions by ourselves using our motion capture equipment. Figure 3 shows the extracted trajectory data of the right hip and knee joints for one period of locomotion from [2]. In the next section, we will use these joint trajectories as human demonstration for the learning of biped locomotion. We identified the period and frequency of this patten by the power spectrum estimation with FFT and autocorrelation as $T = 1.17$ sec and $f = 1/T = 0.855$ Hz respectively.

2) Learning with Locally Weighted Regression: We briefly explain how we find the parameters $\mathbf{w}_k$ in (6) by locally weighted learning [15] for a given demonstrated trajectory y_{demo}. Given a sampled data point $(f_{target}, \tilde{\mathbf{v}})$ at t where

$$f_{target} = \dot{y}_{demo} - \beta(y_m - y_{demo}) \tag{8}$$

the learning problem is formulated to find the parameters $\mathbf{w}_k$ in (6) using incremental locally weighted regression technique [15] in which $\mathbf{w}_i$ is updated by

$$\mathbf{w}_k^{t+1} = \mathbf{w}_k^t + \mathbf{P}_k^{t+1}\tilde{\mathbf{v}}e_k \tag{9}$$

where

$$\mathbf{P}_k^{t+1} = \frac{1}{\lambda}\left(\mathbf{P}_k^t - \frac{\mathbf{P}_k^t\tilde{\mathbf{v}}\tilde{\mathbf{v}}^T\mathbf{P}_k^t}{\frac{\lambda}{\Psi_k} + \tilde{\mathbf{v}}^T\mathbf{P}_k^t\tilde{\mathbf{v}}}\right),\ e_k = f_{target} - \mathbf{w}_k^T\tilde{\mathbf{v}}$$

and $\lambda \in [0,1]$ is a forgetting factor. We chose this locally weighted regression framework as it can automatically find the correct number of necessary basis function, and can tune the h_k parameters of each Gaussian kernel function (7) to achieve higher function approximation accuracy. Moreover, it learns the parameters $\mathbf{w}_k$ of every local model k totally independent of all other local models, which minimizes interference between local models. Figure 4 shows the learning result of the demonstrated trajectories using the rhythmic dynamical primitives. In the book [2], only the trajectory data of the right leg are provided. Thus, we generate the desired trajectory for the left leg by shifting the phase of the oscillator of the right leg by π.

TABLE II

PARAMETERS USED IN THE SIMULATIONS

	Parameter	Description	Value used
Dynamical primitives	r_0	(relative) amplitude	0.7 for all joints
	ω	natural frequency	updated by (19)
	y_m	offset	Hip: $y_m = 0.0$ and Knee: $y_m = 0.35$
PD gains	K_P	position gain	Hip: $K_P = 8.0$ and Knee: $K_P = 6.0$
	K_D	velocity gain	0.05 for all joints
Phase resetting	ϕ_i	phase	$\phi_1 = \phi_2 = 0, \phi_3 = \phi_4 = \pi$ at left leg heel strike
			$\phi_1 = \phi_2 = \pi, \phi_3 = \phi_4 = 0$ at right leg heel strike

III. FREQUENCY ADAPTATION OF LOCOMOTION VIA ENTRAINMENT OF PHASE OSCILLATOR

A. Synchronization of Coupled Phase Oscillators

1) Entrainment with Phase Coupling: This section reviews basic properties of coupled oscillators [18]. Consider the following dynamics of two coupled oscillators as depicted in Figure 6

$$\dot{\phi}_1 = \omega_1 + K_1(\phi_2 - \phi_1) \tag{10}$$
$$\dot{\phi}_2 = \omega_2 + K_2(\phi_1 - \phi_2) \tag{11}$$

where $\omega_1, \omega_2 > 0$ are natural frequencies of the oscillators, and K_1, K_2 are positive coupling constants. Define the phase difference ψ as $\psi = \phi_2 - \phi_1$, and consider its dynamics

$$\dot{\psi} = (\omega_2 - \omega_1) - (K_1 + K_2)\psi. \tag{12}$$

Then, we see that there is a stable fixed point at

$$\psi^* = \frac{\omega_2 - \omega_1}{K_1 + K_2}. \tag{13}$$

As a result, these oscillators run at the same frequency (called coupled frequency) given by

$$\omega^* = \frac{K_2\omega_1 + K_1\omega_2}{K_1 + K_2} \tag{14}$$

with phase difference ψ^*.

2) Synchronization with Frequency Adaptation: In the development above, the oscillators run with the phase difference $\psi = \phi_2 - \phi_1 = \frac{\omega_2 - \omega_1}{K_1 + K_2}$ given ω_1 and ω_2 when they are entrained. Suppose $\omega_1 = \omega_2$, then the phase difference of these oscillators will be zero. Thus, when $\omega_2 = const.$ is given, we introduce a coupling dynamics of the natural frequency for ω_1 in (16) in addition to the phase coupling in order to achieve synchronization of these oscillators with zero phase difference.

$$\dot{\phi}_1(t) = \omega_1(t) + K_1(\phi_2(t) - \phi_1(t)) \tag{15}$$
$$\dot{\omega}_1(t) = -K(\omega_2 - \omega_1(t)) \tag{16}$$
$$\dot{\phi}_2(t) = \omega_2 + K_2(\phi_1(t) - \phi_2(t)) \tag{17}$$

where K is a positive constant. It is straightforward to see that $\omega_1 \to \omega_2$ asymptotically. Thus, the phase difference will be zero as $\psi = \phi_2 - \phi_1 \to 0$.

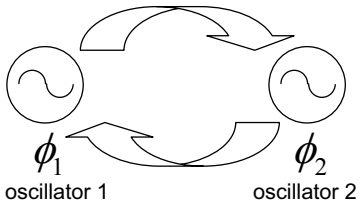


Fig. 6. A coupled phase oscillator system

B. Frequency Adaptation of Locomotion

As depicted in Figure 1, we see that the proposed control system can be regarded as a coupling of the CPG and the mechanical oscillator (robot), which can be modelled analogous to the coupled oscillator system above. Thus, it is natural to introduce such an adaptation mechanism to our dynamical primitives in order to achieve frequency adaptation of the learned periodic motions by the robot itself through the interaction among the CPG, robot and environment.

Consider the following update law of the phase and frequency of the oscillator in the dynamical movement primitives at the instance of heel strike

$$\dot{\phi} = \hat{\omega}^n + \delta(t - t_{heel\ strike})(\phi^{robot}_{heel\ strike} - \phi) \tag{18}$$
$$\hat{\omega}^{n+1} = \hat{\omega}^n + K(\omega^n_{measured} - \hat{\omega}^n) \tag{19}$$

where δ is the Dirac's delta function, n is the number of steps, and $\phi^{robot}_{heel\ strike}$ is the phase of the mechanical oscillator (robot) at heel strike defined as $\phi^{robot}_{heel\ strike} = 0$ at the heel strike of the leg with the corresponding oscillator, and $\phi^{robot}_{heel\ strike} = \pi$ at the heel strike of the other leg. $\omega^n_{measured}$ is the measured frequency of locomotion defined by

$$\omega^n_{measured} = \frac{\pi}{T^n_{measured}} \tag{20}$$

where $T^n_{measured}$ is the time for one step of locomotion (half period with respect to the oscillator). Note that (18) introduces phase resetting to the oscillator at heel strike, and (19) is the discretized version of (16).

IV. Numerical Simulations

In this paper, we present numerical simulations to illustrate the effectiveness of the proposed control algorithm.

A. Robot Model

In the numerical simulations, we use the model of the planar 5-link biped robot [14] depicted in Figure 5. The height of the robot is 40cm and the weight is about 3kg. Kinematic and dynamic parameters of the simulated robot are chosen to match those of the physical system (see Table I). We assume that the motion of the robot is constrained on the sagittal plane. The dynamics of the robot are derived using SD/FAST[2] and integrated using the Runge-Kutta algorithm at 1ms step size. The ground contact force is calculated using a linear spring-damper model.

B. Simulation Results

It is necessary to properly scale the learned trajectories from human demonstration since they cannot be directly applied for the robot model with different dimensions. In the following simulations, the parameters of the dynamical movement primitives and gains of the PD controller are determined empirically as listed in Table II to achieve stable walking. We manually designed the desired trajectory for the initial step from the standing position, and the CPG controller is activated at heel contact of the first step. For the scaling of the natural frequency of the oscillator, the adaptation law proposed in Section III-B is used. Figures 7 and 8 shows the desired and actual joint trajectories[3] for $t = 0 \sim 10$ sec. Figure 9 illustrates the desired and actual joint trajectories, and the timing of heel strke for the left leg. Figure 10 shows the torque command for the left leg, which indicates that the knee joint swings passively since it requires almost no torque (see $t = 14.8 \sim 15.0$ sec).

C. Frequency Adaptation of Locomotion

We present simulation results of the frequency adaptation algorithm proposed in Section III-B. The frequency of the all the oscillators are updated by (18) and (19) at heel contant. Figure 12 (left) depicts the duration for one step and Figure 12 (right) shows the learning curve of the frequency of the CPG with different coupling constants $K = 0.2, 0.5$ and 0.8 in (19) when the initial value is set to $\omega_0 = 4.78$ rad/s (period of oscillation is 1.5 sec). The simulation results demonstrate that robust self-adaptation of the frequency of locomotion is achieved by the proposed algorithm through entrainment. The resultant frequency was $\omega = 8.120$ rad/s. This result may be interpreted as follows: Given the walking frequency of the

[2] `http://www.sdfast.com`

[3] Note that the sign of the trajectories for the hip joints (L_HIP, R_HIP) is opposite to the human demonstration due to the definition of coordinate system.

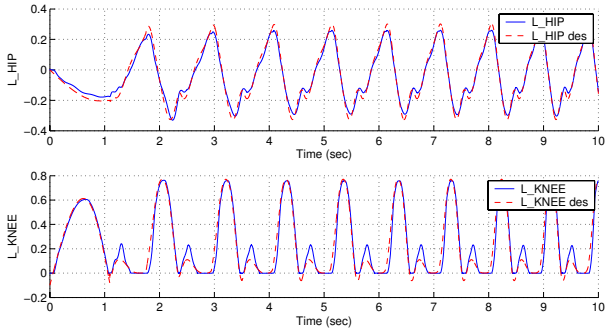


Fig. 7. Joint trajectories of the robot simulation (left leg)

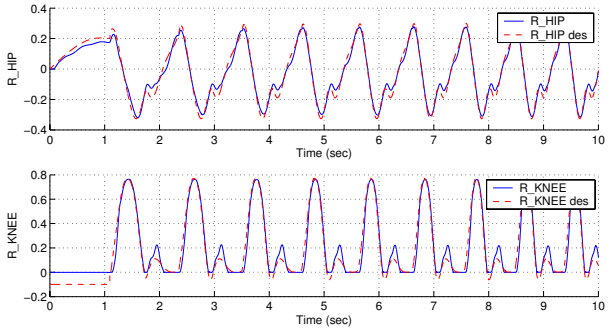


Fig. 8. Joint trajectories of the robot simulation (right leg)

human, ω_{human}, the leg length of the human, l_{human} , and the leg length of the robot, l_{robot}, as depicted in Figure 13, it may be natural to think of the scaling law

$$\hat{\omega}_{robot} = \omega_{human}\sqrt{\frac{l_{robot}}{l_{human}}} \tag{21}$$

which is derived from the ratio of the natural frequency of the simplified linear pendulum. In this paper, l_{human} can be considered as $l_{human} = 1.76 \times 0.49 = 0.86$m since the height of the human subject is 1.76m and it is anatomically known that the leg length is about 49% of the body height [1]. Thus, using the scaling law (21), we can estimate the frequency of locomotion of the robot with $l_{robot} = 0.4$m as

- Frequency: $\hat{\omega}_{robot} = 7.87$ rad/s
- Time for one step: 0.399 sec

As a result of the simulation of frequency adaptation, we obtained

- Frequency: $\hat{\omega}_{robot} = 8.120$ rad/s
- Time for one step: 0.387 sec.

The difference in the frequencies above is roughly 3%. Thus, simple analysis may suggest that the proposed frequency adaptation algorithm achieves the natural frequency of the coupled system through entrainment, i.e., a simple form of resonance tuning.

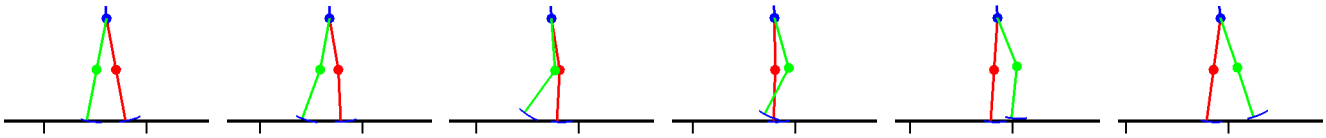


Fig. 11. Snapshots of walking for one step at 15 frames/sec (1 frame ≈ 66msec)

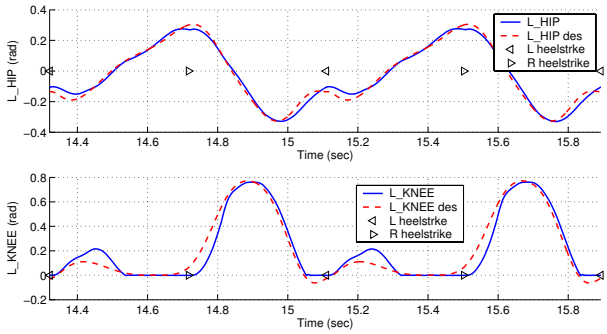


Fig. 9. Joint trajectories for the left leg and heel strike timing of the simulation for two period (4 steps) of walking.

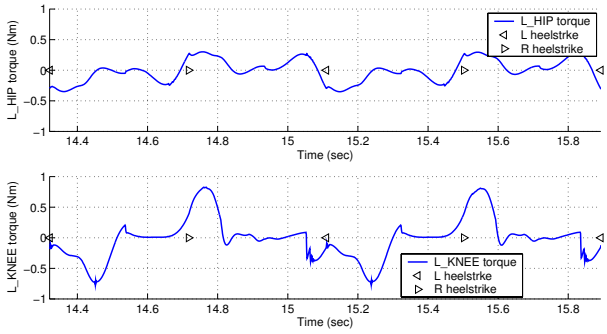


Fig. 10. Torque command to the left hip and knee joints for two period (4 steps) of walking.

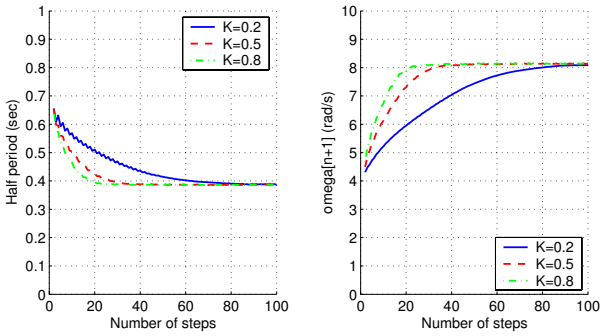


Fig. 12. Frequency adaptation of walking via entrainment.

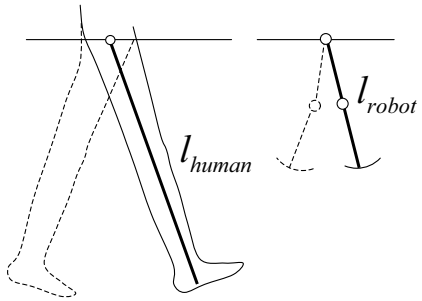


Fig. 13. A simplified pendulum model of the leg with different link length

V. Summary

In this paper, we proposed a method for learning biped locomotion from human demonstration and its frequency adaptation using the dynamical movement primitives. In the dynamical movement primitives, kinematic movement plans are described in a set of nonlinear differential equations with well-defined attractor dynamics, and demonstrated trajectories are learned using locally weighted regression. Specifically, we use rhythmic dynamical movement primitives as a CPG, and introduced a frequency adaptation algorithm through interactions among the CPG, mechanical system and environment. Numerical simulations illustrate the effectiveness of the proposed control algorithm: within a few seconds of walking, the simulation discovered an energy efficient walking frequency, roughly at the natural frequency of the combined robot-oscillator-environment system.

Future work will address intra- and interlimb coordintaion by introducing coupling among oscillators, and recovery from external perturbations. We also consider experimental implementation of the proposed algorithm on our biped robot, and collection of human's walking data under various behavioral conditions. In the long run, we are hopeful that this approach may provide insight into a theoretically sound design principle of biped locomotion control to achieve human-like natural walking.

VI. REFERENCES

[1] Dempster, W. T. and Gaughran, G. R. L., "Properties of body segments based on size and weight," *American Journal of Anatomy*, vol. 120, pp. 33–54, 1965.

[2] Ehara, Y. and Yamamoto, S., *Introduction to Body-Dynamics—Analysis of Gait and Gait Initiation*, Ishiyaku Publishers, 2002, in Japanese.

[3] Fukuoka, Y., Kimura, H., and Cohen, A. H., "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *International Journal of Robotics Research*, vol. 22, no. 3–4, pp. 187–202, 2003.

[4] Hase, K. and Yamazaki, N., "Computational evolution of human bipedal walking by a neuro-musculoskeletal model," *Artificial Life and Robotics*, vol. 3, pp. 133–138, 1999.

[5] Hirai, K., Hirose, M., Haikawa, Y., and Takenaka, T., "The development of honda humanoid robot," In *IEEE International Conference on Robotics and Automation*, pages 1321–1326, 1998.

[6] Ijspeert, A., Nakanishi, J., and Schaal, S., "Movement imitation with nonlinear dynamical systems in humanoid robots," In *IEEE International Conference on Robotics and Automation (ICRA2002)*, pages 1398–1403, 2002.

[7] Ijspeert, A., Nakanishi, J., and Schaal, S., "Learning attractor landscapes for learning motor primitives," In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*. MIT-Press, 2003.

[8] Kagami, S., Kanehiro, F., Tamiya, Y., Inaba, M., and Inoue, H., *AutoBalancer: An Online Dynamic Balance Compensation Scheme for Humanoid Robots*, A K Peters, Ltd., 2001.

[9] Kagami, S., Kitagawa, T., Nishiwaki, K., Sugihara, T., Inaba, M., and Inoue, H., "A fast dynamically equilibrated walking trajectory generation method of humanoid robot," *Autonomouns Robots*, vol. 12, pp. 71–82, 2002.

[10] Kawato, M., "Transient and steady state phase response curves of limit cycle oscillators," *Journal of Mathematical Biology*, vol. 12, pp. 13–30, 1981.

[11] Kotosaka, S. and Schaal, S., "Synchronized robot drumming by neural oscillator," In *Proceedings of the International Symposium on Adaptive Motion of Animals and Machines*, 2000.

[12] Matsuoka, K., "Sustained oscillatons generated by mutually inhibiting neurons with adaptation," *Biological Cybernetics*, vol. 52, pp. 367–376, 1985.

[13] Miyakoshi, S., Yamakita, M., and Furuta, K., "Juggling control using neural oscillators," In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1186–1193, 1994.

[14] Morimoto, J., Zeglin, G., and Atkeson, C. G., "Minimax differential dynamic programming: Application to a biped walking robot," In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, to appear.

[15] Schaal, S. and Atkeson, C. G., "Constructive incremental learning from only local information," *Neural Computation*, vol. 10, no. 8, pp. 2047–2084, 1998.

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

[17] Schaal, S., Ijspeert, A., and Billard, A., "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.

[18] Strogatz, S. H., *Nonlinear dynamics and chaos: with applications to physics*, Addison-Wesley, 1994.

[19] Taga, G., "Nonlinear dynamics of the human motor control - real-time and anticipatory adaptation of locomotion and development of movements," In *Proceedings of the International Symposium on Adaptive Motion of Animals and Machines*, 2000.

[20] Takanishi, A., Tochizawa, M., Karaki, H., and Kato, I., "Dynamic biped walking stabilized with optimal trunk and waist motion," In *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 561–566, 1989.

[21] Vukobratović, M., Borovac, B., Surla, D., and Stokić, D., *Biped Locomotion—Dynamics, Stability, Control and Application*, Springer-Verlag, 1990.

[22] Williamson, M. M., "Neural control of rhythmic arm movements," *Neural Networks*, vol. 11, pp. 1379–1394, 1998.

[23] Yamaguchi, J., Takanishi, A., and Kato, I., "Development of a biped walking robot compensating for three-axis moment by trunk motion," In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 187–192, 1993.

Imitation of Human-Demonstrated Movements with Nonlinear Dynamical Systems in Humanoid Robots

Auke Jan Ijspeert [1], Jun Nakanishi [2], Stefan Schaal [3]

(1) Biologically Inspired Robotics Group
School of Informatics & Communication, Lausanne, Switzerland.
auke.ijspeert@epfl.ch
http://lslwww.epfl.ch/birg

(2) ATR Human Information Science Laboratories, Kyoto 619-0288, Japan

(3) University of Southern California, Los Angeles, CA 90089-2520, USA

Many control problems take place in continuous state-action spaces, e.g., as in manipulator robotics, where the control objective is often defined as finding a desired trajectory that reaches a particular goal state. While reinforcement learning offers a theoretical framework to learn such control policies from scratch, its applicability to higher dimensional continuous state-action spaces remains rather limited to date. Instead of learning from scratch, we suggest to learn a desired complex control policy by transforming an existing simple canonical control policy. For this purpose, we represent canonical policies in terms of differential equations with well-defined attractor properties. By nonlinearly transforming the canonical attractor dynamics using techniques from nonparametric regression, almost arbitrary new nonlinear policies can be generated without losing the stability properties of the canonical system.

We demonstrate our techniques in the context of learning a set of movement skills for a humanoid robot from demonstrations of a human teacher. Policies are acquired rapidly, and, due to the properties of well-formulated differential equations, can be re-used and modified on-line under dynamic changes of the environment. The linear parameterisation of nonparametric regression moreover lends itself to recognize and classify previously learned movement skills. Evaluations in simulations and on an actual 30 degree-of-freedom humanoid robot exemplify the feasibility and robustness of our approach. Movies demonstrating how the system can be used to learn movements such as tennis swings and drumming beats with the humanoid robot will be shown.

Ijspeert A.J., Nakanishi J., Schaal S.: **Learning attractor landscapes for learning motor primitives**, *Advances in Neural Information Processing Systems 15*. NIPS 2003. Becker S., Thrun S., Obermayer K. (Eds), to appear.

Ijspeert A.J., Nakanishi J., Schaal S.: **Learning Rhythmic Movements by Demonstration using Nonlinear Oscillators**, *Proceedings of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems* 2002, pp 958-963.

Ijspeert A.J., Nakanishi J., Schaal S.: **Movement imitation with nonlinear dynamical systems in humanoid robots**, *Proceedings of the IEEE International Conference on Robotics and Automation*, 2002, pp 1398-1403 (received the ICRA2002 best paper award).

Determining What to Imitate in a Manipulation Task

Aude G. Billard
Autonomous Systems Lab
School of Engineering, EPFL, Lausanne, Switzerland.
aude.billard@epfl.ch; http://asl.epfl.ch

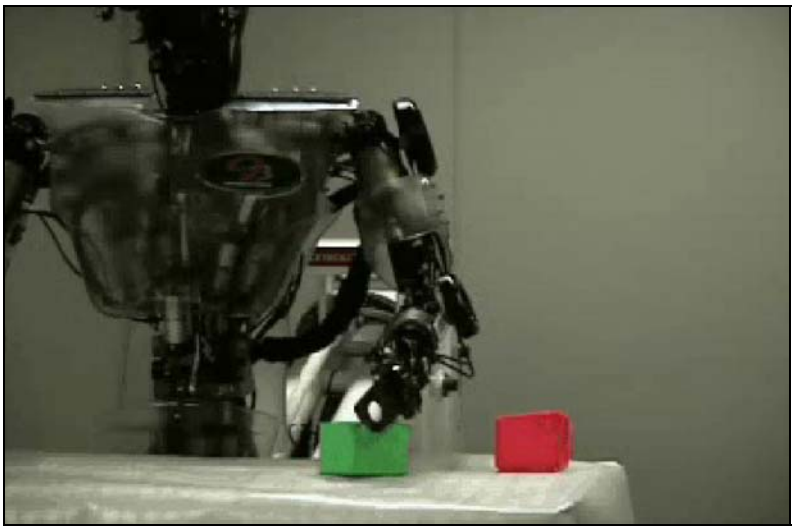
An essential problem of imitation is that of determining "what to imitate" [2,7], i.e. to determine which of the many features of the demonstration are relevant to the task and which should be reproduced. When requested to imitate a manipulation task, children tend to follow a hierarchy of goals [3] reproducing first the goal of the motion (e.g. grasping a pen), and then reproducing the same body gesture (using the same hand to grasp the pen).

(Alissandrakis et al. 2003) [1] illustrated nicely the problem of determining "what to imitate" in a chessworld case-study, in which the imitator agent can follow either of three strategies, "end-point level", "trajectory level" and "path level", to reproduce either subparts or the complete path followed by the demonstrator. We follow a similar taxonomy and apply it to the learning and reproduction of a manipulation task by a humanoid robot. We take the perspective that the features of the movements to imitate are those that appear the most frequently, i.e. the invariants in time. The rational behind the model is the following: Imagine that you must determine the rules guiding a game, solely based on the observation of two players' moves. If the game is Chess, then only the positions of the chess pieces on the board, and not the particular gestures of the players, are relevant to the reproduction (e.g. it does not matter if the players drive the pieces with right or left hand). If, on the other hand, the game is tennis, then both the motion of the ball and the gestures of the players are relevant. Over time, while comparing different players, only the relevant features of each task will remain invariant.

Demonstration of a box manipulation

The model builds upon previous work [4,5,6]. It combines different pattern recognition techniques, namely hierarchical time delay neural networks, hidden markov models, k-mean clustering, to extract invariant features from a manipulation task performed by a human demonstrator. The system analyses:

1. the trajectories of the objects in the 3-D Cartesian space, given by a fixed two-camera tracking system,
2. the trajectories of the joints recorded by an exoskeleton worn by the demonstrator.



Reproduction by the humanoid robot DB (ATR).

By comparing the probabilities of occurrence of the different events, the model determines whether the goal of the manipulation task is:

a. To move a specific object
b. To move the objects in a specific direction
c. To move the objects in a specific sequence
d. To perform a specific gesture

The observation of the manipulation task is then used to drive the reproduction of the task by a full body humanoid robot.

References:

1. A. Alissandrakis, C. L. Nehaniv, K. Dautenhahn, (2002), "Imitating with ALICE: Learning to Imitate Corresponding Actions across Dissimilar Embodiments", IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans, *IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans*, Vol. 32, Issue 4, pp. 482-496.
2. Chrystopher L. Nehaniv and Kerstin Dautenhahn, (2002) „The Correspondence Problem". Imitation in Animals and Artifacts. MIT Press, pp.41-62.
3. Bekkering, H., Wohlschläger, A., & Gattis, M. , (2000), Imitation is goal-directed. *Quarterly Journal of Experimental Psychology*, 53A, 153-64.
4. Billard, A. (2002) Imitation. In M. A. Arbib (ed.), Handbook of Brain Theory and Neural Networks, MIT Press, 566-569.
5. Billard, A, Epars, Y., Cheng, G. and Schaal, S. (2003) Discovering Imitation Strategies through Categorization of Multi-Dimensional Data. *Proceedings of the International Conference on Intelligent and Robotics Systems* ,IROS 2003. Las Vegas. October 2003.
6. Billard, A. and Schaal, S. (2001) Robust learning of arm trajectories through human demonstration. In *Proceedings of the International Conference on Intelligent and Robotics Systems*, IROS 2001, Hawaii, November.
7. Schaal, S., Ijspeert, A.J. and Billard, A. (2003) Computational Approaches to Motor Learning by Imitation. Philosophical Transactions: Biological Sciences (The Royal Society), 358:1431, p.537-547.

Leveraging on a Virtual Environment for Robot Programming by Demonstration

Jacopo Aleotti, Stefano Caselli, Monica Reggiani
RIMLab - Robotics and Intelligent Machines Laboratory
Dipartimento di Ingegneria dell'Informazione,
University of Parma, Italy
Email: {aleotti, caselli, reggiani}@ce.unipr.it

***Abstract*— The Programming by Demonstration paradigm promises to reduce the complexity incurred in programming robot tasks. Its aim is to let robot systems learn new behaviors from a human operator demonstration. In this paper, we argue that while providing demonstrations in the real environment enables teaching of general tasks, for tasks whose essential features are known a priori demonstrating in a virtual environment may improve efficiency and reduce trainer's fatigue. We next describe a prototype system supporting Programming by Demonstration in a virtual environment and we report results obtained exploiting simple virtual tactile fixtures in pick-and-place tasks.**

I. INTRODUCTION

Programming by Demonstration (PbD) aims at solving the persistent problem of programming robot applications [8]–[10], [13]. Robot programming is known to be a complex endeavor even for robotic experts. Simplifying robot programming has become of prominent importance in the current context of service robotics, where end users with little or no specific expertise might be required to program robot tasks.

The PbD tenet is to make robots acquire their behaviors by providing to the system a demonstration of how to solve a certain task, along with some initial knowledge. A PbD interface then automatically interprets what is to be done from the observed task, thus eliminating the need for alternative, explicit programming techniques. Providing a demonstration of a task to be reproduced by others is an effective means of communication and knowledge transfer between people. However, while PbD for computer programming has achieved some success [4], teaching tasks involving motion of physical systems, possibly in dynamic environments, directly addresses the well-known difficulties of embodied and situated systems [3]. Hence, further research is required to fulfill the goals of PbD in robotics.

The most straightforward way to put into practice the PbD concept is by letting the user demonstrate the task in the real world, while taxing the system with the requirement to understand and replicate it. Recent examples of PbD systems involving demonstration in the real world are [13] and [17]. This is also the most general approach to programming by demonstration, but the complexity of the underlying recognition and interpretation techniques strongly constrains its applicability. To circumvent this problem, a PbD system might require to restrict the objects and actions involved in the task to be demonstrated to a predefined set, or set up a highly engineered demonstration environment. However, if objects and actions occurring in the task are constrained in number and type, the same *a priori* knowledge can be transferred into a virtual environment.

Based on this observation, we have begun an investigation into a virtual environment for PbD, focusing on the approach to PbD based on task-level program acquisition [5], [9], [10], [13]. Previous works evaluating PbD in virtual environments include [11], [12], [16]. Performing the demonstration in a virtual environment provides some functional advantages which can decrease the time and fatigue required for demonstration and improve overall safety by preventing execution of incorrectly learned tasks:

- tracking user's actions within a simulated environment is simpler than in a real environment and there is no need for object recognition, since the state of the manipulated objects is known in advance;
- human hand and grasped object positions do not have to be estimated using error-prone sensors like cameras;
- multiple virtual cameras and view point control are available to the user during the demonstration;
- the virtual environment can be augmented with operator aids such as graphical or other synthetic fixtures [15], and force feedback;
- a virtual environment enables task simulation prior to execution for task validation.

In service robotics applications the simulation feature is even more important since in has been pointed out that, in many cases, it would be almost impossible to ask for multiple demonstrations by the user [6]. Of course, these functional advantages should be weighed against the very drawback of a virtual environment, namely its need for advance explicit encoding of a priori knowledge about the task, which restricts the applicability of the approach.

The remaining of this paper presents a prototype PbD

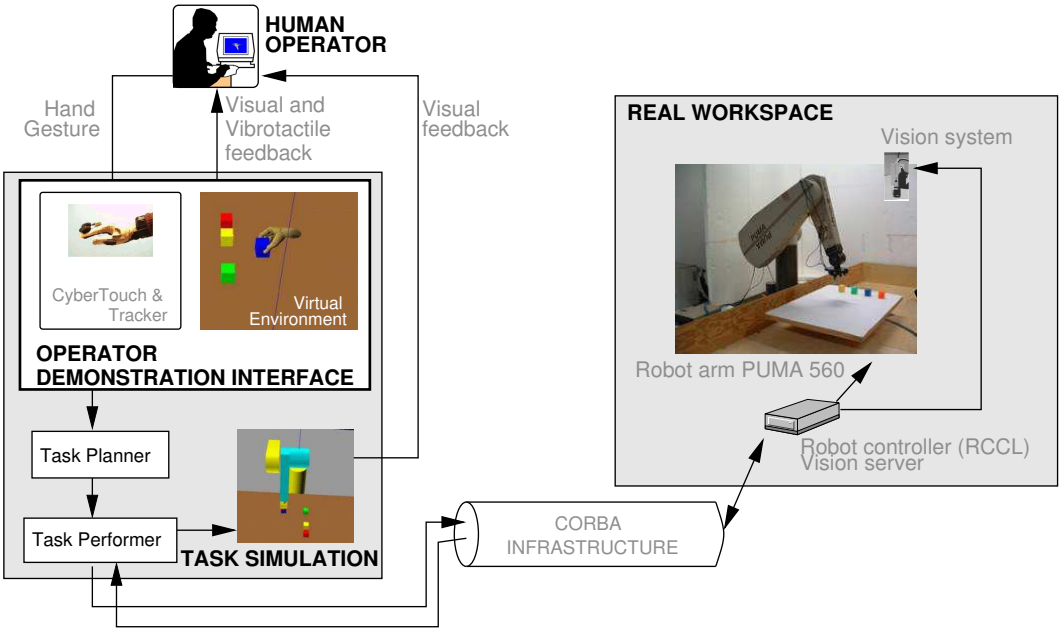


Fig. 1. PbD system architecture.

system that we have set up for simple pick-and-place tasks, and our initial investigation into the exploitation of virtual fixtures to simplify task demonstration.

II. SYSTEM OVERVIEW

The PbD system described hereafter handles basic manipulation operations in a 3D "block world". As mentioned, the system targets task-level program acquisition. We assume that trajectories will eventually be computed by path planning based on the actual location of objects and status of the working environment.

In the proposed robot teaching method, an operator, wearing a dataglove with a 3D tracker, demonstrates the tasks in a virtual environment. The virtual scene simulates the actual workspace and displays the relevant assembly components. The system recognizes, from the user's hand movements, a sequence of high level actions and translates them into a sequence of commands for a robot manipulator. The recognized task is then performed in a simulated environment for validation. Finally, if the operator agrees with the simulation, the task is executed in the real environment referring to actual object locations in the workspace. A library of some simple assembly operations has been developed. It allows to pick and place objects on a working plane, to stack objects, and to perform peg-in-hole tasks.

The architecture of the PbD system (Figure 1) follows the canonical structure of the "teaching by showing" method, which consists of three major phases. The first phase is task presentation, where the user wearing the dataglove executes the intended task in a virtual environment. In the second phase the system analyzes the task and extracts a sequence of high-level operations, taken from a set of rules defined in advance. In the final stage the synthesized task is mapped into basic operations and executed, first in a 3D simulated environment and then by the robotic platform.

Figure 1 shows the main components of the PbD testbed. The actual robot controlled by the PbD application is a six d.o.f. Puma 560 manipulator. A vision system (currently operating in 2D) is exploited to recognize the objects in the real workspace and detect their initial configurations. The whole application is built on top of a CORBA-based framework which interconnects clients and servers while providing transparent access to the various heterogeneous subsystems [2].

A. Demonstration interface

The demonstration interface includes an 18-sensor CyberTouch (a virtual reality glove integrating tactile feedback devices, from Immersion Corporation, Inc.) and a six d.o.f. Polhemus tracker. The human operator uses the glove as an input device. For demonstration purposes, operator's gestures are directly mapped to an anthropomorphic 3D model of the hand in the simulated workspace.

In the developed demonstration setup, the virtual environment is built upon the *Virtual Hand Toolkit (VHT)* provided by Immersion Corporation. To deal with geometrical information in a formal way, VHT uses scene graphs data structure (Haptic Scene Graph - HSG) containing high-level descriptions of environment geometries. VRML models can be easily imported in VHT through a parser

included in the library. To grant a dynamic interaction between the virtual hand and the objects in the scene, VHT allows objects to be grasped. A collision detection algorithm (V-Clip) generates collision information between the hand and the objects, including the surface normal at the collision point. A grasp state is achieved if the contact normals provide sufficient friction; otherwise, if the grasp condition for a grasped object is no longer satisfied, the object is released. The user interface also provides a vibratory feedback using CyberTouch actuators. Vibrations convey proximity information that helps the operator to grasp the virtual objects.

The current implementation of the virtual environment for assembly tasks in the "block world" consists of a plane, a set of 3D colored blocks on it, and possibly one or more containers (holes) whose shape and location are known in advance. This scene includes the same objects of the real workspace configuration, although, in general, the actual locations of blocks will be different.

B. Task recognition

The task planner analyzes the demonstration provided by the human operator and segments it into a sequence of high-level primitives that should describe the user actions. To segment the human action in high-level operations, a simple algorithm based on changes in the grasping state has been implemented: a new operation is generated whenever a grasped object is released. The effect of the operation is determined by evaluating the achieved object configuration in the workspace.

Three high-level tasks have been identified, so far, as basic blocks to describe assembly operations in the simple pick-and-place domain. The first task picks an object and places it onto a support plane (*PickAndPlaceOnTable*), the second task stacks an object onto another (*PickAndPlaceOnObj*), and the third task inserts a small object in the hole of a container lying on the working plane (*PegInHole*). The three high-level tasks have been implemented in C++ as subclasses of a *HighLevelTask* abstract class (Figure 2). Information about the recognized high-level task is passed to the constructor when the *HighLevelTask* class is instantiated.

C. Task generation

A set of *BasicTasks* have been implemented for the basic movements of the real robot. The available concrete classes (Figure 2) include basic straight-line movements of the end effector, such as translations in the *XY* plane, parallel to the workspace table, and along the *z* axis. Two classes describe the basic operations to pick up and to release objects by simply closing and opening the on-off gripper of the manipulator.

The high level tasks identified in the task recognition phase are then decomposed in a sequence of *BasicTasks* objects describing their behavior. In this simple domain, decomposition is straightforward and the three tasks only differ in the height z_f of the release operation. Since the available manipulator has no force sensor, z_f is computed in the virtual demonstration environment based on contact relations. For the peg-in-hole task the grasped object must be released after its initial insertion in the hole.

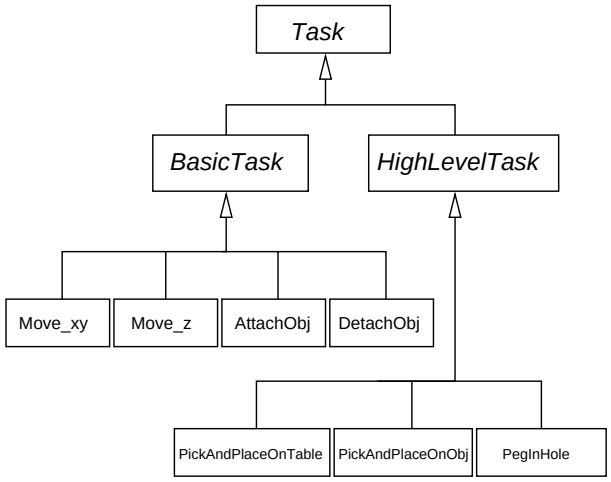


Fig. 2. Task hierarchy.

Each concrete class of the task tree provides two methods to perform the operation, one in the simulated environment and one in the real workspace. Once the entire task has been planned, the task performer (Figure 1) manages execution in both the simulated and real workspaces.

D. Task simulation

After the recognition phase, the system displays to the human operator a graphical simulation of the generated task. This simulation improves safety, since the user can check the correctness of the interpreted task. If the user is not satisfied after the simulation, the task can be discarded without execution in the real environment.

The simulation is non-interactive and takes place in a virtual environment exploiting the same scene graph used for workspace representation in the demonstration phase. The only difference is that the virtual hand node in the HSG is replaced by a VRML model of the Puma560 manipulator. The simulated robot has the ability to perform all the operations described in the previous section. The movement of the VRML model is obtained applying an inverse kinematics algorithm for the specific robot and is updated at every frame. In the simulation, picking and releasing operations are achieved by attaching and detaching the nodes of the HSG representing the objects to the last link of the VRML model of the manipulator.

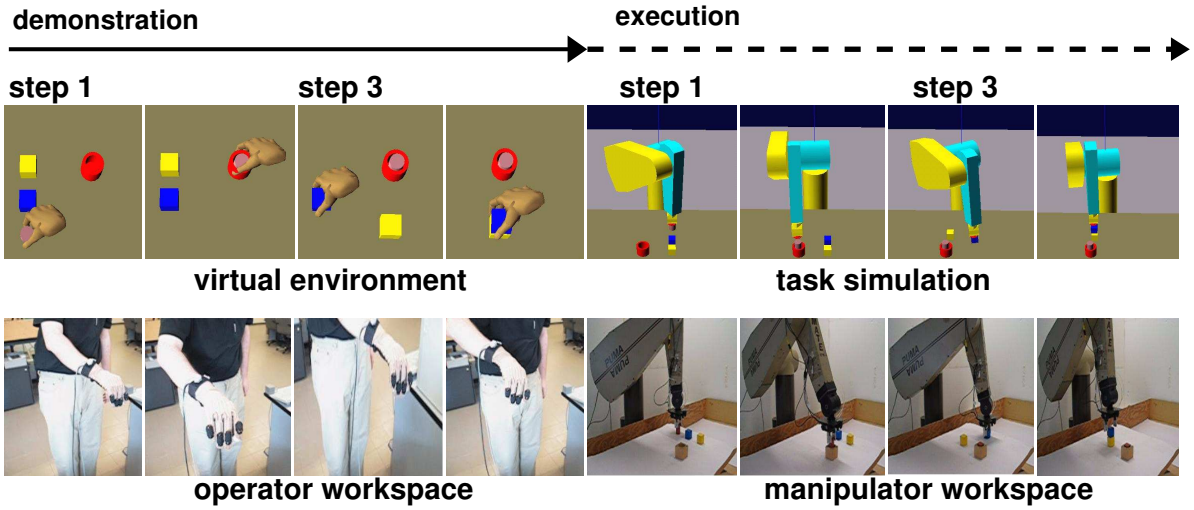


Fig. 3. PbD and execution of a task involving peg-in-hole and stacking operations.

E. Task execution

Execution in the real workspace exploits a C++ framework [2] based on CORBA. The PbD system builds a client-server CORBA connection using a Fast Ethernet switch. The client side runs on MS Windows 2000, whereas the server controlling the manipulator runs on Solaris 8 and the vision server on Linux. The methods of the concrete classes in the task list invoke blocking remote calls of the servant manipulator object which transforms them into manipulator commands based on RCCL – the Robot Control C Library.

F. Sample experiment

Figure 3 shows the demonstration, simulation and execution steps of a pick-and-place task (only initial and final frames are shown). In this experiment the workspace contains four objects: two colored boxes, a cylinder and a container (a cylinder with a hole). The user demonstration consists of a sequence of three steps. The user first picks up the cylinder and puts it in the container, then puts the yellow box on a different position on the table, finally grasps the blue box and stacks it on top of the yellow one. While performing the demonstration, the user can dynamically adjust the point of view of the virtual scene. This feature, typical of demonstration in virtual environments, can help in picking partially occluded objects, releasing them on the plane or on other boxes, and inserting them in the container. Movies of this and other PbD experiments are available at the web page: `http://rimlab.ce.unipr.it/Projects/PbD/pbd.html`.

G. Discussion

Since a large amount of information is readily available in the virtual environment (object locations, hand pose) and since we target pick-and-place tasks, in the proposed PbD system tasks are learned at an abstract level and a single demonstration usually suffices. Hence, the demonstration phase is less demanding than with alternative approaches, even though performing a single demonstration would be simpler for the user in the real environment than in the virtual one.

Task simulation has proven an effective tool to prevent some erroneous executions in the real world. The operator can check whether the learned task is correct and whether it can be executed by the target robot taking into account also its reachability and kinematic constraints.

Task execution by the real robot requires the availability of a sensory system to locate objects in the real workspace and of adequate path planning and robot control capabilities. Once the task has been correctly learned, successful task execution depends on the quality of the robot controller and the accuracy of the vision system. So far, we have not stressed these aspects in our PbD system, although we have successfully executed peg-in-hole tasks with a clearance of 3 *mm*.

III. EXPLOITING VIRTUAL FIXTURES

One of the potential advantages of a virtual demonstration environment, as mentioned earlier, is the ability to incorporate in a simpler way virtual fixtures, i.e. artificial clues that help the operator in performing the task. Virtual fixtures have been introduced as a general concept in robot teleoperation [14], [15]. We argue here that they can play an important role in simplifying task demonstration in PbD as well. While the PbD system described in the previous section is admittedly simple, it allows some analysis of the impact of synthetic fixturing as described in the following.

Our PbD system incorporates virtual fixturing in two ways. First, demonstration in the virtual environment

is somehow easier than it would be with an accurate representation of real world constraints, since we accept some error in the positioning of the grasped object. For example, with default parameter setting, attempting to deposit an object 1 cm below the plane results in a valid *PickAndPlaceOnTable* operation. Likewise, in Figure 3 clearance of the peg-in-hole task in the virtual environment is about three times the actual clearance in the physical world. Thresholds defining an *acceptability zone* are defined for any action, the tradeoff being between the degree of assistance provided to the operator and the ability to discriminate between the contact relations to be established and to achieve the required accuracy in positioning. More cluttered environments would require, thus, stricter thresholds.

Whenever the object is released within the acceptability zone, its location is corrected and re-aligned in the virtual environment. As described, this feature is appropriate only for simple domains like the block world above, yet the underlying concept extends to more general applications. I.e., if sufficient *a priori* knowledge is available, a virtual environment can *guide* the user performing the demonstration toward semantically significant actions, rather than simply *record* user actions. Clearly, if the application demands accurate, free positioning, a different task hierarchy must be defined, along with proper acceptability thresholds.

A second type of virtual fixture is implemented in the PbD system by exploiting the vibrotactile feedback available in the CyberTouch glove. The underlying idea is that exploiting multimodality reduces the perceptual overload of the operator's visual channel [1]. In the current implementation, vibration is activated whenever the object lies within the acceptability zone previously discussed for a release operation. The operator can take advantage of this explicit information by immediately releasing the object, or decline it by moving the object to another location.

Possible variations in this scheme include:

- activating vibrotactile feedback for a short amount of time, so that the user can decide whether take advantage of predefined object alignment (by immediately releasing the object), or override it in favor of a fine manual positioning (by holding the object until the end of vibration);
- providing vibrotactile feedback (which in principle could also be modulated in amplitude) in a wider volume than the acceptability zone, so as to provide a hint guiding user motion.

A. Evaluation

We have asked five subjects, 2 females and 3 males, to demonstrate three elementary tasks and one composite task in the virtual environment. Prior to the actual data collection experiment, subjects were asked to play for 5

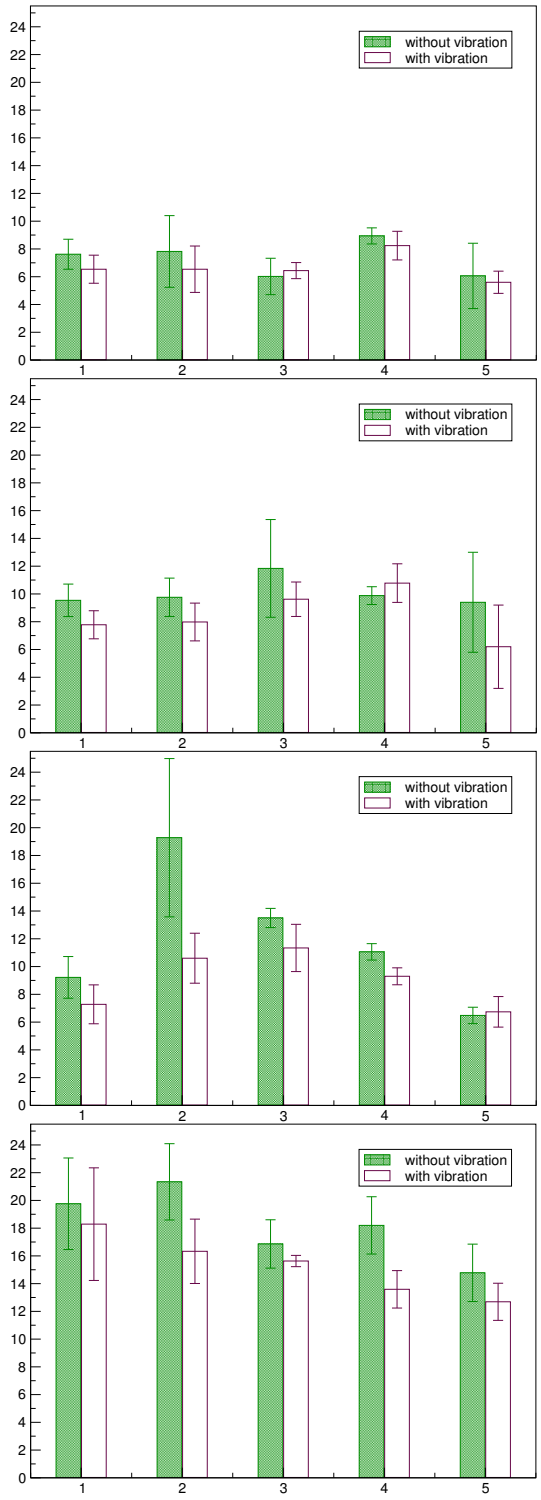


Fig. 4. Average and standard deviation in task completion times for five subjects performing tasks with and without vibration assistance. Vertical axis: time in seconds. Horizontal axis: subject index. Charts refer to different tasks. From top to bottom: object displacement task; object stacking task; peg-in-hole task; composite task.

minutes with the virtual environment, picking and releasing objects.

The elementary tasks to be demonstrated were displacement of a cubic object on table, stacking of a cubic object on top of another one, insertion of a cylindrical peg into a cylindrical hole. Each task also included approach motion, object grasping, and object transportation phases. The composite task was a routine comprising the three elementary tasks in sequence, although with a different object arrangement. For each task, time to completion (time required to perform the demonstration) was measured and the average value and standard deviation computed. Finally, each subject performed the experiment five times using only the graphical output of the virtual environment, and then five times with the virtual tactile fixture on. Task completion time was measured by an external supervisor and triggered when the system reported a successful recognition of the last required *HighLevelTask*.

Figure 4 shows the average and standard deviation in task completion times without and with the virtual tactile fixture in the four experiments. Task completion times in both modality are clearly influenced by the different difficulty of the various tasks. According to Fitt's law [7], a Difficulty Index can be defined for each task, and correlation with average task completion time established (we are currently performing such analysis). As a general remark, the additional tactile fixture helps in decreasing average demonstration times, even though for each elementary task one subject performed slightly worse with the tactile fixture on. For the composite task, the tactile fixture improved execution performance for all subjects. It should be mentioned that the virtual environment is somehow slower with the tactile feedback activated. A higher latency is perceived by the user, which therefore might tend to perform the demonstration more cautiously. The resulting delay might play a role in the outlier data. Moreover, due to differences in object arrangement and initial operator pose, completion times for the composite task cannot be compared with completion times of the elementary tasks.

Figure 5 compares results across the four tasks by scaling each subject performance with the value obtained without the tactile fixture. Dashed lines refer to the average task completion time across all subjects. Qualitatively, virtual tactile fixturing appears to play a more important role for more complex tasks. For the composite task the average degree of improvement is smaller, although all subjects improve their completion times with the tactile virtual fixturing. This is due to the fact that the task includes multiple transfer phases where virtual tactile fixturing plays no role. Our ongoing work attempts to assess the correlation between task difficulty and completion time in a more quantitative manner.

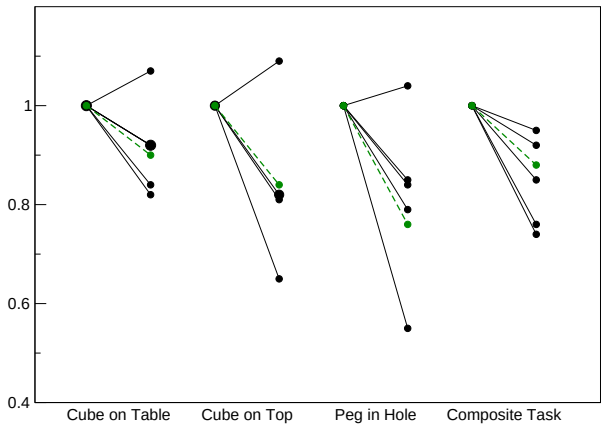


Fig. 5. Assessing the improvement in task completion time using vibration for the four tasks. Vertical axis: ratio between average completion times with and without vibration. Each dot represents a subject, whereas the dashed line connects to the average improvement across all subjects.

IV. CLOSING

We have described an ongoing investigation into exploiting a virtual environment to assist the user in PbD of robot tasks. We have developed a prototype PbD system that uses a data glove and a virtual reality teaching interface to program pick-and-place tasks in a block world.

In this context, we are investigating the potentials of virtual fixtures, both visual and tactile, and their effect on task recognition performance. The ability to easily integrate such virtual fixtures is one of the major advantages of a virtual demonstration environment.

V. ACKNOWLEDGMENTS

This research is partially supported by MIUR (Italian Ministry of Education, University and Research) under project RoboCare (A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components).

VI. REFERENCES

[1] J. Aleotti, S. Caselli, and M. Reggiani. Multimodal User Interface for Remote Object Exploration with Sparse Sensory Data. In *11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002.

[2] S. Bottazzi, S. Caselli, M. Reggiani, and M. Amoretti. A Software Framework based on Real-Time CORBA for Telerobotic Systems. In *IEEE Int. Conf. on Intelligent Robots and Systems*, 2002.

[3] R. A. Brooks. Intelligence without Reason. In *12th Int'l Joint Conf. on Artificial Intelligence*, 1991.

[4] A. Cypher, editor. *Watch What I do: Programming by Demonstration*. The MIT Press, 1993.

[5] 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 Int'l Symp. of Robotics Research*, 1999.

[6] M. Ehrenmann, R. Zöllner, O. Rogalla, and R. Dillmann. Programming Service Tasks in Household Environments by Human Demonstration. In *11th IEEE International Workshop on Robot and Human Interactive Communication*, 2002.

[7] P.M. Fitts. The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement. *Journal of Experimental Psychology*, (47), 1947.

[8] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, and M. Sassin. Robot Programming by Demonstration: Supporting the Induction by Human Interaction. *Machine Learning*, pages 163–189, May 1996.

[9] K. Ikeuchi and T. Suehiro. Towards an Assembly Plan from Observation, Part I: Task Recognition with Polyhedral Objects. *IEEE Trans. on Robotics and Automation*, 10(3), 1994.

[10] 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*, 10(6), 1994.

[11] E. Lloyd, J. S. Beis, D. K. Pai, and D. G. Lowe. Programming Contact Tasks Using a Reality-Based Virtual Environment Integrated with Vision. *IEEE Trans. on Robotics and Automation*, 15(3), 1999.

[12] H. Ogata and T. Takahashi. Robotic Assembly Operation Teaching in a Virtual Environment. *IEEE Trans. on Robotics and Automation*, 10(3), 1994.

[13] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi. Extraction of Essential Interactions through Multiple Observations of Human Demonstrations. *IEEE Trans. on Industrial Electronics*, 50(4), 2003.

[14] L. Rosenberg. Virtual Fixtures: Perceptual Tools for Telerobotic Manipulation. In *IEEE Virtual Reality Annual International Symposium*, 1993.

[15] C. P. Sayers and R. P. Paul. An Operator Interface for Teleprogramming Employing Synthetic Fixtures. *Presence*, 3(4), 1994.

[16] T. Takahashi and T. Sakai. Teaching Robot's Movement in Virtual Reality. In *IEEE/RSJ Int. Workshop on Intelligent robots and systems*, 1991.

[17] R. Zöllner, O. Rogalla, R. Dillmann, and M. Zöllner. Understanding Users Intention: Programming Fine Manipulation Tasks by Demonstration. In *IEEE/RSJ Int'l Conference on Intelligent Robots and Systems*, 2002.

A Posture Sequence Learning System for an Anthropomorphic Robotic Hand

Ignazio Infantino[1] Antonio Chella[1,2] Haris Džindo[1] Irene Macaluso[1]

[1]ICAR-CNR sez. di Palermo
Viale delle Scienze, edif. 11,
90128, Palermo, Italy

[2]DINFO Università di Palermo
Viale delle Scienze,
90128, Palermo, Italy

***Abstract* — The paper deals with a cognitive architecture for posture learning of an anthropomorphic robotic hand. Our approach is aimed to allow the robotic system to perform complex perceptual operations, to interact with an human user and to integrate the perceptions by a cognitive representation of the scene and the observed actions. The anthropomorphic robotic hand imitates the gestures acquired by the vision system in order to learn meaningful movements, to build its knowledge by different conceptual spaces and to perform complex interaction with the human operator.**

I. INTRODUCTION

The control of robotic systems has reached a high level of precision and accuracy, but often the high complexity and task specificity are limiting factors for large scale uses. Today, robots are requested to be both "intelligent" and "easy to use", allowing a natural and useful interaction with human operators and users. A promising approach towards simple robot programming is the "learning by imitation" paradigm (see [19], [21], [26] for reviews on different aspects on imitation). Many working systems have been proposed in the literature [2], [3], [4], [12], [14], [15], [16], [22], [23], [28]. However, these systems, although effective, are generally based on movements recordings obtained by gloves, by particular equipments or by simplified vision; moreover, the imitation capabilities are sometimes limited to simple mimicking of the teacher movements.

We claim that, in order to have a system able to learn by imitation, the system itself may have the capabilities of deeply understand the perceived actions to be imitated. Therefore, the system may be able to build an inner conceptual representation of the learned actions. In this paper, we present an architecture based on learning by imitation that performs visual interaction between an human user showing his moving hand and an anthropomorphic robotic hand (a DIST-Hand built by GraalTech, Genova, Italy). The core of the architecture is a rich inner conceptual level [10] where the representation of perceptual data takes place starting from a real time unconstrained vision system [5], [6], [7], [8], [13]. Our long term project goal is to build a system that may help and collaborate with elderly and impaired persons in everyday life (e.g. a system that helps to pick up an object or to perform some movements).

The current system is equipped with a stereo video camera that acquires the movements of the hand of the user, in order to perform a direct visual control of the robot (by movements imitation) or to interact using a given sign formalism. The acquired visual data are anchored to symbolic descriptions of the human hand postures and operations [7]. The system takes as input a sequence of images corresponding to subsequent phases of the evolution of the scene (the movements of the human hand and their effects on the whole scene), and it generates an output as a suitable action performed by robotic hand, along with the description of the scene. Such a symbolic description may be employed to perform high-level inferences, e.g. those needed to generate complex long-range plans of interaction, or to perform reasoning about the user operations. In order to test our system and to have quantitative data on human system interaction, we consider a measurable experimental setup in which the user plays Rock-Paper-Scissors game. The system task in this setup is to understand the strategy of the human player.

The paper is organized as follows. In the next section, the cognitive architecture is summarized and detailed description of the conceptual representations is given. The third section describes a simple application that involves conceptual space representation and reasoning: human user plays rock, paper, scissors game against the system. Short conclusions follow.

II. THE COGNITIVE ARCHITECTURE FOR VISUAL PERCEPTION AND LEARNING

The aim of the architecture is to integrate visual perception with knowledge representation, with particular emphasis on man–machine interaction. Our proposal is based on the hypothesis that a principled integration of the approaches of artificial vision and of symbolic knowledge requires the introduction of an intermediate representation between these two levels [6]. Such a role is played by a conceptual space, according to the approach proposed by Gärdenfors [10].

The implemented architecture is organized in three computational areas. Fig. 1 schematically shows the relations among them. The *subconceptual* area is concerned with the

low-level processing of perceptual data coming from the sensors. We call it subconceptual because here information is not yet organized in terms of conceptual structures and categories. The subconceptual area includes a 3D model of the perceived scenes. Even if such a kind of representation cannot be considered "low-level" from the point of view of artificial vision, it still remains below the level of conceptual categorization. In the linguistic area, representation and processing are based on the formalism of probabilistic reasoning based on Bayesian networks [17]. In the conceptual area, the data coming from the subconceptual area are organized in conceptual categories, which are still independent from any linguistic characterization.

The purpose of the subsequent discussion is to show how a conceptual representation can be viewed as a composition of three different space, that driven step by step from sensorial to symbolic level. An overview of the system is depicted in Fig. 2.

A. The subconceptual area

As previously stated, the task of the implemented architecture is to deeply understand the postures and movements of the human hand. To this aim, we need the exact 3D reconstruction of the hand to individuate the orientation and reciprocal position with other body parts (arms, face, and so on).

Different methods have been proposed to capture human hand motion. Rehg and Kanade [18] introduced the use of a highly articulated 3D hand model for the tracking of a human hand. Heap and Hogg [11] used a deformable 3D hand shape model. The hand is modeled as a surface mesh which is constructed via PCA from training examples. In [9], Cipolla and Mendoca presented a stereo hand tracking system using a 2D model deformable by affine transformations. Wu and Huang [29] proposed a two-step algorithm to estimate the hand pose, first estimating the global pose and subsequently finding the configuration of the joints. In [27], the Eigenspace method is used to classify hand shape and estimate hand position.

Our method, described in details in [13], uses fingertips as features, extracted from gray level images with black background. Each finger (except the thumb) is considered as planar manipulator. The hand postures is defined reconstructing its joint angles, and the Kalman Filter is used to track the fingertips in an image sequence and computes the 3D coordinates of each of them. The coordinates of the four fingertips and the wrist are used to solve the inverse kinematics problem for joint angles, provided a kinematics model of a human hand.

The model is designed to remain simple enough for inverse kinematics to be done in real-time, while still respecting human hand capabilities. We take into account static and dynamic hand constraints [24] which allow us to reduce the number of DOF of the model to 15.

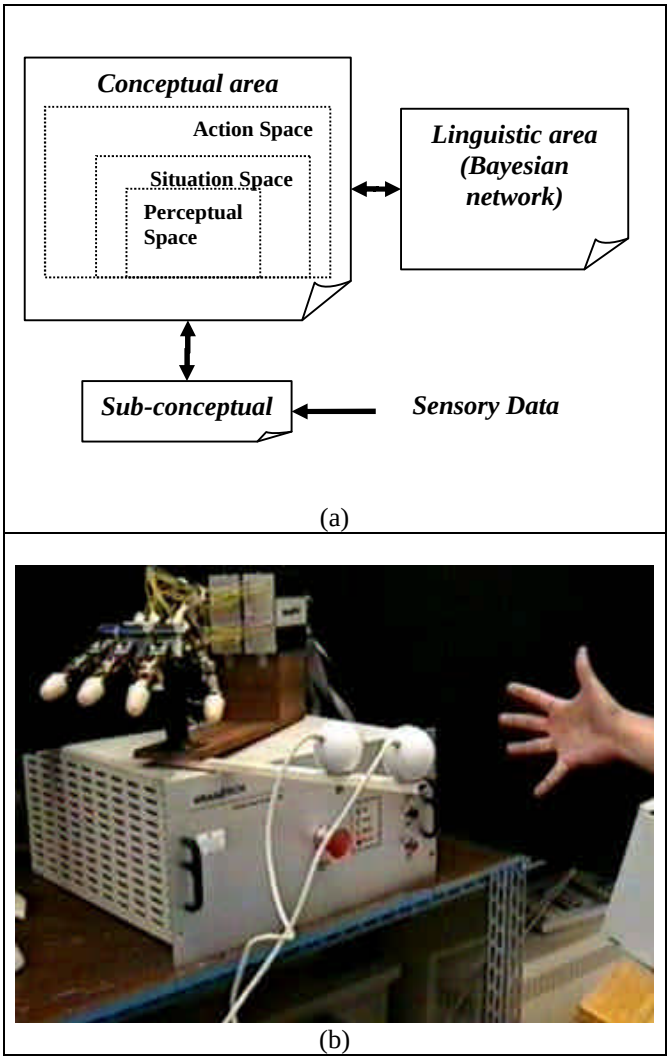


Fig. 1. (a) The three areas of the conceptual representation and the relations among them. (b) The posture reconstruction system and the robotic hand

The robustness of this algorithm has been tested using various fingers configuration: also in the more complicated case of two very close fingers on the background palm the system follow the correct feature. The addition of artificial noise or the use of a lower image scale does not degrade the performance of the algorithm. No constraints of the possible hand postures are necessary and gray-level video images are sufficient to obtain good results. A limitation of the proposed approach is that the background may be uniform in order to obtain real time segmentation; other approaches presented better segmentation algorithms based on color [30] or 3D models [20] but without dealing with real time problems.

We have tested our method by using two different stereo rigs: the first one was composed by two Sony cameras, and the second one by two USB Webcams. The software runs on a personal computer (Pentium III, 450 MHz), equipped with two video grabber cards and an Ethernet card.

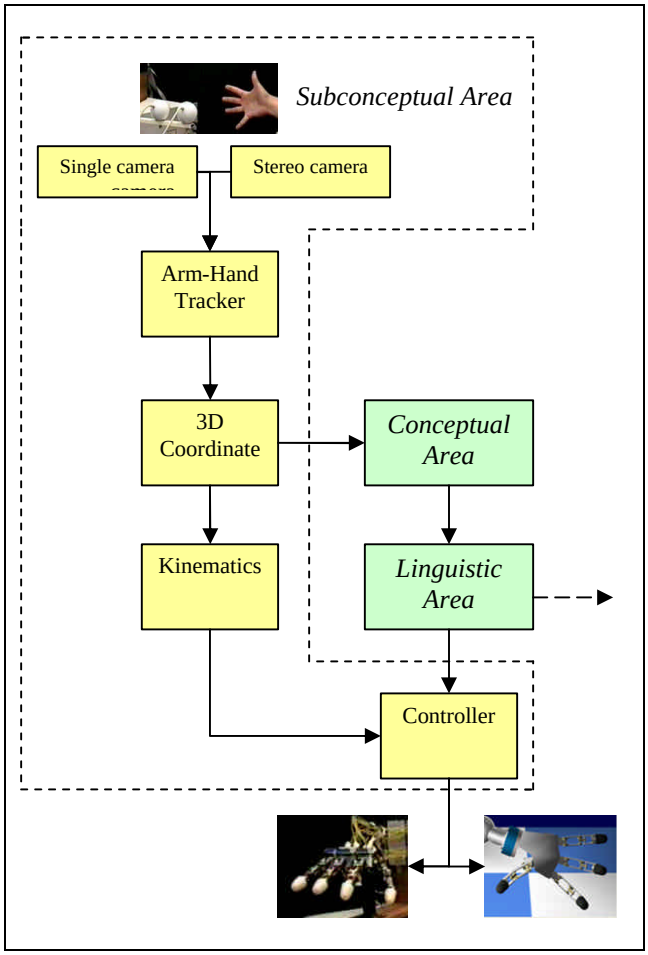


Fig. 2. The system architecture.

The movements command are send to the DIST-Hand using a TCP-IP link to it. The various procedures implemented to perform the posture reconstruction allow a frame rate of 10 images per second, permitting a qualitative correct recognition of the movements of the real hand presented

B. Perceptual Space (PS)

The perceptual space PS is part of the *conceptual area* of the architecture and it is a conceptual space in the sense of Gärdenfors [10], in particular it is a metric space whose dimensions are strictly related with the quantities processed in the subconceptual area. By analogy with the term pixel, we call *knoxel* a point in a PS. A knoxel is an epistemologically primitive element at the considered level of analysis. The basic blocks of our representations in PS are geometric primitives (the joint angle values, or superquadric parameters) describing the acquired scene. In order to account for the dynamic aspects of actions, we adopt a

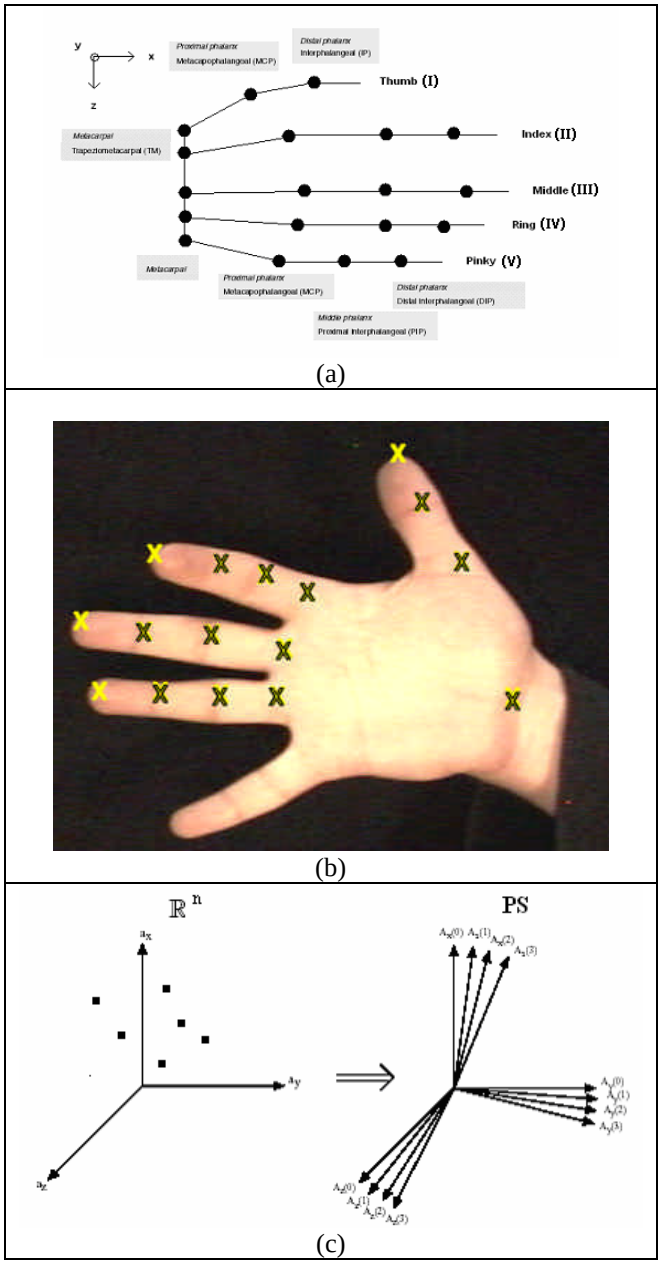


Fig.3. (a) Human hand model. (b)The basic blocks of PS representations are 3D geometric primitives such as phalanx joint angles. (c) Each point of the PS represents a whole simple motion.

perceptual space PS in which each point represents a whole simple motion. In this sense, the space is intrinsically dynamic since the generic motion of an object is represented in its wholeness, rather than as a sequence of single, static frames. The decision of which kind of motion can be considered simple is not straightforward, and it is strictly related to the problem of motion segmentation. In the line of the approach described in [8], we consider a simple motion as a motion interval between two subsequent generic discontinuities in the motion parameters.

In the static PS mentioned above, a moving component had to be represented as a set of points corresponding to subsequent instants of time. This solution does not capture the motion in its wholeness. The implemented alternative has been previously investigated in [8]. We adopt as the conceptual space for the representation of dynamic scenes a dynamic space which can be seen as an "explosion" of the static space. In this space, each axis is split in a number of new axes, each one corresponding to a harmonic component.

Fig. 3 is an evocative, pictorial description of this approach. In the leftmost part of the figure, representing the static PS, each axis corresponds to a 3D geometric parameter; in the rightmost part of the figure, representing the dynamic PS, each group of axes corresponds to the harmonics of the corresponding geometric parameter. Also in this case, a knoxel is a point in the conceptual space, and it corresponds to the simple motion of a geometric component.

C. Situation Space (SS)

A simple motion of a component corresponds to a knoxel in PS. Objects may be approximated by one or more geometric primitives. Let us now consider a scene made up by the human hand. Consider the index opening, as in Fig. 4. We call Situation this kind of scene. It may be represented in PS by the set of the knoxels corresponding to the simple motions of its components, as in Fig. 4, where each knoxel corresponds to a phalanx. In this case, each knoxel corresponds to a moving phalanx of the index and its harmonic components are not zero, while the other knoxels correspond to the phalanxes of quiet fingers (the figure depicts only some of them). Each point in the Situation Space (SS) is a collection of points in PS. SS is a pictorial representation of the global perceived situation.

D. Action Space(AS)

In a Situation, the motions of all of the components in the scene occur simultaneously, i.e. they correspond to a single configuration of knoxels in the conceptual space. To consider a composition of several motions arranged according to a temporal sequence, we introduce the notion of Action in the sense of Allen [1]. An Action corresponds to a "scattering" from one Situation to another Situation of knoxels in the conceptual space.

We assume that the situations within an action are separated by instantaneous events. In the transition between two subsequent configurations, a "scattering" of at least one knoxel occurs. This corresponds to a discontinuity in time that is associated to an instantaneous event. Fig. 5 shows a simple Action performed by the human hand. The figure shows a human hand while opening. This Action may be represented in CS (Fig. 5) as a double scattering of the knoxels representing the phalanxes (the figure depicts only one of them). The knoxel representing the palm remains

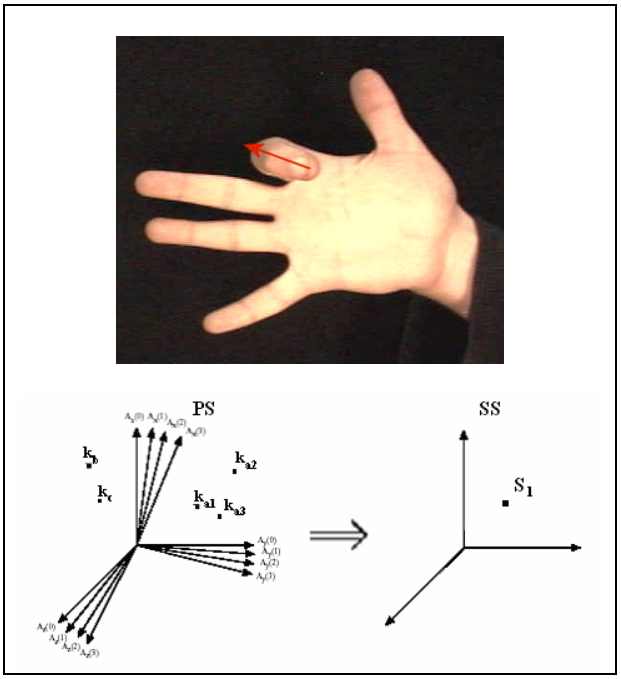


Fig. 4. S_1 is the collection of the points in PS describing the finger movement.

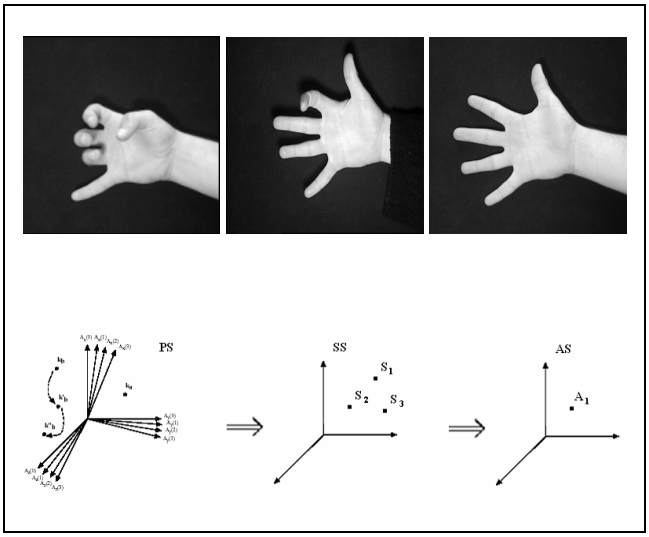


Fig. 5. A_1 is the collection of the points in SS describing the hand action.

unchanged. Each point in the Action Space (AS) is a collection of situations, i .e., of points in SS and it represents a hand action. AS is a pictorial representation of the action performed by human hand.

E. Linguistic area

Long term declarative knowledge is stored at the *linguistic* area. The more "abstract" forms of reasoning, that are less

perceptually constrained, are likely to be performed mainly within this area. The elements of the linguistic area are terms that have the role of summarizing the situations and actions represented in the conceptual spaces previously described, i.e., linguistic terms are anchored to the structures in the conceptual spaces [7].

The symbolic inferences in the linguistic area aimed to plan and decision making, are performed by suitable Bayesian networks. At a given instant, the chosen decision depends from past events and actions executed with a given probability. The interactions between human user and robotic system, initially random, are used to update the tables of probability of the network in order to learn suitable strategies [17].

F. Learning in the architecture

The structures of the conceptual spaces allows to manage the learning of the link between perception and action at different level of representation.

In the Perceptual Space, we need to recognize and classify the motions of single phalanxes (e.g., $UpPhalanx_1$): it is an easy task and the system uses a classifier based on a perceptron neural network.

In the Situation Space, we need to classify and recognize complex dynamic postures: the system uses a recurrent neural network able to learns the different hand configurations. Let us consider a set of knoxels s= {pk_1, pk_2, …, pk_m} corresponding to an instance of a Situation concept C, e.g., Hand Opening. When a knoxel of s, say pk_1, has been individuated by the subconceptual area and it is presented as input to the recurrent network associated to C, the network generates as output another knoxel of s, say pk_2. In this way, the network predicts the presence of pk_2 in SS. The expectation is considered confirmed when the subconceptual area individuates a knoxel pk* so that pk_2~pk* . If the expectation is confirmed, then the network receives as input pk_2 and generates a new expected knoxel pk_3,and so on. The network therefore recognizes the configuration of knoxels of the associated concept according to a recognition and expectation loop.

In the Action Space, we performs a similar mechanism to classify complex actions: when C is an Action, the previously described sequences now refer to a succession of different SS configurations. It should be noted that the SS case is an example of synchronic attention, while the AS case is an example of diachronic attention.

Recurrent neural networks make it possible to avoid an exhaustive linguistic description of conceptual categories: in some sense, prototype Situations and Actions arises from the activity of the neural networks by means of a training phase based on examples. In addition, the measure of similarity between a prototype and a given Situation or Action is implicit in the behavior of the network and is determined by learning. As stated before, in the linguistic area, where long term memory is the instrument to plan and to decide strategies, we use suitable Bayesian networks. The history that determines the behavior of the system is a group of sequential action: the current decision is dependent from past events and actions executed, with a given probability.

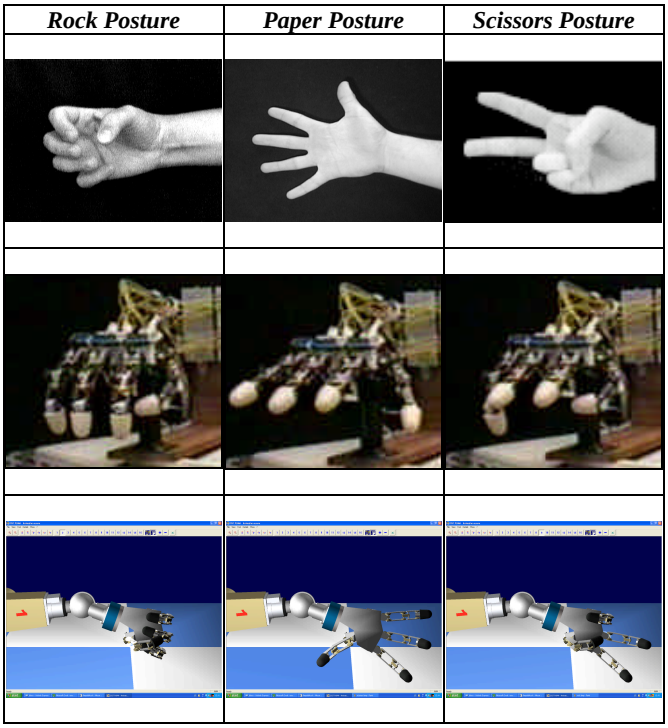


Fig. 6. Elementary postures of the RPS game executed by human user, robotic hand and simulator.

The interactions between human user and robotic system, initially random, are used to update the tables of probability of the network in order to learn strategies of behaviors according to the Bayesian learning algorithms [17].

III. EXPERIMENTS: THE CASE OF ROCK, PAPER, SCISSORS GAME

We adopted an experimental setup that allowed us to measure the degree of learning of the system during human-robot interactions. In this setup, human user plays the Rock, Paper, Scissors game against the robotic hand (see Fig. 6).

We have chosen the RPS (Rock, Paper, Scissors) game because it is simple, fast, involving hand dexterity and strategy between two players. Moreover, RPS game is based on standard hand signs. Also the rules are simple and well-known:

- players contemporarily show one of the three signs;
- rock: wins against scissors, loses to paper and stalemates against itself;
- paper wins against Rock, loses to scissors and stalemates against itself;

- scissors wins against paper, loses to rock and stalemates against itself.

Players may use any combination of these throws at any time throughout the match. Any throws that are not conforming to the standard hand positions and thus deemed to be a rock, paper, or scissors is considered to be an illegal throw and it is thus forbidden.

A. Learning game behavior

The robotic system, in order to choose one of the three game signs, uses a suitable sequential mechanism of expectations. The recognition of a certain component of a Situation (a knoxel in PS) will elicit the expectation of other components of the same Situation in the scene. In this case, the mechanism seeks for the corresponding knoxels in the current PS configuration. The recognition of a certain situation in PS could also elicit the expectation of a scattering in the arrangement of the knoxels in the scene; i.e., the mechanism generates the expectations for another Situation in a subsequent PS configuration. In this way expectations can prefigure the situation resulting as the outcome of an action.

This implements a predictive behavior of the robotic hand, and it represents the ability of a player to predict the opponent action before its completion and using only sensorial input. For example, when the robot recognizes a starting instance of the Paper path situation, it immediately performs the Scissors action.

We take into account two main sources of expectations. On the one side, expectations could be generated on the basis of the structural information learned in the Bayesian network. As soon as a Situation is recognized and the situation is the precondition of an Action, the symbolic description elicit the expectation of the effect situation. Fig. 7 shows an example of the Bayesian network that computes the most probable sign after two throws using a strategy based on the repetition of successful moves. On the other side, expectations could also be generated by purely associative mechanism between situations by means of the previously described neural networks.

Each concept C is associated with a suitable recurrent neural network which acts as a "predictive filter" on the sequences of knoxels corresponding to C.

B. Playing a game

The system has played 500 matches against human user which uses a defined complex strategy based on "gambit" composition. A gambit is a series of three throws used with strategic intent. "Strategic intent" in this case, means that the three throws are selected beforehand as part of a planned sequence. There are only twenty-seven possible gambits, but they can also be combined to form longer, complex combination moves.

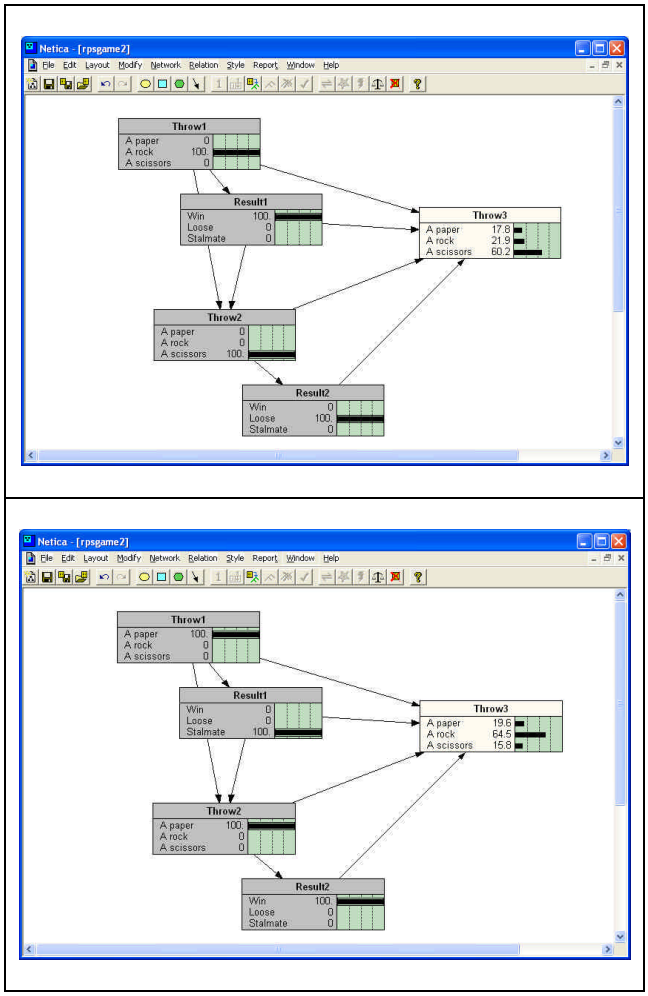


Fig. 7. An example of the Bayesian network that computes the most probable sign after two throws using a strategy based on the repetition of successful moves.

The strategy followed by human player related to the Fig. 8 is represented by the union of the gambit (PSR) and (RPR), with the random choose to repeat the same sign or change gambit after a stalemate or the conclusion of a set.

A single game uses the best of three of three format (max 3 sets, ended when a player wins 2 throws). In the first phase of the challenge (match #1-#50), the system plays at random, obtained a success rate near to 33% (stalemate is counted as fail). The continuous updating of the tables of the Bayesian network introduces the knowledge of opponent's strategy.

After approximately 250 matches the system has completely learned the inner behavior of the human player and has obtained a success rate near to 61,2%. The various experiments done have highlighted a profile of learning process characterized by a random initial phase that lasts 50~75 matches depending from player strategy, a second phase with constant converging learning rate, and a final phase in which the system does not improve its skill.

Match #001			Match #201		
	Human	Robot		Human	Robot
Set 1	P	P	Set 1	**P***	R
	R*	S		S	R*
	S	R*		**R***	S
	R*	S			
			Set 2	R	**P***
Set 2	P	**S***		R	**P***
	R	R			
	R	R	Set 3	R	R
	P*	R		R	**P***
	R	**P***		P*	R
				S	**R***
Set 3	R	R			
	R*	S			
	P	S*			
	R*	S			
Result: Human wins Robot %tw: 30,77%			Robot wins 55,56%		

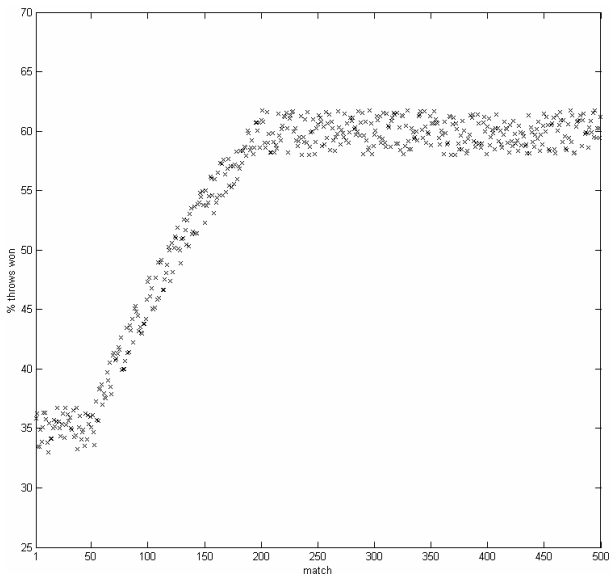


Fig. 8. The results of 2 matches (#1 and #355 of 500) are reported. The graph reports the percentage of throws won by robotic system in a single match. After 250 matches the system has reconstructed the behavior of the human player and has obtained a success rate near to 61,2%.

IV. FUTURE WORKS: EMOTIONAL BEHAVIOR

On the surface, RPS appears to be a game of chance. Whether because of associations with the symbols or the hand positions that represent them, players perceive the three throws to have distinct characteristics. These vary from player to player, but generally fall into some common patterns. For example the World RPS Society web site [25] associates different behaviors to the three possible opening throw:

" rock: use of rock as an opening move is seen by many players to be a sign of aggression; paper: it is actually the most challenging of the basic opening moves since it requires the manual displacement of the most digits. It is therefore generally viewed as the least obvious of opening throws; scissors: opening with a pair of scissors assumes that you are playing against an opponent who has tight control over their aggressive tendencies …".

We are considering a knowledge base that collects emotional tendencies of different human players. If, before starting the game, the system knows its opponent, it could decide the initial sign. On other side, during a game, the history of the throws could be stored and used to associate a typical behavior of current opponent.

IV. CONCLUSIONS

A cognitive architecture for posture learning of an anthropomorphic robotic hand has been presented. Our approach is aimed to allow the robotic system to perform complex perceptual operation, to interact with human user and to integrate the perceptions with a cognitive representation of the scene and the actions. The anthropomorphic robotic hand imitates gestures showed to the vision system in order to learn movements, to build its knowledge by different conceptual spaces and to perform complex interaction with the human operator.

V. ACKNOWLEDGMENTS

This research is partially supported by MIUR (Italian Ministry of Education, University and Research) under project RoboCare (A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components).

VI. REFERENCES

[1] J.F. Allen. Towards a general theory of action and time. Artif. Intell., vol. 23(2), pp. 123–154, 1984.

[2] C. G. Atkeson, S. Schaal, "Learning Tasks From A Single Demonstration", in proc. of IEEE-ICRA 1997, pp. 1706-1712, Albuquerque, New Mexico, 1997.

[3] A. Billard, M.J. Mataric, "Learning human arm movements by imitation: Evalutation of a biologically inspired connectionist architecture, Robotics and Autonomous System, no. 37, pp. 145-160, 2001.

[4] A. Billard, S. Schaal, "Robust learning of arm trajectories through human demonstration", proc. of IROS 2001, pp.734-739, Hawaii, USA, 2001.

[5] A. Chella, M. Frixione, S. Gaglio, “An architecture for autonomous agents exploiting conceptual representations”, Robotics and Autonomous Systems, vol. 25 (3–4), pp. 231–240, 1998.

[6] A. Chella, M. Frixione, S. Gaglio, “A Cognitive Architecture for Artificial Vision”, Artificial Intelligence 89, no. 1-2, pp. 73-111, 1997.

[7] A. Chella, M. Frixione, S.Gaglio, “Anchoring symbols to conceptual spaces: the case of dynamic scenarios”, Robotics and Autonomous Systems, special issue on Perceptual Anchoring, vol. 43, 2-3, pp. 175-188, 2003

[8] A. Chella, M. Frixione, S. Gaglio, Understanding dynamic scenes, Artificial Intelligence, no. 123, pp. 89-132, 2000.

[9] B.D.R. Stenger, P.R.S. Mendonca, R. Cipolla, “Model based 3D tracking of an articulated hand”, *in Proc. CVPR‘01*, pp. 310-315, 2001.

[10] P. Gärdenfors, Conceptual Spaces, MIT Press- Bradford Books, Cambridge, MA, 2000.

[11] A. J. Heap, D. C. Hogg, “Towards 3-D hand tracking using a deformable model”, in 2nd International Face and Gesture Recognition Conference, pp. 140-145, Killington, Vermont, USA, October 1996.

[12] J.A Ijspeert, J. Nakanishi, S. Schaal, “Movement imitation with nonlinear dynamical systems in humanoid robots”, in proc. of Intl. Conf. on Robotics and Automation (ICRA2002), Wahington, 2002.

[13] I. Infantino, A. Chella, H. Džindo, I. Macaluso, “Visual Control of a Robotic Hand”, IROS 2003, Las Vegas, November 2003.

[14] J. Lee, T.L. Kunii, “Model-based analysis of Hand Posture”, in IEEE Comp. Graphics and Appl., pp. 77-86, 1995.

[15] K. Ogawarw, J. Takamatsu, H. Kimura, K. Ikeuchi, “Generation of a task model by integrating multiple observations of human demonstrations”, in proc. of IEEE-ICRA 2002, Washington, DC, Usa, May 2002.

[16] V. Pavlovic, R. Sharma, T.S. Huang., “Visual interpretation of hand gestures for human-computer interaction: a review”, IEEE PAMI, vol. 19(7), pp. 677-695, 1997.

[17] J. Pearl, “Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference”, Morgan Kaufmann, San Francisco, CA, 1988.

[18] J.M. Rehg, T. Kanade, “DigitEyes: Vision-Based Hand Tracking for Human-Computer Interaction”, *in Proc. of the IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, Austin, Texas, pp.16-22, 1994.

[19] J. Rittscher, A. Blake, A. Hoogs, G. Stein, “Mathematical modelling of animate and intentional motion”, Philosophical Transactions: Biological Sciences (The Royal Society), no. 358, pp.475-490, 2003.

[20] R. Rosales, V. Athitsos, L. Sigal, S. Sclaroff, “3D Hand Pose Reconstruction Using Specialized Mappings”, *in Proc. IEEE Int.l. Conf. on Computer Vision (ICCV’01)*, Canada, July 2001.

[21] S. Schaal, A. J. Ijspeert, A. Billard, “Computational Approaches to Motor Learning by Imitation”. Philosophical Transactions: Biological Sciences (The Royal Society), no. 358, pp.537-547, 2003.

[22] T. Starner, J. Weaver, A. Pentland, “Real-time American sign language recognition using desk- and wearable computer-based video”, IEEE PAMI, vol. 20(12), pp. 1371-1375, 1998.

[23] B.D.R. Stenger, P.R.S. Mendonca, R. Cipolla, “Model based 3D tracking of an articulated hand”, in Proc. CVPR‘01, pp. 310-315, 2001.

[24] Y. Wu, J. Y. Lin, T. S. Huang, "Modeling Human Hand Constraints", in Proc. Of Work-shop on Human Motion (Humo2000), Austin, TX, 2000.

[25] Web site of The World RPS Society, http://www.worldrps.com/.

[26] D. M. Wolpert, K. Doya, M. Kawato, “ A unifying computational framework for motor control and social interaction”, Philosophical Transactions: Biological Sciences (The Royal Society), no. 358, pp. 593-602, 2003.

[27] Y. Sato, K. Bernardin, H. Kimura, K. Ikeuchi, “Task analysis based on observing hands and objects by vision”, in Proc. of IROS 2002, Lausanne, Switzerland, Oct. 2002.

[28] A. Ude, T. Shibata, C.G. Atkeson, “Real time visual system for interaction with a humanoid robot”, Robotics and Autonomous System, vol. 37, pp. 115-126, 2001.

[29] Y.Wu, T. S. Huang, “View-independent Recognition of Hand Postures”, in Proc. of IEEE CVPR'2000, Vol. II, pp.88-94, Hilton Head Island, SC, 2000.

[30] X. Zhu, J. Yang, A. Waibel, “Segmenting Hands of Arbitrary Color”, in proc. of Fourth IEEE Intl. Conf. on Automatic Face and Gesture Recognition, Grenoble, France, March 2000.

Motor Representations for Hand Gesture Recognition and Imitation

Manuel Cabido Lopes José Santos-Victor
Instituto de Sistemas e Robótica
Instituto Superior Técnico
Lisbon, Portugal
{macl,jasv}@isr.ist.utl.pt

***Abstract*— We present an approach for grasp recognition and imitation based on models for canonical and mirror neurons, recently found in neurophysiological experiments. *Canonical Neurons* seem to code object affordances, e.g. possible ways of grasping. *Mirror Neurons* code goal directed tasks, like precision or power grasping of an object. The major feature of this neuron population is the use of motor information in the recognition step.**

We propose a Bayesian approach that encompasses all these aspects. Recognition is performed in the motor space and we solve the problem of getting motor information while observing another person. Our approach avoids the complexity of other approaches based on the $3D$ reconstruction of the hand from images, considering that the hand is a multi-articulated object subject to frequent occlusions.

The results obtained illustrate the benefits of designing artificial machines inspired on biological findings and hypotheses, while at the same time, offering robotics technologies as a testbed for such hypotheses.

I. Introduction

Despite being often ignored, an artificial system can retrieve a large amount of knowledge, simply by looking at other individuals, humans or robots working in the same area. In fact, similarly to human infants, a robot could learn significant information if it were able to recognize and imitate what the others are doing.

The long-term goal of our work is two-fold. On one hand, we want to develop methodologies whereby a system can learn how to perform complex tasks through imitation. On the other hand, our approach relies on recent findings in neuroscience and developmental psychology, hoping to contribute to a better understanding of the fundamental problem of how humans imitate each other and how they recognize and understand the observed behavior and actions.

This work is motivated by the recent discovery of *mirror* and *canonical* neurons [1], [2] in the F5 area of the macaque's brain. These neurons discharge during the execution of hand/mouth movements. In this paper we will focus on hand gestures, often referred to as grasp actions or grasps.

In spite of their localization in a pre-motor area of the brain, *mirror* neurons fire not only when the animal performs a specific goal-oriented grasping task, but also when observing that same action being performed by another individual. Canonical neurons [3] have the intriguing characteristic of responding when objects, that afford a *specific* type of grasp, are present in the scene, even if the grasp action is not performed or observed.

By establishing a direct connection between gestures performed by a subject and similar gestures performed by others, mirror neurons may be intimately connected to the ability to imitate found in some animal species [2], establishing an implicit level of communication between individuals.

The discovery of mirror neurons raises the fundamental question of understanding the role of motor information for "visual" gesture recognition, and how can it be facilitated by the fact that we know how to perform those gestures. This is clearly distinct from most approaches for gesture recognition, where only visual information is involved. In our paper, instead, recognition is performed in the motor space and we show that it really simplifies the problem by affording a larger degree of invariance to viewpoint modifications.

Visuo-motor representations can be acquired during extensive periods of self-observation, as well as from observing other individuals. The subject can learn how to perform various gestures and what effect they produce on the visual space and on world objects. Observation can be useful in different ways:

(i) By manipulating objects, one can learn which grasp types are successful for a certain class of objects. Also, if we observe *other* people manipulating objects, we can learn the most likely grasps or functions, for a given class of objects. We will refer to these grasp types or functions as a particular type of *affordances* [4] associated to a certain object. For recognizing gestures, affordances provide prior information as to which gestures are more likely, when acting upon a certain object class. This is a possible interpretation of the role played by *canonical* neurons in the overall process of gesture recognition and imitation.

(ii) When observing one's own gestures, the hand appearance can be estimated and directly related with the corresponding motor commands. We will refer to this

association as the *Visuo-Motor Map* (VMM). Once the VMM has been estimated, one can transform views of observed gestures to motor descriptions that can either be used for recognition or to elicit the corresponding (imitated) gesture.

Grasp actions are usually partitioned into the *transport* and *grasp* phases [5]. During the transport phase, the hand moves towards the target and the grasp phase corresponds to the final segment, immediately before and after touch. It has been shown that the transport phase can change significantly, according to the particular grasp type that is performed in the end of the movement. However, it seems that this information is not used by humans for gesture recognition. Mathematically, this can be interpreted as poor (uncertain) predictive capabilities, as it is only in the final (grasp) part of the gesture that recognition takes place.

Similarly, in our work, recognition will only be based on the grasp phase of the gesture. Figure 1 illustrates the hand appearance during the approach phase, together with the final phase of two broad classes of grasps that will be used in this work: precision grip and power grasp.

Fig. 1. Hand appearance during the approach phase (left), power grasp (center) and precision grip (right).

Gesture recognition has been addressed in the computer vision community in many different ways [6]- [11]. The difficulty of hand tracking and recognition arises from the fact that the hand is a deformable, articulated object, that may display many different appearances depending on its configuration, viewpoint or illumination. In addition, there are frequent occlusions between hand parts (e.g. fingers).

Modeling the hand as an articulated object in the 3D space implies extracting and tracking finger-tips, fingers, and other notable points in the image. This is in general quite difficult, depending on the taken viewpoints and image acquisition conditions. To overcome this difficulty, we exploit more iconic representations for the hand shape, that are commonly believed to be used by humans when recognizing (known) gestures. Also, our approach will make use of motor information, since it is invariant to the viewpoint, as suggested by the existence of mirror neurons.

The recognition of other individuals and imitation are always intertwined and imitation mechanisms can allow better recognition. Several works suggest imitation as a very important paradigm for programming robots [12]. The imitation mechanism can be better understood if some computational models are developed that emulate the brain. An important work [13] modeled several components of the brain, presumably involved in imitation. This work was extended for the case of grasp recognition (mirror neurons) [14] and an implementation with video data was used. Although good results were obtained, the visual features used are very difficult to extract, which makes it difficult to use in real world conditions. For the case of learning motor skills, [15] presents a biologically motivated architecture. This systems works with real data and allows learning of repetitive patterns and precise movements for grasp and reaching. Several other works used biological principles in order to achieve imitation [16] - [19]. Instead of mapping different brain regions and modeling the way they function, our goal in this work is to investigate the mathematical properties of some mechanisms, hypothetically developed through evolution, that allow to recognize and imitate others. Most of the cited works, although recognizing the complexity, simplified the perception either using markers or reducing the possible postures of the demonstrator.

Imitation can be done with simple mechanisms. If the motor system is activated in order to reduce some error function derived from the visual perception, imitation emerges. This *homeostatic* behavior was used in [20], [21] in order to imitate hand trajectories.

A discussion between passive and active imitation is present in [22]. The first case relies on a perceive-recognize-reproduce sequence, while the second uses a map from perception to a set of behaviors. In the case of gesture imitation the traditional way would be to have a visual gesture classifier and then generate a similar problem. Active imitation needs a direct link from perception to action. In the cited work this two modules are mixed allowing imitation of known and unknown actions. In our work, for the case of visual features, some action generation would be necessary after the classification (passive imitation), for the case of motor features the action generation is temporally mixed with recognition (active imitation). We show, in this work, that the use of motor features allows better and more robust classification and imitation. Although for low-level imitation the map allows for imitation of unknown sequences, for grasp actions only known gestures can be imitated.

As a final comment, we would like to remark that, to consider gestures performed by the entire arm, we would

need to include some sort of visual transformation to deal with the problem of viewpoint shape variance [23]. For hand movements, our approach is invariant to large variety of view points. Also, during self-observation, the system can generate a large variety of hand visual stimuli that will be used for the construction of visuo-motor maps. The viewpoint transformation for arm gestures is specifically addressed in [24].

In the next section, we will detail the main structure of our approach. In Section III we describe our Bayesian framework for grasp actions recognition and imitation, that involves models of canonical and mirror neurons. We detail how to learn the prior densities and likelihood function from data and how to estimate a visuo-motor map (VMM), using data acquired during self-observation. As suggested by studies of mirror neurons, recognition takes place in motor variables rather than visual. Finally we present some experimental results in Section IV and discuss the main conclusions in Section V.

II. Approach

Gesture recognition is, in general, a complex task [6]-[11]. Traditional approaches imply performing full $3D$ reconstruction of the hand, followed by a pose classifier. To make the 3D reconstruction, it is necessary to track the fingertips, while handling the multiple occlusions generated by the complex hand motion. State-of-the-art algorithms rely on good initial estimates and require sophisticated kinematic models of the hand.

The approach we propose here differs from other works in several ways: (i) use of object affordances in the recognition process (canonical neurons); (ii) recognition is performed in the motor space (mirror neurons) and (iii) use of global descriptors of the hand appearance.

Many objects are grasped in very precise ways, since they allow the object to be used for some specific purpose. A pen is usually grasped in a way that affords writing and a glass is hold in such a way that we can use it to drink. Hence, if we recognize an object that is being manipulated, it immediately tells us some information about the most likely grasping possibilities (expectations) and hand appearance, simplifying the task of gesture recognition.

This link between objects and their affordances is possibly played in the macacque's brain by the *canonical neurons* of the area F5. If two objects can be grasped in the same way, the same neurons will fire when either object is presented. The affordances of the object have thus an attention property because the number of possible (or likely) events are reduced, thus overcoming possible ambiguities. This will be the first module of our overall system architecture.

We have seen in the previous section that, in spite of their localization in a motor area of the brain, mirror neurons are also active during pure visual (recognition) tasks. When observing someone doing a familiar gesture, the same neurons, that would fire when performing this same gesture, become active. It has also been shown that lesions in the motor part of the brain do affect recognition capabilities.

This observation suggests that the motor system responsible for triggering an action is also involved when recognizing that same action, leading to the question of how to use motor information for recognition. Since during the recognition, only visual information is available, the solution lies in making a transformation from visual to the motor space, where recognition will eventually be done.

The common approach to recognition involves comparing acquired visual features to data from a training set. Instead, we will first use a *Visual-Motor Map* to convert such measurements to the motor space and then perform the comparison/recognition in terms of motor variables.

The advantage of doing this inference in the motor space is two-fold. Firstly, while visual features can be ambiguous, we show that converting these features to the motor space may reduce ambiguity. Secondly, as the motor information is directly exploited during this process, imitation can be done immediately, as all the information/signals are readily available.

To use motor representations for grasp recognition, we need to define *Visuo-Motor maps* (VMMs) to transform visual data onto motor information. The VMM can be learnt during an initial phase of self-observation, while the robot performs different gestures and learns its visual effects.

The question that remains is that of choosing what visual features to use. As we will focus on the classification and imitation of coarse gestures (power grasp and precision grip), we will rely on global appearance-based image methods. Together with the prior information provided by the canonical neurons, appearance based methods offer an easier, fast and more robust representation than point tracking methods.

In the next section we will present a Bayesian approach for a gesture recognition that includes models of the *canonical* and *mirror* neurons, using visual appearance methods. The approach leads to excellent classification rates and classification occurs in the motor space.

III. A Bayesian model for Canonical and Mirror Neurons

Gesture recognition can be modeled in a Bayesian framework, which allows to naturally combine *prior* information and knowledge derived from observations (likelihood). The role played by canonical and mirror neurons will be interpreted within this setting.

Let us assume that we want to recognize (or imitate) a set of gestures, G_i, using a set of *observed* features, F.

For the time being, these features can either be represented in the motor space (as mirror neurons seem to do) or in the visual space (directly extracted from images). Let us also define a set of objects, O_k, present in the scene, that represents the goal of a certain grasp action.

The prior information is modeled as a probability density function, $p(G_i|O_k)$, describing the probability of each gesture given a certain object. The observation model is captured in the *likelihood function*, $p(F|G_i, O_k)$, describing the probability of observing a set of (motor or visual) features, conditioned to an instance of the pair gesture and object. The *posterior* density can be directly obtained through Bayesian inference:

$$p(G_i|F, O_k) = p(F|G_i, O_k)p(G_i|O_k)/p(F|O_k),$$

$$\hat{G}_{MAP} = arg\max_{G_i} p(G_i|F, O_k) \quad (1)$$

where $p(F|O_k)$ is just a scaling factor that will not influence the classification.

The MAP estimate, G_{MAP}, is the gesture that maximizes the posterior density in Equation (1). In order to introduce some temporal filtering, features of several images can be considered:

$$p(G_i|F, O_k) = p(G_i|F_t, F_{t-1}, ..., F_{t-N}, O_k),$$

where F_j are the features corresponding to the image at time instant j. The posterior probability distribution can be estimated using a naive approach, assuming independence between the observations at different time instants. The justification for this assumption is that, recognition does not necessarily require the accurate modeling of the density functions. We then have:

$$p(G_i|F_t, ..., F_{t-N}, O_k) = \prod_{j=0}^{N} \frac{p(F_{t-j}|G_i, O_k)p(G_i|O_k)}{p(F_{t-j}|O_k)}$$

A. The role of canonical neurons

The role of canonical neurons in the overall classification system lies essentially in providing the affordances, modeled as the *prior* density function, $p(G_i|O_k)$ that, together with evidence from the observations, will shape the final decision. This density can be estimated by the relative frequency of gestures in the training set.

Canonical neurons are also somewhat involved in the computation of the likelihood function, since it depends both on the *gesture* and *object*, thus implicitly defining another level of association between these. Computing the likelihood function, $p(F|G_i, O_k)$, is more elaborated and is described in detail in Section III-B.

B. Estimating the likelihood function

As the likelihood function may correspond to a complex distribution, it will be modeled it by a Gaussian mixture, which is fitted to data points. In what follows we will describe the process of fitting a mixture model to a density, $p(x)$:

$$p(x) = \sum_{j=1}^{K} \pi_j \, p(x|j),$$

where $p(x|j) \sim N(\mu_j, \sigma_j)$, is a Gaussian distribution. For a proper probability density function, we need to ensure that $\sum_{i=1}^{K} \pi_i = 1, \ \pi_i \geq 0$.

The Expectation-Maximization (EM) algorithm can be used to estimate the parameters μ_i, σ_i, π_i that best fit the data. The main problem with this solution is the necessity of knowing in advance the number of kernels, K. In [25], [26] there is the option of modifying the number of Gaussian kernels used to best fit the data. The number of kernels can be increased during the learning process, based on a new measure designated as the total *kurtosis*, $\mathcal{K}$:

$$\mathcal{K} \triangleq \int_{-\infty}^{\infty} \left(\frac{x - \mu_j}{\sigma_j} \right)^4 \frac{p(j|x)}{\pi_j} p(x)dx - 3$$

The *kurtosis* measures how far a distribution is from a Gaussian and it is zero for a Gaussian function. If the *kurtosis* is not close to zero for a given kernel, it means that the data are not Gaussian and this kernel is split. On the other hand, the number of kernels can sometimes be reduced (merged) in order to reduce the model complexity. A "closeness" metric between two kernels, can be defined as follows:

$$d(p_1, p_2) = \frac{\prod_{x_i \in X_1} p_2(x_i) \prod_{x_i \in X_2} p_1(x_i)}{\prod_{x_i \in X_1} p_1(x_i) \prod_{x_i \in X_2} p_2(x_i)}$$

where X_i stands for the data points used for the estimation of $p_i(x)$.

Two different kernels can be merged if the distance between them is sufficiently small. At the end of this process, we have an estimate of the likelihood function directly from the data, without imposing a particular structure for the underlying distribution. An important point worth mentioning is that this method can cope with clusters that with very irregular shapes and that it automatically adapts to the shape of such clusters..

C. Mirror Neurons

The classification done by our system as several properties similar to the mirror neurons. In this section we will see how to account to some of the observations regarding this neurons into our Bayesian framework. We must first consider a Visuo-Motor Map that transforms observed visual data, to the motor representations that will eventually drive the recognition process.

1) Visual versus motor features: An image contains a large amount of highly redundant information. This allows for the use of methods whereby the image information is compacted in lower dimensional spaces, thus boosting computational performance. Our visual features consist of projections of the original image onto linear subspaces, using Principal Components Analysis (PCA). As a result, our images can be compressed to a 15 dimension coefficient vector.

Rather than representing the hand as a kinematic model built from tracked fingers and finger tips, we code directly the image as templates projected in the low-dimensional subspace. This method has the advantage of being robust and fast.

In a real (robotic or living) system, motor features would correspond to proprioceptive information about the hand/arm pose/motion. In our experiments [27], this is obtained through the use of a data-glove that records 23 joint angles of someone's hand performing gestures.

2) Visuo-Motor Map: As referred previously, the *Visuo-Motor Map* must transform the features defined in the previous section, from the visual space to the motor space.

$$VMM : \mathbf{F}^V \rightarrow \mathbf{F}^M$$

As the structure of the transformation is quite complex, it was learned with a Multi-Layer Perceptron, for each joint angle. For each network, i, the input consists of a 15-dimensional vector $\mathbf{F}^V$, which are the PCA components of the imaged hand appearance. The output consists of a single unit, coding the corresponding joint angle, $\mathbf{F}_i^M$. There are 5 neurons in the hidden layer.

We assume that $\mathbf{F}^V$ is captured across many different view points. This is possible to generate during self-observation since a huge variety of hand configurations can be easily displayed. Otherwise, some kind of viewpoint transformation is needed to pre-transform the visual data [24].

The VMM can lead to impossible (temporal) trajectories, as errors in input frames can cause discontinuities in the motor space. To overcome this problem, continuity is imposed in the motor data through a first-order dynamic filter.

Each network was trained with momentum and adaptive *back-propagation* with the data pre-processed to have zero mean and unitary variance. It converges to an error of 0.01 in less than 1000 epochs.

Figure 2 shows trajectories (solid-line) for a joint angle of the little finger when performing several precision grips.

It is noticeable that, even inside each grasp class, the variability is very large. This is due to the differences between the grasped objects, and illustrates how the observed features depend not only on the "grasp" type but also on the manipulated object (see Section III-A for discussion). The dashed-line in the figure shows that the trajectory

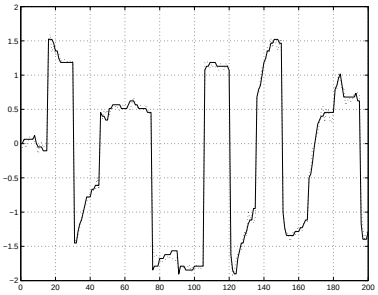


Fig. 2. A sequence of several trials of a precision grip experiment. Solid line: original motor information. Dotted Line: reconstructed motor information using the Visual-Motor Map (VMM)

reconstructed through the neural-VMM is in a very close agreement with the "true" values.

A final aspect worth mentioning is that the VMM can be learned very naturally during an initial phase, when a system (natural or artificial) performs hand/arm gestures and observes the (visual) consequences of such gestures. During self-observation, both proprioceptive (motor) and visual data are present and the association can be established. As an additional aspect, self-observation would allow the system to search and tune the most interesting visuo-motor features such that a more compact representation could be used.

IV. Experimental results

For the results presented here, we use a data set prepared at the Lira Lab, University of Genova, [27], with a specially designed experimental setup. Several subjects were asked to perform different types of grasp on different objects. The experiment begins with the subject sitting in a chair, with the hand on the table. Then, the subject is told to grasp the object that is in front of him.

The experiments include two types of grasp: power grasp and precision grip. Power grasp is defined when all the hand fingers and palm are in contact with the object. Instead, in precision grip, only the fingertips touch the object.

We considered three different objects: a small sphere, a large sphere and a box. The small sphere is sufficiently small so that only precision grip is allowed. The big sphere allows only power grasps. The box is ambiguous because it allows all possible grasps with different orientations.

Every experiment was repeated several times under varying conditions. The subject and the camera go around the table to cover a large variation of viewpoints. To record the sequences we use a stereo-pair. In total, we record the experiments from 6 different azimuths (12 if we consider the stereo-pair). In order to record the motor information, a data-glove [28], capable of recording 23 values of the hand configuration, is used. We used the first 15 values that correspond to all the joint angles (3

for each finger). Finger's abduction and palm and wrist flexion were also available but they were not used in the recognition. Altogether the data-set contains sixty grasp sequences with three objects, two grasps with six different azimuths.

Figure 3 shows sample images of the data set acquired according to process just described. Notice the multiplicity of grasps and view points. Some external observations of an arm are impossible to have when looking to one's arm. For the case of an hand this is not the case because moving the arm allows observing the hand from all viewpoints. Because of this some arm images that might appear impossible have realistic hand observations.

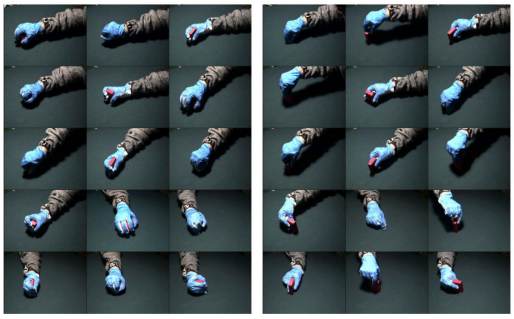


Fig. 3. Data set illustrating some of the used grasp types: power (left) and precision (right). Altogether the tests were conducted using 60 sequences, from which a total of about 900 images were processed.

Every video sequence is automatically processed in order to segment the hand. First, a color-based clustering method, in the Y-Cr-Cb space, was applied to extract skin-colored pixels. The bounding box is determined based on the vertical/horizontal projections of the detected skin region. Finally, the hand is resized for a constant scale before applying the PCA. This approach yields uniformly scaled hand image regions. Figure 4 presents some segmentation results.

Fig. 4. Segmentation results of scale-normalized hand regions automatically detected from colour clustering.

Table I shows the obtained classification rates. It allows us to compare the benefits of using motor representations for recognition as opposed to visual information only. The results shown correspond to the use of the ambiguous objects only, when the recognition is more challenging. We varied the number of viewpoints included in both the training and test sets, so as to assess the degree of view invariance attained by the different methods.

In the first experiment, both the training and test sets correspond to one single view point. Training was based on 16 grasp sequences, while test was done in 8 (different) sequences. The achieved classification rate was 100%. The number of visual features (number of PCA components) was also tuned and the value of 5 provided good results. The number of modes (gaussians in the mixture) were typically from 5 to 7.

The second experiment shows that this classifier is not able to generalizes to other view points / camera positions. We used the same training-set as in $Exp.I$, but the test-set is formed with image sequences acquired with 4 different camera positions. In this case, the classification rate is worse than random (30%).

In the third experiment, we added view point variability in the training set. When sequences from all camera positions are included in the training-set, the classification rate in the test-set drops to 80%. While this is a more acceptable value, it is nevertheless a significant drop from the desired 100%. This result shows that the view point variation introduces such challenging modifications in the hand appearance that classification errors occur.

The final experiment corresponds to the main approach proposed in this paper. The system learns a visuo-motor map during an initial period of self-observation. Then, the VMM is used to transform the (segmented) hand images to motor information, where classification is conducted. A very high degree of classification was achieved (97 %). Interestingly, the number of modes need for the learning is between 1-2 in this case as opposed to 5-7, when recognition takes place in the visual domain. This also shows that mapping visual data to motor representations, helps clustering the data, as it is now view-point invariant.

Notice that view-point invariance is achieved when the training set only contains sequences from one single view point.

TABLE I

GRASP RECOGNITION RESULTS. NOTICE THE GAIN OBTAINED IN THE CLASSIFICATION RATE AND VIEWPOINT INVARIANCE DUE TO THE USE IF MOTOR FEATURES.

	Exp. I (visual)	Exp. II (visual)	Exp. III (visual)	Exp. IV (motor)
	Training			
# Sequences	16	24	64	24
View Points	1	1	4	1
Classif. Rate	100%	100%	97%	98%
# Features	5	5	5	15
# Modes	5-7	5-7	5-7	1-2
	Test			
# Sequences	8	96	32	96
View Points	1	4	4	4
Classif. Rate	100%	30%	80%	97%

These experiments show that motor representations describe the hand better, for gesture recognition, due to the inherent viewpoint independence. As only visual information is available during recognition, the process greatly depends on the *VMM*. The results also validate our approach to estimate the VMM. For the case of only one camera position the quality obtained was very good, with 15 visual features.

The use of motor features for the recognition, has the additional advantage of making imitation a straight forward process, as all the reasoning is performed in motor terms. Figure 5 shows a hand imitating an observed gesture.

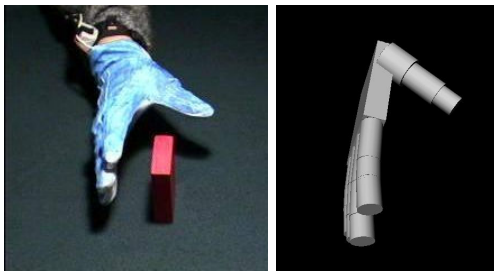


Fig. 5. Reconstruction results of our model hand, obtained with the VMM

V. Conclusions

Neurophysiology can provide many useful ideas for engineers to build more efficient artificial systems. On the other hand, designing artificial systems, grounded on such biological principles, is a valuable means of validating hypotheses or theories in biology.

In this work we propose a framework for gesture recognition based on a model for *canonical and mirror neurons*, that seem to play a fundamental role for grasp recognition or imitation in primates.

Canonical neurons provide prior information in terms of object affordances which narrows the attention span of the system, since very unlikely gestures or hand appearances can be discarded immediately . The fact that, despite being located in a motor area of the brain, mirror neurons are active during both the execution and recognition of an action, suggest that recognition takes place in the motor space rather than on the visual space.

We propose a Bayesian formulation where all these observations are taken into account. We describe how to estimate the prior density and likelihood functions directly from the data. A Visuo-Motor Map is used to transform image data to the motor space, and is learnt during an initial period of self-observation. The use of the VMM is good for the classification and, as an extra advantage, gives the possibility of doing gesture imitation directly.

Although hand posture recognition is in general quite difficult, grasp classification benefits from using extra information. Temporal integration and object-related cues are very useful for recognition. Occlusions and ambiguous positions of the hand can also be solved with temporal information. The observation of a given object "conveys" information about the possible and the most probable grasp types for that object class. Expectations of the hand appearance can also be created.

The results show that it is possible to achieve 100% recognition rates based on this approach. Notably, we avoid using complex schemes for detecting and tracking fine details of the hand on a video sequence. Rather, we rely on the global hand appearance for this purpose.

In our opinion, the results obtained are an encouraging step in the endeavor of understanding the biological grounding of imitation and, at the same time, develop the principles to build more performing and robust machines, able to cope with complex tasks and to interact with humans.

VI. Acknowledgments

The experimental data set was prepared by Matteo Schenatti, Lorenzo Natale, Giorgio Metta and Giulio Sandini [27], LiraLab, University of Genova. This work was (partially) supported by the portuguese Fundação para a Ciência e Tecnologia (FCT), Programa Operacional Sociedade de Informação (POSI) in the frame of QCA III and the EU-Project IST-2000-28159, Mirror, "Mirror Neurons for Recognition", www.lira.dist.unige.it/projects/mirror/.

VII. References

[1] L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Visuomotor neurons: ambiguity of the discharge or 'motor' perception? *International Journal of Psychophysiology*, 35, 2000.

[2] V.S. Ramachandran. Mirror neurons and imitation learning as the driving force behind the great leap forward in human evolution. *Edge*, 69, June 2000.

[3] A. Murata, L. Fadiga, L. Fogassi, V. Gallese, V. Raos, and G. Rizzolatti. Object representation in the ventral premotor cortex (area f5) of the monkey. *Journal of Neurophysiology*, 78(4):2226–2230, October 1997.

[4] J. J. Gibson. *The Ecological Approach to Visual Perception.* Houghton Mifflin, Boston, 1979.

[5] L. Fogassi, V. Gallese, G. Buccino, L. Craighero, L. Fadiga, and G. Rizzolatti. Cortical mechanism for the visual guidance of hand grasping movements in the monkey: A reversible inactivation study. *Brain*, 124(3):571–586, March 2001.

[6] James M. Rehg and Takeo Kanade. Visual tracking of high DOF articulated structures: an application to human hand tracking. In *ECCV (2)*, pages 35–46, 1994.

[7] Ying Wu and Thomas S. Huang. Capturing articulated human hand motion: A divide-and-conquer approach. In *ICCV (1)*, pages 606–611, 1999.

[8] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *ECCV (1)*, pages 329–342, 1996.

[9] D. M. Gavrila. The visual analysis of human movement: A survey. *CVIU*, 73(1):82–98, 1999.
[10] James M. Rehg and Takeo Kanade. Model-based tracking of self-occluding articulated objects. In *ICCV*, pages 612–617, 1995.
[11] Ying Wu and Thomas S. Huang. View-independent recognition of hand postures. In *CVPR*, pages 88–94, June 2000.
[12] S. Schaal. Is imitation learning the route to humanoid robots. *Trends in Cognitive Sciences*, 3(6), 1999.
[13] Andrew H. Fagg. *A Computational Model of the Cortical Mechanisms Involved in Primate Grasping.* PhD thesis, University of Southern California, 1996.
[14] Erhan Oztop. *Modeling the Mirror: Grasp Learning and Action Recognition.* PhD thesis, University of Southern California, August 2002.
[15] M. A. Arbib, A. Billard, M. Iacoboni, and E. Oztop. Synthetic brain imaging: grasping, mirror neurons and imitation. *Neural Networks*, 13:975–997, 2000.
[16] Aude Billard and Maja J. Matarić. A biologically inspired robotic model for learning by imitation. In *International Conference on Autonomous Agents*, Barcelona, 2000.
[17] Aude Billard. Learning motor skills by imitation: A biologically inspired robotic model. *Cybernetics and Systems*, 32:155–193, 2001.
[18] Aude Billard and Stephan Schaal. Robust learning of arm trajectories through human demonstration. In *International Conference on Intelligent Robots and Systems*, pages 734–739, Maui, Hawaii, USA, 2001.
[19] Maja J. Matarić. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In C. Nehaniv and K. Dautenhahn, editors, *Imitation in Animals and Artifacts*. MIT Press, 2000.
[20] P. Andry, P. Gaussier, S. Moga, J.P. Banquet, and J. Nadel. Learning and communication in imitation: An autonomous robot perspective. *IEEE Transaction on Systems, Man and Cybernetics, Part A*, 31(5):431–444, September 2001.
[21] P. Andry, P. Gaussier, and J. Nadel. From sensori-motor development to low-level imitation. In *2nd International Workshop on Epigenetic Robotics*, pages 7–15, 2002.
[22] Yiannis Demiris and Gillian Hayes. Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model. In K. Dautenhahn and C. Nehaniv, editors, *Imitation in Animals and Artifacts*. MIT Press, 2002.
[23] J.S. Bruner. Nature and use of immaturity. *American Psychologist*, 27:687–708, 1972.
[24] Manuel Cabido-Lopes and José Santos-Victor. Visual transformations in gesture imitation: What you see is what you do. In *to appear in International Conference on Robotics and Automation*, Taiwan, 2003.
[25] Paul M. Baggenstoss. Statistical modeling using gaussian mixtures and hmms with matlab. http://www.npt.nuwc.navy.mil/Csf/htmldoc/pdf/.
[26] N. Vlassis and A. Likas. A kurtosis-based dynamic approach to gaussian mixture modeling. *IEEE Trans. Systems, Man, and Cybernetics, Part A*, 29:393–399, 1999.
[27] Matteo Schenatti, Lorenzo Natale, Giorgio Metta, and Giulio Sandini. Object grasping data-set. Lira Lab, University of Genova, Italy, 2003.
[28] CyberGlove. http://www.immersion.com.

Improving robot programming flexibility through physical human - robot interaction

M. Frigola[1], J. Poyatos[1], A. Casals[1] and J. Amat[2]

[1]Dep. Automatic Control
Universitat Politècnica de Catalunya
{manel.frigola, joan.poyatos, alicia.casals}@upc.es

[2]Robotics Institute. (IRI) UPC / CSIC,
Barcelona, SPAIN
josep.amat@upc.es

Abstract. Robotics applications in the services sector require more and more human-robot interaction. This fact has promoted the development of more flexible programming techniques, even extending such improvements to industrial applications. Based on the measure of the force applied over the robot end effector and/or the contact point over its arm, the goal of this work is to provide a means for increasing flexibility in the programming phase, as well as enabling the user to correct, on-line, in the execution phase, a robot programmed trajectory by means of manual guidance. This paper focuses on the study of the deviation of the robot trajectories produced by an external human interaction. The work considers the robot performing complex trajectories, operating under several behavior hypotheses, such as movements with very low inertia or with elastic or plastic reflection.

1. Introduction

The continuous evolution of robotic applications towards the service sector, or in tasks requiring more flexibility, creates the need of providing robots with higher programming performances. In these kind of applications it is common that the presence of a human is required to interact with the robot controlling, guiding or simply supervising the robot operation, due to the still too limited performances of current perception systems and artificial intelligence techniques.

Analyzing the wide scope of programming methods, going from the manual robot guidance programming techniques, or programming by demonstration, up to the high level programming languages, one can find multiple possibilities combining both techniques.

Programming by demonstration techniques enable users that are not computer programming experts, to program a robot based on the human expertise and knowledge of the task itself. Robot programming using computer languages provides the robotic system with computer performances such as high computing capabilities, reference frame changes, sensor based operation etc. Thus changes of some parameters in the program code, from any input device or through an external computer, modify, at convenience, the robotic task.

The availability of performances of both kinds of programming techniques operating simultaneously adds the computing power of computers to the intelligence and adaptability of a human to the working environment that potentially contains unknown parts or non modeled objects. If these complementary performances can be applied operating interactively in real time, the robotic system can react to changes in the environment which are not perceptible by current perception systems, or take decisions depending on the evolution of the robotic task, at levels far away from those achievable by computers, based on deep human knowledge and expertise.

Such interactive programming techniques start to become a real need for robot programming when they operate in human environments, as for instance at home. For this robot programming technique, the common keyboard and mouse become insufficient as described in [1], where a multimodal recognition system detects hand postures and spontaneous speech to design a friendly user-robot interaction at any time both in the programming and in the execution phase.

Service robotics and especially domestic robots, where the user is not an expert robot programmer, impose new needs, as programming by demonstration. In [2] some basic concepts for mapping typical human actions at home are explained, while in [3] the effort is devoted to understand human intention, by analyzing fine motion manipulation through the use of integrated tactile data.

The interaction using vision has been used in [4], with the AVAT system (Adaptive Vision-based Attentive Tracker). The human –machine interaction is achieved isolating intended actions from ordinary walking movements. The use of force sensors in [5] provides the information for following a programmed trajectory carrying an object in cooperation with a human.

The interactive programming procedure developed is based on the detection and measure of the force applied by a human interacting with the robot by pushing, pulling or steering the robot end effector, or its arm, in the programming phase

or at some time during the execution of a task. In this way, the way of programming by demonstration all the task or part of it, enables the user to introduce some adjustments by demonstration as well. This is quite useful taking into account that it can be difficult to achieve the desired results with only one go. Thus, programming parts of the task, or changing them, can be done without interrupting the execution of the task and the improvement of the programmed task can be tested by reiterating the robot actuation as many times as required to get the desired results.

A force sensor on the robot wrist measures the force applied to the robot end effector [4], thus enabling its manual guidance. To deal with risks situations a vision system can detect visually the human intention, from a quick reaction of the operator to some possible incidences, such as potential collisions. This is achieved through the perception of the operator hitting the robot arm.

A previous work demonstrating the potential use of the visual detection on human postures to detect their intention can be found in [6], and its application to robot programming in [7].

2. System configuration

The robotic system with manual programming or movement correction capabilities consists of the robot itself, endowed with a six degrees of freedom sensor located on the wrist, and several cameras strategically located around the working area to visualize the scene and detect possible actions of a human applying a force over the robot arm. Fig. 1 shows the configuration of the robotic system.

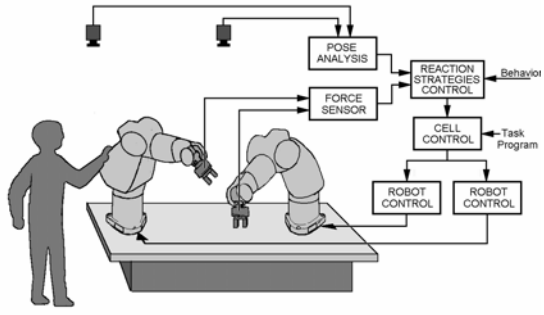


Fig. 1 System configuration

The force sensor mounted on the wrist enables the user not only to program the robot by demonstration during the programming phase, but also to deviate the robot or change its programmed trajectory, on line, during the execution phase. This means that the execution of a previously programmed task can be modified during its execution by a non expert robot user, according to the task needs. This performance provides flexibility when operating with different shaped objects, if it is convenient to change the operation strategy, when it is necessary to adapt the task to unforeseen situations or to avoid collisions.

The force sensor detects the efforts applied over the robot end-effector or the robot handling device. The detection of efforts applied over the robot arm is carried out by the remote vision system, since the robot is not covered by artificial skin that would sense such contacts or pressure all over its body. The vision system developed appreciates human intention by means of their posture and the position of their hands over the robot surface. This interaction is aimed to move the robot apart, for instance when it is going to collide, more than moving precisely the arm to a desired position or executing a given trajectory.

The combination of both kind of perception systems enables a human to interact effectively with a robot, increasing, for some applications, the performances of the commonly used robot programming by demonstration techniques.

During the execution phase the human interaction can affect qualitatively, from the application of a certain force over the arm to move it apart, a hit on the arm, or quantitatively, measuring the forces and pairs that can deviate the robot trajectory at human's will, according to the task and task conditions.

3. Trajectory typologies in programming by demonstration

Analyzing the typologies of the trajectories carried out by a robot, three kinds of tasks have been considered:

- Trajectories that have a fixed origin and destination, and it is possible to interact with them, over the passing points, modifying their flying trajectories (manipulation, palletizing, assembly)
- Trajectories where multiple points or the whole path is relevant, as in spot welding or sealing tasks respectively.
- Trajectories for tasks based on following surfaces, where the task to be carried out is more important than tracing a concrete path (painting, polishing, cleaning…)

In the first two cases the task is composed by a succession of trajectories in space, which are chained to achieve the predefined goals. On the contrary, in the third case, there is not a composition, but a superposition of two or more different trajectories.

On the other hand, in the three cases, the resulting trajectory of the programmed task contains two different components. A component considered as the most relevant, from the point of view of the task, and a secondary one.

In tasks of the two first types, the trajectory is much simpler, while in the third case the whole programming process demands for more steps, which are described in more detail the next subsections. Fig. 2 shows two examples of such trajectories, in a) a pick and place type task, and in b) a polishing type one.

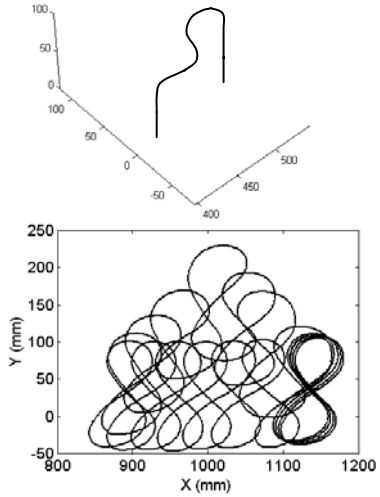


Fig. 2 Trajectory examples. a) a point to point trajectory (first type), and b) a complete complex trajectory (third type).

3. 1 Tasks with simple trajectories

Many tasks can be performed as a sum of subtasks of the type: approximation, grasping, loading, transfer, actuation, assembling, insertion, verification or return. Such composed tasks can be included in the two first above mentioned sets.

Since these trajectories are composed by a sequence of subtrajectories, each one of them can be changed, improved, or suppressed very easily, since they can be clearly separated. For instance, in a pick and place task or in spot welding the working points can be considered as the relevant ones, while the flying trajectory between them can be considered as secondary. In sealing or arc welding tasks the passing points are the relevant ones.

In tasks of these two first typologies the differential factor between the primary and the secondary trajectory components can be the speed. In theses cases, the lower velocities are associated to the precision requirements.

3. 2 Tasks with complex trajectories

In the third case, the primary and secondary components of the task can be associated, the first, to a component of higher frequency that characterizes the type of task, and a second one, of lower frequency that determines the robot run over the part or working area. For instance, a welding cordon is defined by a run over the borders of the two parts to be joined (the second component), that can be modulated by a tooth saw signal (the first), thus making the welding more robust. Another example would be a polishing task. In this application the basic task movements can be circles or a zigzag that sweeps the surface to be polished. Thus, we will name the task specific movement, the modulator signal (the one with the highest frequency), while the robot run all over the surface will be the carrier (the one with lowest frequency). It is necessary to decompose the trajectory in these two components, for each coordinates axis, since the future interaction of a human would have to influence differently over each of them. According to the applied force the robot will deviate or change the programmed run, but the specific task movements would only be modified during the transitory, the time of the interaction of the disturbing force.

3. 2.1 Model of the task movements

The characterization of a model that differentiates the kind of trajectory requires first the characterization of the task.

In such complex tasks, for obtaining the model of the basic movements of the task (the modulator signal), in the programming phase, the user has to repeat several times this movement over a fixed area. Afterwards he can teach the complete task by sweeping the working surface, following the classical programming by demonstration procedure.

Once the task has been programmed, a modulating signal is available. That is, a trajectory P composed by n samples p_i is generated, where for each sample the robot position and the time in which it is sampled is memorized. The data stored for each sample is: $p_i = (x_i, y_i, z_i, t_i)$.

The repetition of the task movements (the modulating signal) by the user is imprecise, describing non coincident trajectories. One such P trajectory is shown in fig. 3.

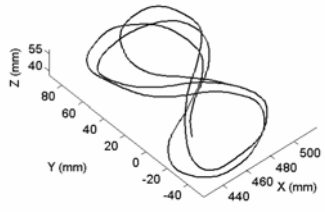


Fig. 3 Non coincident trajectories of the modulating signal

From P, the searched modulating trajectory Q, has to be obtained from the average of the P sections. The proposed method is divided into two steps. In the first step, the time independent path Q_f is

found, and in the second, we generate Q assigning a speed value to each point of Q_f.

From the P trajectory obtained in the learning phase, repeating several times the task, a trajectory P' is obtained. P' follows the same path than P, but all its points p' are equidistant. Fig. 4 shows both the P and P' paths, visualizing respectively a constant time sampling and a constant distance.

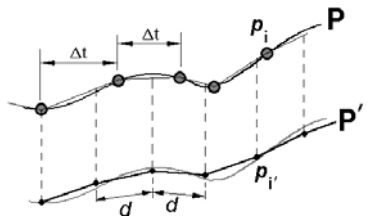


Fig. 4 .Trajectory P' obtained with equidistant space intervals

Two cycles of this trajectory are considered equal if they follow the same path, independently of speed with which the cycle and sampling period have been generated. P' is used to learn the path (Q_f) to be tracked by the robot.

3. 2. 2 Extraction of the period of the modulating signal

In order to obtain the modulating signal Q_f, first, the periodic trajectory in P' has to be found. The process is as follows. First P' is divided into its periodic sections and they are averaged. Each component of P' can have a different number of samples per cycle, but , as they represent a unique trajectory in the 3D space, they behave synchronously, so the periods of all components must be multiple of the smaller one and the largest period is taken as the period of the trajectory

This periodicity can be obtained from autocorrelation [6]. The peaks in the autocorrelation function can be used to determine whether the signal is a periodic sequence, and its period. If the signal is not periodic, peaks occur at aperiodic intervals and their amplitudes decrease rapidly. In fig. 5, the position evolution of a periodic component (a), and the auto-correlation obtained from it (b) are shown.

For each coordinate, the mean value of each component is removed, then the auto-correlation function is applied, and the highest peak (at lag 0) is found. Then, we must look for the next peaks, taking the highest as the one that defines the period of this component. Then the highest period of all the components is taken as the trajectory's period and each coordinate is divided into sections of period T. Each section must be adjusted to get a closed trajectory (a loop). All the trajectory loops are then averaged to get the modulation signal from the movement acquired in the learning phase. The extracted Q_f trajectory is shown in fig. 6.

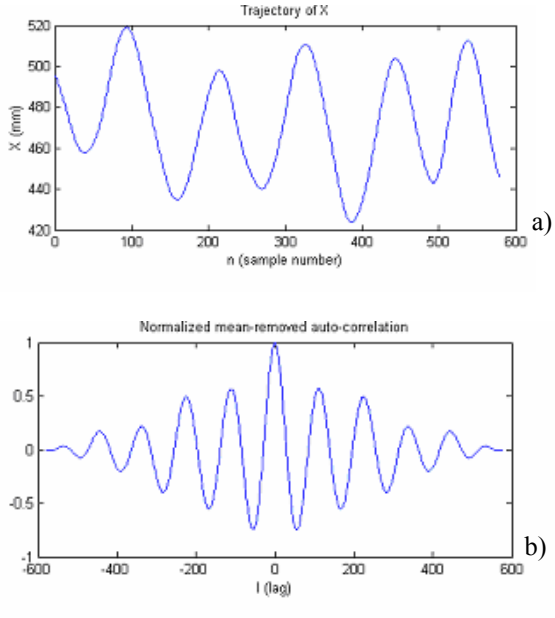


Fig. 5 Evolution a), and auto-correlation b), of a component of P'

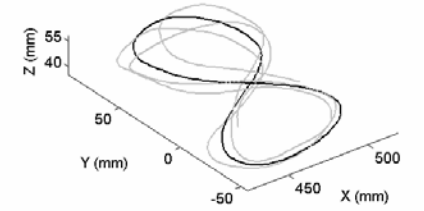


Fig. 6 Average trajectory from several loops

3. 3 Extraction of the carrier signal

The cross-correlation between the modulating signal trajectory and the compound one produces a pick in every position of the compound trajectory where the modulating trajectory starts a cycle. The subtraction of the modulated trajectory to the compound one, gives as result the carrier signal.

4. Human-Robot-Environment interaction

Classical robot programming techniques require propioceptive sensors and an accurate description of an invariant environment. In more advanced robotic systems, specially in those having interaction with other "living" elements in the environment, it is indispensable to provide the system with some environment perception, necessary to develop shared control strategies, able to assure safety, adaptability and cooperation.

In this application, aside from the force sensor located on the wrist, a vision system has been adopted as a sensor able to detect the human presence, not as an obstacle to be avoided, but as an active agent in robot control. The first provides precise measures of the forces exerted over the

robot, while vision gives a qualitative appreciation of the human intended action.

4. 1 Human robot interaction through force sensors

The availability of force and torque sensors in the robot wrist allows the use of new control strategies that can improve the human-robot cooperation, through a better interaction system.

Once a given trajectory has been programmed, it is still possible to interact with the robot to change its predefined path.

In order to fulfil the required task reacting smooth and reliably to human intentional forces, the robot movements are controlled by considering several force components, all focused on the robot tool. Some of these forces are internally generated and provide the robot with a particular behaviour, the others are externals forces applied on the tool, and they are measured by a six degrees of freedom force sensor located on the wrist. Fig. 7 shows a schema of the influence of these forces over the robot behaviour.

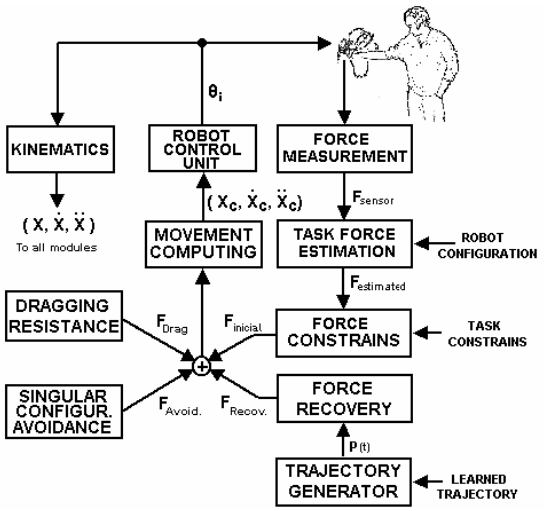


Fig. 7 Teaching by demonstration schema

4.1.1 Force components

The first of these forces to be considered is an internal one, acting as a damping factor that hampers fast movements of the robot tool. The damping factor is simulated as if the robot tool were virtually 'immersed' in a fluid medium of a given viscosity with no gravity influence. This dragging resistance force F_{Drag} opposite to the movement direction is parameterised as follows:

$$F_{Drag} = -\mathbf{D}\,\dot{X}(t) \qquad (1)$$

where $\dot{X}(t)$ is the Cartesian vector velocity of the tool tip and $\boldsymbol{D}$ is a diagonal damping matrix, which components D_{11}, D_{22} and D_{33} are damping factors associated to the translations of the tool and D_{44}, D_{55} and D_{66} are the damping factors associated to its rotations.

The second force is the recovering force F_{Rec} that acts as a spring with a variable stiffness factor whose mission is to move the tool to the designed position *P(t)* specified by the predefined trajectory, after the perturbation produced by a human external force. This force can be expressed as:

$$F_{\mathrm{Recov}} = 2 \cdot \mathbf{K} \cdot \left[Sigmoid(\ \mathbf{N} \cdot (P(t) - X(t))) - 0.5 \right] \qquad (2)$$

where $\boldsymbol{K}$ is a diagonal matrix that indicates the maximum applicable force and torque. The *Sigmoid* function $1/(1+e^{-x})$, applied to the six force components, is used as a means to limit the maximum recovering force, and $\boldsymbol{N}$ is a diagonal matrix factor that adjusts the distance and angle values in the linear range of the sigmoid function (around – 2.0 and +2.0). Here, only a proportional control law is used since it is assumed that the hardware control loop compensates gear frictions and death zones, thus there is no factor.

The third force is the intentional force F_{Intent}, that is, the force that the user can apply to the tool without compromising the task. The user is only allowed to apply the forces with those components that are not artificially constrained by the task. For instance, in a polishing task the forces measured in the tangential plane of the working surface will be understood as user interaction forces. Thus the user can interact with the tool moving it along the surface, but not in its normal direction. The method used here to obtain the intentional force assumes that the force control loop of the specific task will operate maintaining the artificial constrains, characteristic of the task. Under this assumption, the forces not constrained by the task and not predicted (i.e. frictional forces) are estimated as intentional forces as:

$$F_{Intent} = T\_C(X) \cdot (F_{Sensor} - F\_P(X, X', X'', T) \qquad (3)$$

being T_C and F_P the task constrains and force prediction respectively.

The force F_{Sensor} value is the averaged wrist-sensor force measurements, once the offsets of the force sensor and the weight of the tool are eliminated. All forces are referred to the robot reference frame. With such sensor it is possible to extract the weight of the tool, and, once the offsets and the forces and pairs produced by accelerations of the robot are eliminated, the sensor system provides the required data for reacting to external applied forces with almost ideal behaviors [7].

The last force to be considered is F_{Avoid}, which aim is to avoid singular configurations. This internal force pushes or steers the robot a part from singular configurations, from auto-colliding positions or when reaching the articulation limits.

When the robot is close to a joint limit or a singularity, a repulsive force F_{Avoid} is generated in order to move away the robot from this configuration. Thus, F_{Avoid} is the composition of two forces: a force F_{Limits} that pushes the robot

away from any joint limit and F_{Sing} that does the same with configuration singularities.

F_{Limits} is computed in the joint space because this is the space where the joint limits are defined. The direction and magnitude of this force, θs, is given by the intersection of a sphere centred on the current configuration (in the joint space) with all the joints limit surfaces. To do so, the sphere's surface is discretized in a certain number of points. Its representation is shown in fig. 8, for a robot of only three degrees of freedom, for clarity.

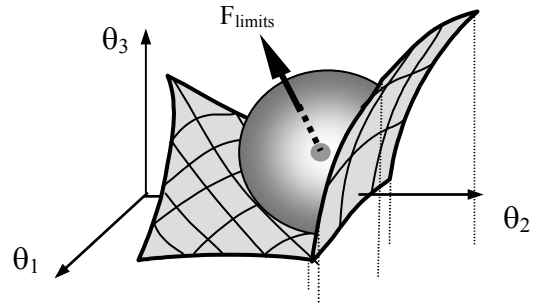


Fig. 8. Representation of the constrained space for a three degrees of freedom robot

A point that is out of the joint limits is called an invalid point θ_i. For each point θ_i a force opposite to the vector that goes from the centre of the sphere θc (current configuration joints) to the point is applied. θ_S is the sum of all these forces:

$$\theta_S = \sum_{i=0}^{M} (\theta_C - \theta_i) \qquad (4)$$

where *M* is the number of invalid points θ_i. Fig. 9 provides, for a given robot trajectory, the proximity to any joint limit. The degree of blackness for each point of the curve indicates the proximity to a joint limit. Once θ_S has been computed in joint coordinates, it must be transformed into a repulsion force in Cartesian coordinates using the Jacobian.

$$F_{limits} = K_L \cdot J(\theta_C) * \theta_S \qquad (5)$$

where K_L is a diagonal matrix used to adjust the amount of repulsion from a joint limit.

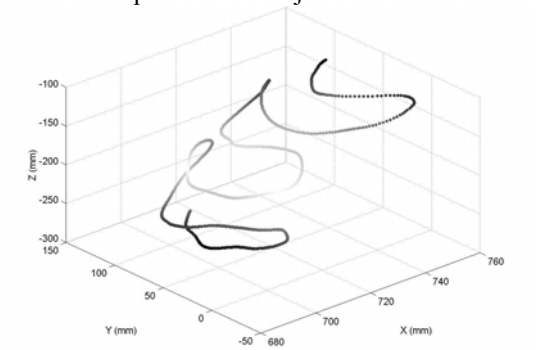


Fig. 9 Visualization of the proximity to a joint limit for a given trajectory

To avoid singular configurations, another repulsion force F_{Sing} is computed when the robot is too close to a singularity. In order to have a measure of how near the robot is to a singular configuration the determinant of the Jacobian matrix is used as follows:

$$w(q) = |\det(J(q))| \qquad (6)$$

The repulsion force F_{Sing} is activated only when w(q) is smaller than a given threshold. When this condition happens the system finds what direction, in cartesian coordinates, produces a larger joint displacement. This is done by computing the increment in the joint variables Δθ caused by an increment in the end-effector position and orientation Δx as follows:

$$\Delta\theta = J^{-1}(\theta_C) * \Delta x \qquad (7)$$

Eq. (7) is evaluated in several (Δx) directions in order to find the one that produces a larger joint displacement ($\Delta\theta_{Max}$). If Δx_{Max} is the direction that moves the manipulator closer to the singular configuration, the opposite direction is used to obtain F_{Sing}:

$$F_{Sing} = K_S * (-\Delta x_{Max}) \qquad (8)$$

where K_S is a constant diagonal matrix that transforms displacements into forces.

4.1.2 Movement computing

The computed movement is the programmed one modified by the possible user's interaction, in such a way that it presents a variable impedance to the user.

This impedance represents the force that the operator perceives as a reaction to his or her action. If this human action is not conditioned by restrictions, the perceived force will be null (infinite impedance), or the one that corresponds to the inertia assigned to the robot.

If the modification of the trajectory due to the human action leads the robot close to a singular point or close to a joint run end, a variation of the trajectory will be generated, that will be perceived by the operator as an opposition force that increases with the proximity to such restriction.

To facilitate experimentation, this computing movement module takes the highest of the forces corresponding to the different restrictions, all of them modelled according to the function indicated in fig. 10, in which the two inflection points are programmed separately in the four modules corresponding to dragging, singular points, inertias and task constrains. In this way it is possible to achieve limitations, more or less progressive (soft or hard) as a function of the minimum distances d_i to each joint.

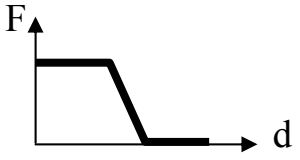


Fig. 10 Function to model the forces

4. 2 Contact detection from visual analysis

In order to detect the direct action of a person over the robot, the vision system used identifies the silhouette of a person and obtaining the relative position of the robot using a simplified model [6].

The extraction of human figures from the working scenario is based on the subtraction of successive gradient images. When the dynamics of the movement of the person is high enough, this procedure gives the best results since the subtraction vector is highly independent of lighting conditions and provides a good movement detection sensitivity [8]. The time interval for the computation of the difference of successive images is carried out at a variable rate, chosen automatically, in order to follow adequately the different dynamics of the robot.

With the aim of detecting and recognizing the upper limb configurations, robustly enough, a simplified cylindrical model of the body is used. The requirement that the detected object fits with the model enables us to automatically reject all the other elements in the scene.

When the vision system detects that the end of the arm contacts the robot, fig. 11, the effect that produces on the robot trajectory is equivalent to a new restriction corresponding to a vertical plane, that is superposed to the other task constraints.

With this criterion the problem of not being able to perceive the action of the person over the robot is avoided, achieving a logical interaction easily assimilated by the operator. In this way, a human operator can contribute to the robot movement as an additional security measure.

Fig. 11 Contact detection. a) One of the two stereo images (original image) b) the processed image (movement detection) c) modeling of a human body and interaction detection

4. 3 Reactive behaviors

The interaction with a robot movement or trajectory can follow two different behaviors, producing either permanent or transitory deviations, at the user's criterion, and in any case it is possible to emulate different elasticity responses. The characterization of such behaviors can be parameterized by two parameters, the equivalent mass of the robot and its elasticity [7].

Therefore, fig 12 shows three different behaviors during the correction of the trajectory in the execution phase. In fig. 12 a) the trajectory changes, with a permanent effect, after human interaction; while in fig 12 b) three different returns to the trajectory B1, B2 and B3, correspond to three different masses considered, and c) shows the smoothing of the trajectory where punctual accelerations produce discontinuities.

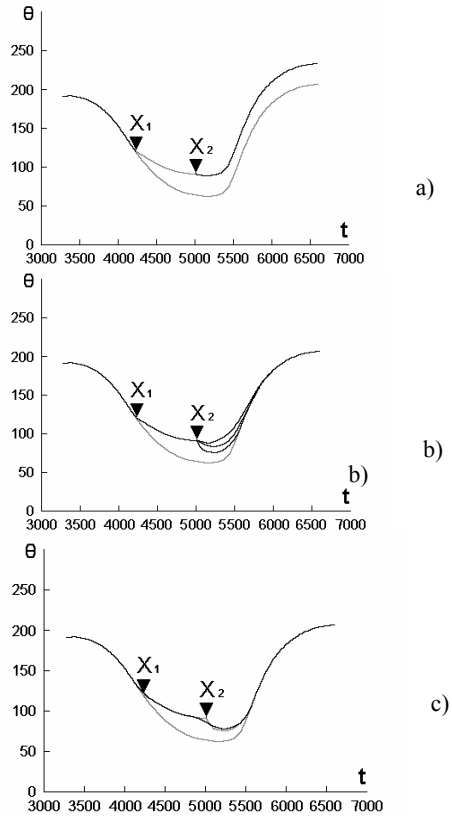


Fig. 12 Trajectory interaction between X_1 and X_2: a) with permanent effect, b) return to the original trajectory with three different responses, and c) a smoothed trajectory with return to the original run

6. Conclusions

In this work several strategies for improving robot programming and reprogramming through the direct interaction of an operator with the robot,

during the programming and the execution phase, have been experimented.

So as to make the interaction with the robot in a natural and intuitive way, the robot is endowed with a force sensor, enabling the user to interact by steering the end-effector, and thus generating the trajectory in the programming phase, as well as to test strategies for changing the programmed trajectories, at any time, by interacting directly with the robot. Thus the program can be adapted to changes, more easily, in real working conditions. In case it is necessary, the user can also modify the robot trajectory touching or hitting the robot arm, from the detection of this action by means of vision. This interaction is less precise; it is conceived as a reflective behavior for task safety and reliability.

The appreciation of the interaction from the action of the force applied over the end-effector, allows to robot to control highly accurately the deviation of the trajectories, since it has available the measure of tridimensional forces and pairs.

The appreciation of the interaction from vision uniquely permits to perceive the contact, but not the magnitude, neither the orientation of the action carried out by the user. In this case, the best behaviors have been obtained using the interpretation of the visually detected contact, as a corrective action that reacts in the opposite direction of the robot movement, perpendicular to the 3D perceived from the operator's arm.

The possibilities of applying such techniques of interaction with the robot trajectories can have their application in service robotics, in which the task to be developed occurs close to people that can interact with the robot, either through the end effector, or just steering the mobile robot through its handles.

Acknowledgements.

The work is being done with the support of CYCIT, Spanish Research Agency, under the projects: DPI2001-0822 and DPI2002-04286-C2-02

References

[1] Iba, S.; Paredis, C.J.J.; Khosla, P.K.*;* **"**Interactive multi-modal robot programming" *IEEE International Conference on Robotics and Automation*, ICRA'02., Volume: 1 , pp. 161 –168, 2002

[2] Ehrenmann, M.; Miner, R.; Rogalla, O.; Dillmann, R.*;* "Programming service tasks in household environments by human demonstration", *11th IEEE International Workshop on Robot and Human Interactive Communicati*on, pp. 460 –467, 2002

[3] Zöllner, R.; Rogalla, O.; Dillmann, R.; "Understanding Users Intention: Understanding Fine Manipulation Tasks by Demonstration", *IEEE/RSJ Int. Conference* on *Intelligent Robots and System, IROS'02.* pp. 1114-1119, 2002

[4] Ho, M.A.T.; Yamada, Y.; Umetani, Y**."**An HMM-based temporal difference learning with model-updating capability for visual tracking of human communicational behaviors" *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pp: 163-168*,* 2002**.**

[5] M.Sato and K.Kosuge, Handling of objects by mobile manipulator in cooperation with human using object trajectory following method. Proc. Int. Conf. on Intelligent Robots and Systems, pp. 541-546, 2000

[6] Amat, J.; Casals, A.; Frigola, M.; "Human Body Acquisition and Modeling for Persons -Humanoid Robots Cooperation" *The Third IARP Workshop on Humanoid and Human Friendly Robotics*, pp: 76-82, 2002

[7] M. Frigola, J. Poyatos, A. Casals and J. Amat, "Force and Contact based Control for Human Robot Interaction", *Inter. Conference on Advanced Robotics, ICAR'03*, pp: , July 2003

[8] Mita, S.K.; "Digital Signal Processing. A Computed-Based Approach" *McGraw-Hill*, 2002.

[9] Xiao, D.; Ghosh, K. ; Xi, N.; Tarn, T. J.; "Sensor Based Hybrid Position-Force Control of a Robot Manipulator in an Uncalibrated Environment", *IEEE Trans. on Control Systems Technology, Vol 8, N. 4*, pp: 635-645, 2000

[10] Amat, J.; Casals, A.; Frigola, M.; "Virtual exoskeleton for telemanipulation", *Lecture Notes in Control and Information Sciences (271) Experimental Robotics VII,* pp. 21-30, 2000

Learning issues in a multi-modal robot-instruction scenario

J. J. Steil, F. Röthling, R. Haschke, and H. Ritter

Bielefeld University, Neuroinformatics Group, Faculty of Technology
P.O.-Box 10 01 31, D-33501 Bielefeld, Germany
{jsteil,helge}@techfak.uni-bielefeld.de

Abstract— **One of the challenges for the realization of future intelligent robots is to design architectures which make user instruction of work tasks by interactive demonstration effective and convenient. A key prerequisite for enhancement of robot learning beyond the level of low-level skill acquisition is situated multi-modal communication. Currently, most existing robot platforms still have to advance to make the development of an integrated learning architecture feasible. We report on the status of the Bielefeld GRAVIS-robot architecture that combines statistical methods, neural networks, and finite state machines into an integrated system for instructing grasping tasks by human-machine interaction. It combines visual attention and gestural instruction with an intelligent interface for speech recognition and linguistic interpretation and a modality fusion module to allow multi-modal task-oriented communication. It further integrates imitation of human hand postures to allow flexible grasping of every-day objects. With respect to this platform, we sketch the concept of a learning architecture based on several interlocking levels with the goal to demonstrate speech-supported imitation learning of grasping.**

I. Introduction

How can we endow robots with enough cognitive capabilities to enable them to serve as multi-functional personal assistants that can easily and intuitively be instructed by the human user? A key role in the realization of this goal plays the ability of *situated learning*: Only, when we can instruct robots to execute desired work tasks by means of a combination of spoken dialog, gestures, and visual demonstration, robots will loose their predominant role as specialists for repeatable tasks and become effective to support humans in every-day live.

A basic element of *situated learning* is the capability to observe and successfully *imitate* actions and – as a prerequisite for that – to establish a common focus of attention with the human instructor. For multi-modal communication, additional perceptive capabilities in the fields of speech understanding, active vision, and in the interpretation of non-verbal cues like gestures or body posture are essential and have to be coordinated and integrated.

We report on progress in building an integrated architecture within the framework of the Special Collaborative Research Unit SFB 360 "Situated Artificial Communicators". In the course of this long term program, many modules implementing partial skills were at first realized and evaluated as stand alone applications [4], [7], [15], [17], [27], but their integration is an additional research task and a key issue towards the realization of complex architectures.

As the development of integrated architectures for real world tasks poses an enormous challenge, there can hardly be found any efforts to scale learning from the lower level of training single skills up to a multi-stage learning across the overall architecture. A primary reason is that most learning approaches rely on highly pre-structured information and search spaces. Prominent examples are supervised learning of target outputs, unsupervised learning of clusters, or learning of control tasks with a (usually small) number of predefined variables (pole balancing, trajectory learning). Here there exist well understood approaches like gradient based learning, support vector machines, vector quantization, or Q-learning, which yield for certain tasks remarkable results, for instance in speech-image integration [21], trajectory learning [10], [16], [36], in object recognition and determination of grasp postures [23], sensor fusion for grasp planning [1], or autonomous grasp optimization [25].

In real world learning a well defined pre-structuring of the data with respect to the given task is an essential part of the learning itself: the system has to find lower-dimensional relevant manifolds in very high dimensional data and to detect important regularities in the course of learning to use these to improve its capabilities. Furthermore, for a complex architecture with many motor degrees of freedom or for cognitive architectures – as the one discussed here – finding a solution by exploration of new actions is not suitable because the search spaces involved are extremely high-dimensional and by far too complex.

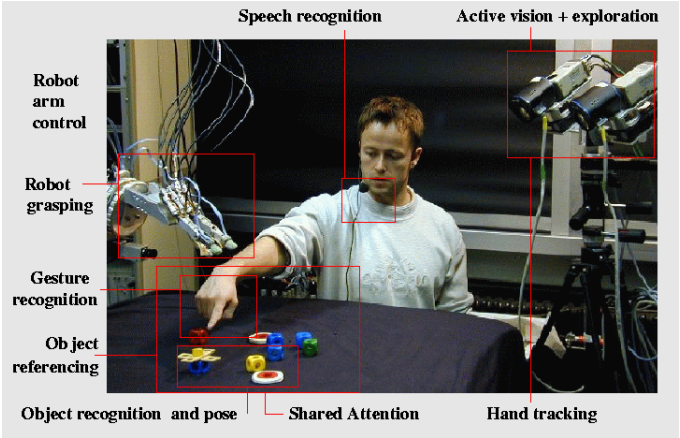


Fig. 1. Interaction with the GRAVIS-system by speech and gesture and some functional components.

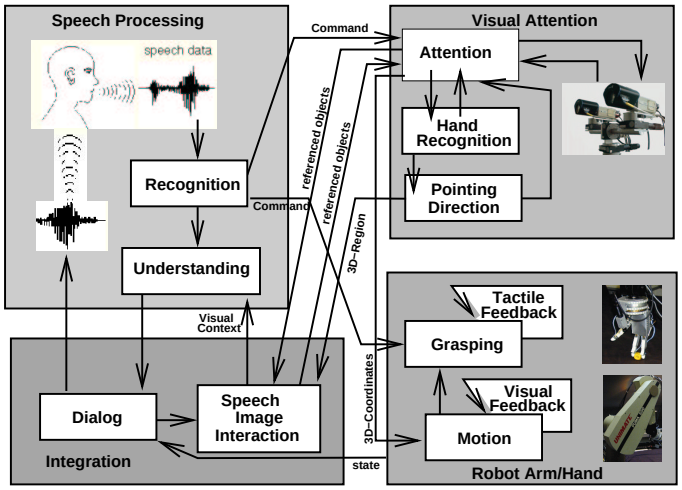


Fig. 2. Schematic picture of the current GRAVIS architecture.

Current practice aims at developing well-scalable, homogeneous and transparent architectures to create complex systems. Ironically, successful examples of this strategy tend to cluster in the small- or mid-size range, while truly large and complex systems seem to defy our wishes for "formatting away" their complexity by good bookkeeping alone. It seems not unlikely that it is one of the hallmarks of complex systems that they confront us with limited homogeneity, evolutionarily grown layers of overlapping functionality and bugs that may even amalgamate with features. Looking at biological systems with their enormous complexity, we see that these by no means resemble orthogonal clockworks; instead, they consist of a tangle of interwoven loops stabilized by numerous mechanisms of error-tolerance and self-repair. This suggests that a major challenge to be met for moving to higher complexity is to successfully adopt similar approaches to come to grips with systems that we cannot analyze in their full detail.

To make learning in such situations feasible, the approach of imitation learning appears very appealing [2], [3], [6], [8], [18], [19], [26], The basic idea is to find a "template" for a successful trajectory by observation of a (human) instructor. This requires (i) to endow the robot system with sufficient perceptive capabilities to visually acquire the action to imitate; (ii) to transfer the observed action into an internal representation, which accounts as well for the system's parameters and copes with the different accessibility of sensor data and the possibly different "instrumentation" with actuators; (iii) to be able to physically execute a suitable action by an actuator.

In our scenario we choose the challenging task to observe human hands in grasping actions, transfer this to an internal perspective to simulate grasping of an anthropomorphic robot hand, and finally to carry out grasping with the robot hand. It is motivated because an important basic capability for intelligent behavior and cognition is the ability to manipulate one's environment purposively. Therefore, we may suspect that the need to control sophisticated manipulators as hands and arms, or a full body is a major driving force for any cognitive processing architecture. In particular, the control of hands is connected with a large number of highly demanding and in many ways generic information processing tasks whose coordination already forms a major base for intelligent behavior.

The following sections give a short overview on the system architecture realized so far and focus especially on the system's learning capabilities. We describe the adaptive methods used, discuss potentials in the architectural design to facilitate high level learning, give an outlook on the planned overall learning architecture, and highlight the first steps we took to enhance our system with capabilities of observation and recognition of human hand postures, simulation of grasping starting from the observed postures, and carrying out the corresponding action with an anthropomorphic robot hand.

II. The current system architecture: Overview

The architecture design is one of the key issues in realizing a complex intelligent robot system. Our entire system is implemented as a larger number of separate processes running in parallel on several workstations and communicating with the distributed architecture communication system (DACS [13]) developed earlier for the purpose of this project. The submodules use different programming languages (C, C++, Tcl/Tk, Neo/NST), various visualization tools, and a variety of processing paradigms ranging from a neurally inspired attention system to statistical and declarative methods for inference and knowledge representation. More details can be found in [27], [20].

Figure 2 shows a coarse overview of the main information processing paths. The speech processing (left) and the attention mechanism (right) provide linguistic and visual/gestural inputs converging in an integration module. It has a short term memory of objects and their 3D-coordinates and passes control to the manipulator if an object is unambiguously referenced by speech or gesture or their combination. Additionally, there are control commands for parts of the system (e.g. *"on", "off", "calibrate skin", "park robot arm", ...*). Some of the modules and their interactions are further described in the following.

A. Robot Arm and Hand

Manipulation is carried out by a standard 6DOF PUMA manipulator operated with the real-time RCCL-command library [29]. It is further equipped with a wrist camera to obtain local visual feedback during the grasping phase. The grasping is carried out by a 9DOF dextrous robot hand developed at the Technical University of Munich [30]. It has three approximately human-sized fingers driven by an oil hydraulics system. The fingertips have custom built fingertip sensors to provide force feedback for control and evaluation of the grasp. Recently we have added a palm and rearranged the fingers in a more human-like configuration in order to allow a larger variety of two- and three-finger grasps and to equip the hand with a tactile display of 8×8 force sensors on the palm.

Starting from the 3D-coordinates determined by the vision and integration modules, approaching movements

and the grasping are executed in a semi-autonomous fashion relying on local feedback only. The grasp sequence starts with an approach movement, centers the manipulator above the object based on visual feedback, chooses a grasp prototype according to the recognized object, aligns the hand along the main axis of the object and executes the grasp prototype. After successful grasping, a similar chain of events allows the robot to put the object down in another gesturally selected location.

B. Visual Attention and Symbol Grounding

A necessary prerequisite for successful human-machine interaction is to establish and maintain a common visual focus of attention between the user and the robot. To enable grasping, we work in full 3D-space and use a binocular active vision head with two 3-chip-CCD color-cameras, controllable pan, tilt, left/right vergence and motorized lenses for focus, zoom and aperture, which combine to a total of 10 DOFs. The basic behavior of the active vision system is – driven by the attention system – to explore the scene autonomously and to search for salient points and objects.

Our attention system places a high emphasis on the spatial organization of visual low level clues and is a more elaborated version of a design proposed in [17]. It consists of a layered system of topographically organized neural maps for integrating different feature maps into a continually updated focus of attention. Similar mechanisms have also been employed in [9], [11], [31]. Currently we use feature maps indicating the presence of oriented edges, HSI-color saturation, intensity, motion (difference map), and skin colors, and a special map for detecting moving skin (mostly hands). As one of the main goals of the system is to A weighted sum of these feature maps is multiplied by a fadeout-map to form a final attention map whose highest peak determines the next fixation. After stereo matching, the resulting loop continuously generates saccades for fixations and this active exploration behavior persists during the whole system operation.

To bridge the lower perceptual level, on which the attention focuses, and the symbolic level, on which linguistic references can be made, we process for both eyes a subimage in the focus of both cameras by a holistic, neural network based object recognition system [15]. If the same object is found and a stereo match can be made, then the corresponding 3D-object coordinates are sent to a short term memory module and such objects can be referenced further by gestures or speech.

C. Speech and Language

To enable speech interaction and communication between the user and the artificial communicator, our system imports a module for understanding speaker independent speech input [12]. The recognition process is directly influenced by a partial parser which provides linguistic and domain-specific restrictions on word sequences derived from previous investigations on a word corpus. Therefore, partial syntactic structures instead of simple word sequences are generated, like e.g. object descriptions

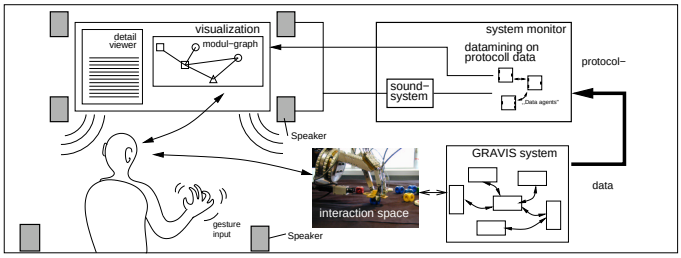


Fig. 3. System monitoring by visual and auditory feedback: the user can obtain a condensed view of the currently active modules together with sounds signaling their current status.

(*"the red cube"*) or spatial relations (*"...in front of..."*). These are combined by the subsequent speech understanding module to form linguistic interpretations. The instructor neither needs to know a special command syntax nor the exact terms or identifiers of the objects. Consequently, the speech understanding system has to face a high degree of *referential uncertainty* from vague meanings, speech recognition errors, and un-modeled language structures.

D. Modality Integration and Dialog

In integration of speech and vision, this referential uncertainty has to be resolved with respect to the visual object memory. Also verbal object descriptions and object recognition results, such as vague attributes (e.g. *"the long, thin stick"*), vague spatial and structural descriptions (e.g. *"the object to the left of the cube", "the cube with the bolt"*) have to be disambiguated. Here the system uses an Bayesian network approach [33], where the different kinds of uncertainties are modeled by conditional probability tables that have been estimated from experimental data. The objects which are denoted in the utterance are those explaining the observed visual and verbal evidences in the Bayesian network with the maximum a-posteriori probability. Additional causal support for an intended object is defined by an optional target region of interest that is provided from the 3D-pointing evaluation. The intended object is then used by the dialog component for system response and manipulator instruction.

The dialog system is based on an investigation of a corpus of human-human and simulated human-machine dialogs [7]. It integrates utterances of the instructor, information of the visible scene, and feedback from the robot to realize a natural, flexible and robust dialog strategy. It is realized within the semantic network language *Ernest*. In particular, it asks for a pointing gesture to resolve ambiguities in the current spoken instruction with respect to the actual state of the memory. The overall goal of this module is to continue the dialog in every situation. Actions which cannot be executed are immediately rejected. For verbal instructions which could not be analyzed a repetition is requested up to two times. If the dialog has gathered too contradictory information the system expresses its confusion and asks for a new instruction.

E. System monitoring and feedback

To add learning or other functionality at the system level one further very important prerequisite is a system for diagnosis and monitoring. Our system currently employs more than thirty distributed processes with many functional submodules such that the detection of errors becomes a nontrivial task. To address this problem is crucial in two respects: first, to generate appropriate feedback for the user to enable him or her to react suitably to the system, e.g. by interrupting, repeating, or restarting an instructive behavior. Secondly, the monitoring of parameters is naturally an important means to analyze and optimize the behavior from an engineering point of view. Thus, we have recently added a system monitoring screen (see Fig. 3), where the activity of modules is visually displayed and, additionally, auditory feedback can be generated. All functional modules send protocol messages to this application to transmit results, useful status information together with timestamps, which enables us to carry out an analysis of the time-behavior of the complete system.

F. Action and imitation

The system as described above allows interactively guided and visually instructed pick-and-place operations based on (i) autonomous exploration of the attention system to transfer visually recognized objects into memory; (ii) subsequent spoken instructions to take objects (with reference to their spatial relationships) which are integrated with 3D-pointing gestures; (iii) a grasping sequence [20]. The system monitor allows a detailed analysis of the temporal behavior of such sequences like in Fig. 4, where a grasp with a a stored pre-shape is chosen with respect to an object recognized by the hand camera followed by a tactile feedback based closing phase. In [20], the objects had to be trained in the recognition system to allow to choose from visual feedback a corresponding pre-programmed grasp.

To gain flexibility in grasping, we reduce the object recognition to the task to find the center of the object and optionally the orientation of its main axis. Then we use imitation of visually perceived human hand postures to choose pre-grasp templates instead of stored distinct grasp strategies for each of the known objects. This approach (detailed in Sections IV, V below) now allows grasping of a large number of objects with different sizes and various shape as shown in Fig. 8.

III. System design, adaptivity, and robustness

Our architecture has integrated a larger number of modules, which have been developed in the course of several years in different research contexts and under different programming environments and languages ranging form C/C++ to an object-oriented graphical environment, Neo/NST, which has been developed in-house during the recent years. Many of these modules originally have not been developed in view of being utilized in the described system. Thus, the ideal perspective to define constraints and a unified framework beforehand to facilitate building

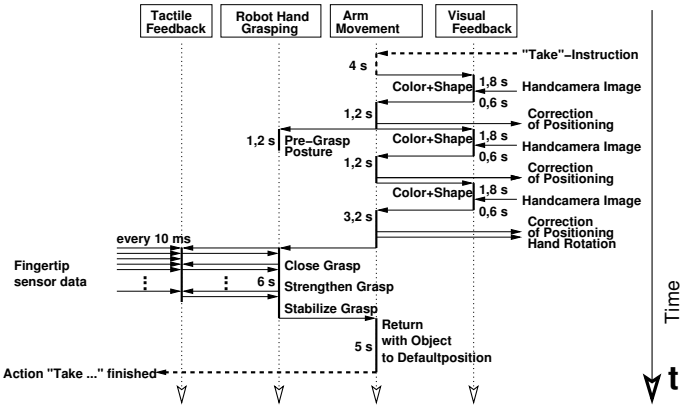


Fig. 4. Evaluation of predefined grasp sequence.

a cognitive learning architecture had to be replaced by an "evolutionary approach" to integrate modules as heterogeneous components in the form they were made available to us. Therefore it is rather the level of the architecture that has to support this integration to provide the environment of an evolutionarily growing system with layers of overlapping and possible redundant functionality. In our case, this led to the development of a rather flexible architecture where our modules mainly communicate by message passing and, once activated, rely on local feedback mechanisms, and which allows to add newly developed skills quite easily.

From the perspective of learning, our architecture is affected at three levels: the lowest level concerns the incremental shaping of basis and fallback behaviors in order to provide a robust set of primitives on which to build the higher-order capabilities of the system. This iterative shaping and refinement may be viewed as a (human-assisted) analogue to an evolutionary learning process and paves the way for the use of learning methods in the more traditional sense on the next higher levels.

The intermediate level is concerned with "adaptivity in the small". This takes the form of various adaptive properties of single modules to calibrate and refine their parameters during use, e.g. an adaptive weighting of the feature maps in the saccadic system or the training of neural networks for gesture recognition and classification and the object recognition together with a fast online color calibration. Here, adaptivity is confined to act "locally", within individual modules and, therefore, largely bypassing any difficult credit assignment problems. Still, from a systems perspective such "local adaptivity" can be extremely valuable, since it endows modules with a degree of "elasticity" against perturbations in their working conditions such the overall system robustness is enhanced.

The higher level finally is concerned with "adaptivity in the large". By this we mean the adaptive coordination of several modules to achieve new action sequences at the task level. We attempt to realize this adaptivity by providing the system with interfaces to flexibly "bind" to observed action structures in order to imitate them. This seems a much more realistic way than trying to self-organize comparable capabilities from pure trial-and-error. Currently, we

focus on enabling the system to bind to visually perceived hand postures. This allows a human to trigger grasping sequences of the robot arm by informing the system with a hand pre-shape posture about the grasp to use for picking up an object. The fine-adjustment of the visually instructed hand pre-shape to the target object can then be carried out with the aid of the local hand camera that extracts information about the relative orientation between manipulator and object.

Thus our design philosophy is to proceed in an evolutionary way and consequently to require newly developed methods and modules to have adaptive capabilities, basic fallback behaviors, and a certain robustness against pure data quality together with sparse communication needs rather than to restrict them to obey predefined programming paradigm or strong architectural constraints.

IV. Towards a learning architecture

To enhance our system with an integrated learning architecture on the system level we propose three subsequent stages. Each stage aims at restricting the search space for the to-be-learned action as much as possible.

At the level of **cognition and imitation** the system's learning is targeted at observing the environment, respectively the instructor, in order to *imitate successful action sequences*, see e.g. [26]. Key issues are (i) the extraction of relevant features, events, and chains of observed partial actions, (ii) their translation from the observed to an intrinsic perspective, and (iii) their exploitation for focusing further exploration to promising regions of the a-priori very high-dimensional search space. Here we plan to use *search space restricted reinforcement learning*, where the exploration of actions can be restricted to a neighborhood of the observed successful trajectory to reduce the otherwise unreasonably long learning phases of classical reinforcement learning. This combination of imitation and reinforcement learning is very flexible: the neighborhood can be chosen small where highly reliable observations are available whereas more exploration may be needed where poor data are given. A typical case in our scenario is the initialization of a grasping sequence with respect to the approach direction and hand pre-shaping for the fingertip trajectories based on visual observation of a grasping human hand, which can be obtained by earlier developed hand and fingertip recognition methods (see Section V).

At the level of **action selection and exploration** learning takes place from an "intrinsic" perspective ("simulation") of a possible action. Based on the search space restrictions gained from the first stage, the focus of this second stage is on *exploration of details* and to include available model knowledge (e.g.. about constraints of the used hardware) to generate simulated actions to be verified later. Here we use a dynamics based grasping simulation to apply classical reinforcement learning for action choice. Fig. **??** shows the simulation of grasping a complex object with the TUM-hand. Due to the lack of sufficient tactile sensors in the TUM-hand, it is only this setting where we can obtain full information about joint angles, applied

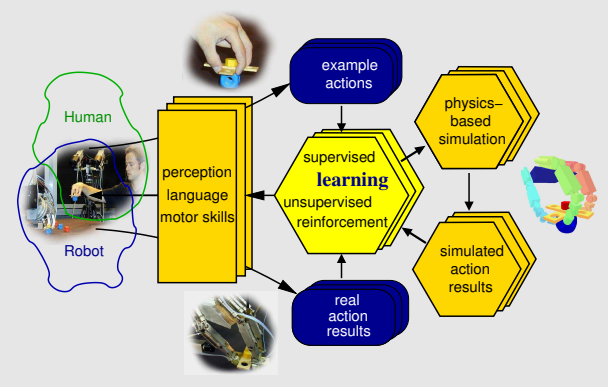


Fig. 5. Multistage learning architecture for integration of imitation learning, reinforcement learning, and statistical learning.

wrenches, or contact points, such that we can measure the grasp quality to generate an suitable reinforcement signal.

At the **sensorimotor level**, actions are executed which seem promising with respect to the results of the previous stages. Here we can distinguish two cases with different evaluation criteria: (i) a maximization of knowledge gain has the consequence of risky actions, for instance near decision boundaries; (ii) a maximization of robustness chooses conservative actions (in maximal distance to decision boundaries). On this level, we use standard methods of statistical learning.

A crucial element to connect the different stages should be an *attention driven focus of plasticity*: standard learning methods are based on the minimization of an error function and distribute incremental learning steps to many parameters. This often leads to unnecessary interference which, however, often can be reduced by the employment of local learning. The concept of attention driven plasticity heads in this direction: We propose that learning can be usefully focused by forming flexible and context dependent groups of parameters whose plasticity is made contingent with respect to a particular situation. This is a way to include prior knowledge to circumvent the credit assignment problem of standard learning algorithms. It can be interpreted as a form of attentional control of the learning process and can be modulated by linguistic inputs of the user.

Fig. 5 shows the interaction of the different stages. The interaction with the user proceeds by means of a *simulation driven clarification dialog* (already used in the system for resolving ambiguities occurring in the speech-image integration). This changes robot learning into an interactive situated learning process, which uses speech and the multimodal perceptual channels for an effective optimization of the system's exploration.

V. First steps to imitation: Observation, simulation, and control of hand posture

We have started the realization of the described integrated learning architecture for grasping tasks like shown in Fig. 6: we generate a robot hand posture from visual observation and 3D-identification of a human hand posture. So we are able to perceive an example action illustrated

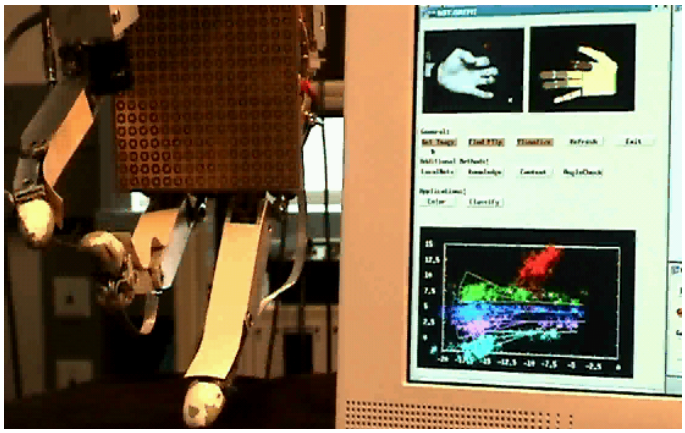


Fig. 6. Observation and recognition of continuous human hand postures to obtain an initial pre-grasp posture for autonomous grasping of the anthropomorphic robot hand. In the upper right part the observed human hand is shown on the screen together with the reconstructed hand posture visualized by the rendered computer hand. Below is shown the operation of the PSOM network which obtains the inverse transform from fingertip positions found from observation in the 2D-images to the joint angles of the hand. To the left the respective TUM hand posture is shown.

as the first stage in Fig. 5. This identified posture can be executed with the real robot hand and is also used as an initial posture for the internal physics-based simulation of a corresponding grasp as shown in Fig. **??**. Therefore all three aforementioned stages are already present in our setup.

The **hand posture recognition** uses a system for visual recognition of arbitrary hand postures which was previously developed in our group. It works in real time, however, currently is restricted to a predefined viewing area. We shortly describe the underlying multi-stage hierarchy of neural networks which first detect the fingertips in the hand image and then reconstruct the joint angles from the fingertip positions, for more details see [22], [24].

In a first step, the pixel image is transformed into a lower-dimensional feature vector of "neural activities" which are responses of Gabor filters with different resolution and orientation. The resulting activity pattern provides an initial, "holistic" input representation of the image, from which we compute a set of meaningful and stable object features – the 2D finger tip locations in the image. For this subtask we use a processing hierarchy in which a neural network first computes a coarse estimate of the centers of up to five image subregions where the finger tips should be located. Subsequently, we provide for each finger subregion an extra network, applying an analogous processing as on the first stage, but focused on the selected finger subregion that was identified by the first, "global" network. It is followed by a post processing which a Gaussian confidence measure for the presence of the corresponding tip and a simple "finger tip filter" that consists of a 5×5 pixel template of a typical "finger tip edge".

In the final step, the obtained 2D-features (finger tip locations) are analyzed to identify the 3D-hand posture. By using the the fact, that human finger joints are highly correlated, we circumvent the problem of the many available degrees of freedom of a 3D-hand. For the corresponding inverse transformation a *Parameterized Self-organizing Map, PSOM* is trained, using data from the analytically computable forward transform. The PSOM is a generalization of the well-known Self-organizing map (SOM) [35]. It replaces the discrete lattice of the SOM with a continuous manifold and allows to retrieve the inverse of each learned mapping automatically.

To **actuate the robot-hand** the reconstructed 3D-joint coordinates, however, can not directly be used, because the robot hand shown in Fig.6 has three fingers only and differs in hand and finger sizes, proportions of the phalanges, and the arrangement of the fingers at the palm. Additionally the sensory equipment and control concepts differ, such that we have to transform the observed joint angles in a way, that we obtain an "equivalent" posture of the robot hand. Geometrically this transformation the different joint angle workspaces and reflects the kinematic constraints imposed by the coupled joints of the robot hand. With respect to executing of a grasp the we have to compensate the lack of direct joint angle measurements by means of a force feedback obtained from the custom built fingertip sensors and a recently added palm sensor (see Fig. 6). With the latter we can evaluate the form of a touched object when carrying out a power grasp, while the fingertip sensors have to be used for precision grasps.

With respect to control we mainly rely on piston potentiometers located at the base station of the hydraulic system. Here we have to cope with hysteresis and nonlinearities due to the long distance of 2.5m between the base station and the finger pistons. We also face sticking and sliding effects caused by return springs integrated in the finger pistons. Nevertheless we achieve an accuracy of about 2 degrees in every joint, which is not enough for a reliable position control but allows a qualitatively sufficient positioning of the fingers to realize suitable pre-grasp postures.

Because of these incompatibilities between a human and our robot hand, it is important to simulate the grasping process before attempting its actual execution. Nevertheless the hand posture recognition is important to get an object and situation specific pre-grasp posture which is an

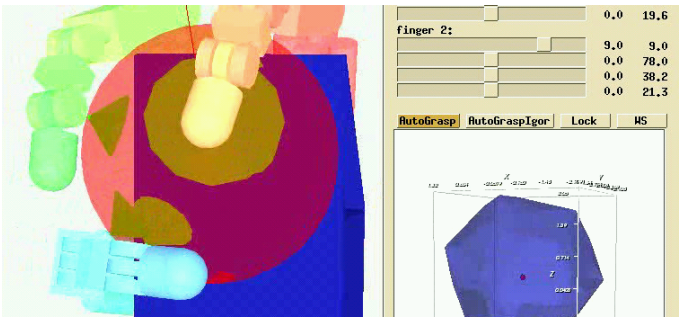


Fig. 7. Evaluation of a TUM-hand grasp in physics-based simulation using contact models with friction (left) and the grasp polytope evaluating force closure (lower right).

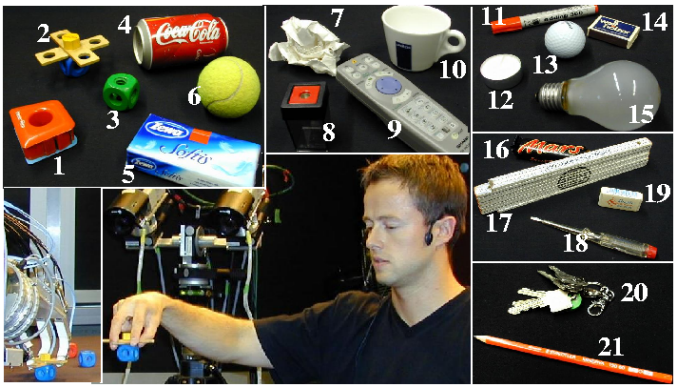
essential requirement for successful grasping and is needed as a starting point for the internal simulation as well. The simulation serves as a filter which adapts the observed pre-grasp position and (in later stages) the grasping strategy to the robots' intrinsic hardware constraints described above.

For **internal simulation** we utilize the real-time dynamics-based package Vortex [32], which allows accurate object motion and interaction based on Newtonian physics. We extended the package to provide static and dynamic friction for contacts, which is crucial for successful grasp simulation. Although contacts are simulated on the basis of point contacts and thus are necessarily coarse, they provide full force feedback, which is not available with our real world tactile fingertip and palm sensors. Hence, in simulation, it is possible to evaluate a successful grasp with respect to measures such as form or force closure [5], which we evaluate by numerical solution of Linear Matrix Inequalities as recently suggested in [14]. Fig. 7 illustrates the friction cones of a successful grasp together with the resulting polytope of applicable forces.

VI. Results for imitation based grasping

When executing the grasp different levels of imitation are possible reaching from the task level down to the realization of similar joint trajectories of the fingers. At an intermediate level, we use observation of the human hand posture to define a pre-grasp position, starting from which we coherently close all fingers until they get in contact with the object, what is detect by the fingertip force sensors. We have also tried to directly transform the observed and reconstructed 3D-joint angles to the robot hand, however with very limited success because this transfer to the very different hardware lead to uncoordinated finger movements. Therefore the imitation in our case leads to a choice between one of a number of possible grasping strategies. This reduction of complexity turns out to be very efficient and allows to grasp many every-day objects of different shape an size as shown in Fig. 8.

The results reveal some interesting details. The object number 2 (propeller) cannot be grasped by any of the standard prototypic grasps and shows that complex shapes can need more specialized grasps. To generate such grasps online, additional information from observation in form of e.g. grasping points could be useful. Furthermore, many objects can be grasp with more than one strategy or in special positions only and in these cases imitation introduces otherwise unknown context knowledge. The preliminary experiments also showed that often small changes in the pre-grasp posture have a large impact on the success rate. These results show that despite the previous reduction of the search space by obtaining a suitable pre-grasp position there is much room for improvements by simulation as proposed in Sections IV/V. Possible free parameters to be evaluated in such simulation are the exact initial joint angles of the fingers, the exact relative position of the hand to the object, the closing speeds of the fingers, etc.



nr.	power	thumb/2.	thumb/2./3.	2./3.	rotate
1	10	-	+	-	+
3	(+)	-	+	10	+
4	10	-	-	-	-
5	10	-	+	-	+
6	10	-	+	-	+
7	9	-	(+)	-	+
8	+	(+)	8	(+)	+
9	8	-	-	-	5
10	9	-	-	-	+
11	-	7	-	-	5
12	-	-	6	-	+
13	7	-	-	-	+
14	+	(+)	7	(+)	+
15	6	-	(+)	-	4
16	-	-		5	4
17	-	4	-	-	3
18	-	3	-	-	2
19	-	4	-	-	+
20	-	-	0	-	-
21	-	0	-	-	-

Fig. 8. Imitation based grasping of every-day objects sorted with respects to 10 grasp attempts using the most suitable strategy, which was determined by an preliminary experiment and is indicated as [-] — grasp not possible, [(+)] — possible but only in non-general position, [+] possible, the propeller (no. 2) needs a specialized grasp. The final column gives the number of trails, which are robust against rotations of the hand after lifting up.

VII. Conclusions

Our initial assumption is that situated and multi-modal communication is a key prerequisite for learning in artificial intelligent perception-action systems. Thus, we will proceed with the development of the current platform and use it as a basis for a systematic design of a learning architecture. The longer term goal is to demonstrate speech enabled imitation learning for instructing grasping tasks (see Fig. **??**), because multi-fingered grasping combines many of the highly developed capabilities of the human cognitive system: the recognition of object shape, type, position and orientation in space; the respective and for the intended task appropriate choice of a grasp; the actual realization of the grasp under complex kinematic constraints; and the following immediate optimization of finger positions and force with respect to grasp stability and manipulability.

We believe that this research program is promising if a sufficiently developed technological basis is available. This basis seems crucial for higher level architectures and includes sophisticated hardware for data acquisition and action like an articulated dextrous hand as well as algorithms

for robust implementation of the perceptual skills. In particular for the imitation of grasps, we expect in the nearer future progress from improvements in the field of multi-fingered hands, especially with respect to robustness and tactile sensing. Concerning intelligent control it is important to have at our disposal a sufficiently high number of robust and adaptive partial skills, a prerequisite toward which many efforts have been made in the course of the Special Collaborative Research Unit SFB 360.

The key towards an integrated architecture now is a systematic design, which endows the system with a fruitful interrelation of different aspects of learning and their various techniques on the different levels to generate a flexible and incrementally improving combination of these partial skills and modules. Here we see the main focus of the sketched learning architecture, knowing that this goal may be reached only by long term efforts and in incremental steps. We are aware, that in view of the enormous complexity of the respective challenges, this research program also calls for a close collaboration of robotics with neighboring disciplines like neurobiology or cognitive science and we expect many insights and inspirations from these fields.

Acknowledgments:

Among many people who contributed to the robot system, we thank in particular G.Fink, J. Fritsch, G. Heidemann, T. Hermann, J. Jockusch, N. Jungclaus, F. Lömker, P. McGuire, R. Rae, G. Sagerer, S. Wrede, S. Wachsmuth, J. Walter. For further contributions of the SFB 360 "Situated Artificial Communicators" and the neuroinformatics and practical informatics groups at the Faculty of Technology of the Bielefeld University see the references.

References

[1] P. K. Allen, A. Miller, P. Y. Oh, and B. Leibowitz, "Integration of vision, force and tactile sensing for grasping," *Int. J. Intelligent Machines*, vol. 4, no. 1, pp. 129–149, 1999.

[2] P. Andry, P. Gaussier, S. Moga, J. P. Banquet, and J. Nadel, "Learning and communication via imitation: An autonomous robot perspective," *IEEE SMC*, vol. 31, pp. 431–442, 2001.

[3] P. Bakker and Y. Kuniyoshi, "Robot see, robot do : An overview of robot imitation," in *Proc. AISB Workshop on Learning in Robots and Animals*, Brighton, 1996, pp. 3–11.

[4] C. Bauckhage, G. A. Fink, J. Fritsch, F. Kummert, F. Lömker, G. Sagerer, and S. Wachsmuth, "An Integrated System for Cooperative Man-Machine Interaction," in *IEEE Int. Symp. on Comp. Int. in Robotics and Automation*, 2001, pp. 328–333.

[5] A. Bicchi and V. Kumar. "Robotic grasping and contact: A review", In *Proc. ICRA*, 2000, pp. 348–353.

[6] A. Billard and M. J. Mataric "A biologically inspired robotic model for learning by imitation," Proc. 4. Int. Conf. on Autonomous agents, Barcelona, Spain, 2000.

[7] H. Brandt-Pook, G. A. Fink, S. Wachsmuth, and G. Sagerer, "Integrated recognition and interpretation of speech for a construction task domain," in *Proc. Int. Conf. on Human-Computer Interaction*, 1999, vol. 1, pp. 550–554.

[8] C. Breazeal and B. Scassellati, "Challenges in building robots that imitate people," in *Imitation in Animals and Artifacts*, K. Dautenhahn and C. Nehaniv, Eds. MIT Press.

[9] C. Breazeal and B. Scassellati, "A context-dependent attention system for a social robot," in *Proc.IJCAI*, 1999, pp. 1146–1151.

[10] R. Dillmann, M. Kaiser, and A. Ude, "Acquisition of elementary robot skills from human demonstration," in *In Int. Symp. on Intelligent Robotic Systems*, Pisa, Italy, 1995, pp. 185–192.

[11] Joseph A. Driscoll, Richard Alan Peters II, and Kyle R. Cave, "A visual attention network for a humanoid robot," in *Proc. IROS*, Victoria, B.C., 1998.

[12] G. A. Fink, "Developing HMM-based recognizers with ESMERALDA," in *LN in AI*, V. Matoušek, P. Mautner, J. Ocelíková, and P. Sojka, Eds., Berlin, 1999, vol. 1692, pp. 229–234.

[13] G.A. Fink, N. Jungclaus, H. Ritter, and G. Sagerer, "A communication framework for heterogeneous distributed pattern analysis," in *Int. Conf. on Algorithms and Architectures for Parallel Processing*, Brisbane, 1995, pp. 881–890.

[14] L. Han and J. C. Trinkle and Z. X. Li, "Grasp Analysis as Linear Matrix Inequality Problems" *IEEE. Trans. on Robotics and Automation*, vol. 16, no. 6, pp. 663–673, 2000.

[15] G. Heidemann, D. Lücke, and H. Ritter, "A system for various visual classification tasks based on neural networks," in *Proc. ICPR, Barcelona*, A. Sanfeliu et al., Ed., 2000, pp. 9–12.

[16] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Trajectory formation for imitation with nonlinear dynamical systems," in *Proc. IROS*, 2001, pp. 752–757.

[17] N. Jungclaus, R. Rae, and H. Ritter, "An integrated system for advanced human-computer interaction," in *UCSB-Workshop on Signals and Images*, USA, 1998, pp. 93–97.

[18] 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.

[19] M. J. Mataric, O. C. Jenkins, A. Fod, and V. Zordan, "Control and imitation in humanoids," in *AAAI Fall Symposium on Simulating Human Agents*, North Falmouth, MA, 2000.

[20] P. McGuire, F. Fritsch, J. J. Steil, F. Röthling, G. A. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter, "Multi-modal human-machine communication for instructing robot grasping tasks," in *Proc. IROS*, 2002, pp. 1082–1089.

[21] P. McKevitt, Ed., *Integration of natural language and vision processing*, Kluver, Dordrecht, 1994.

[22] C. Nölker and H. Ritter "Visual Recognition of Continuous Hand Postures" *IEEE Trans. NN*, vol. 13, no. 4, pp. 983–994, 2002.

[23] J. Pauli, "Learning to recognize and grasp objects," *Autonomous Robots*, vol. 5, pp. 407–420, 1998.

[24] H. Ritter, J. J. Steil, C. Nölker, F. Röthling, and P. McGuire, "Neural Architectures for Robot Intelligence," in Rev. Neurosci., vol. 14, no. 1-2, pp. 121-143, 2003.

[25] B. Roessler, J. Zhang, and M. Hoechsmann, "Visual Guided Grasping and Generalization Using Self-Valuing Learning," in *Proc. IROS*, 2002,pp. 944–949.

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

[27] J. J. Steil, G. Heidemann, J. Jockusch, R.Rae, N. Jungclaus, and H. Ritter, "Guiding attention for grasping tasks by gestural instruction: The GRAVIS-robot architecture," in *Proc. IROS 2001*. IEEE, 2001, pp. 1570–1577.

[28] Joseph O'Sullivan, "Towards a robot learning architecture," in *Learning Action Models* , Wei-Min Shen, Ed., 1993, pp. 47–51, AAAI Press, 1993.

[29] J. Lloyd and M. Parker, "Real time control under Unix for RCCL," in *Robotics and Manufacturing ISRAM'90*, 1990, vol. 3, pp. 237–242.

[30] R. Menzel, K. Woelfl, and F. Pfeiffer, "The development of a hydraulic hand," in *Proc. 2. Conf. Mechatronics and Robotics*, 1993, pp. 225–238.

[31] S. Vijayakumar, J. Conradt, T. Shibata, and S. Schaal, "Overt visual attention for a humanoid robot," in *Proc IROS*, 2001, pp. 2332–2337.

[32] CMLabs "Vortex – physics engine for real-time simulation", www.cm-labs.com

[33] S. Wachsmuth, H. Brandt-Pook, G. Socher, F. Kummert, and G. Sagerer, "Multilevel integration of vision and speech understanding using Bayesian networks," in *Computer Vision Systems: 1. Int. Conf.*, H. I. Christensen, Ed., 1999, pp. 231–254.

[34] S. Wachsmuth, and G. Sagerer, "Bayesian Networks for Speech and Image Integration," in *Proc. of 18th National Conf. on Artificial Intelligence*, Edmonton, 2002, pp. 300–306.

[35] J. Walter and C. Nölker and H. Ritter, "The PSOM Algorithm and Applications", *Proc. Symposion Neural Computation (Berlin)*, 2000, pp. 758–764.

[36] M. Yeasin and S. Chaudhuri, "Toward Automatic robot programming: Learning Human Skill from Visual Data," *IEEE SMC*, vol. 30, no. 1, pp. 180–185, 2000.

Towards Integrating Learning by Demonstration and Learning by Instruction in a Multimodal Robot

Stefan Wermter, Mark Elshaw, Cornelius Weber, Christo Panchev, Harry Erwin
Centre for Hybrid Intelligent Systems
School of Computing and Technology
University of Sunderland
St Peter's Way, Sunderland
UK
www.his.sunderland.ac.uk

***Abstract*— Learning by demonstration and learning by instruction offers a potentially more powerful paradigm than programming robots directly for specific tasks. Learning in humans or primates substantially benefits from demonstration of actions or instruction by language in the appropriate context and there is initial neurocognitive cortical evidence for such processes. Cortical assemblies have been identified in the cortex that activate in response to the performance of motor tasks at a semantic level. This evidence supports that such mirror neuron assemblies are involved in actions, observing actions and communicating actions. Furthermore, neurocognitive evidence supports that cell assemblies are activated in different regions of the brain dependent on the action type being processed. Based on this neurocognitive evidence we have begun to design a neural robot in the MirrorBot project that is based on multimodal integration and topological organisation of actions using associative memory. As part of these studies in this paper we describe a self-organising model that clusters actions into different locations dependent on the body part they are associated with. In particular, we use actual sensor readings from the MIRA robot to represent semantic features of the action verbs. Furthermore, ongoing work focuses on integration of motor, vision and language representations for learning from demonstration and language instruction.**

I. INTRODUCTION

Often robots are restricted in their general autonomous behaviour and only perform what has been preprogrammed. We begin to see initial research in learning by language instruction or action demonstration (e.g. Billard 2002 [2] , Demiris and Hayes 2002 [5]). However, so far, demonstration and language instruction have only played a minor role in intelligent robotics. Some robots like the tour-guide robot Rhino [4] have been quite robust in terms of their localisation and navigation behaviour. However they do not use learning by demonstration or learning from language instructions. Although the conversation office robot jijo-2 [1] can be instructed to navigate to certain landmarks and the Minerva tour-guide [17] interacts by using simply preprogrammed speech, they are restricted in their ability to learn. Furthermore, the Kismet interactive robot [3] can recognise and represent emotions using a sophisticated head but does not learn by imitation or instruction.

Learning through imitation has been a useful approach for primates and therefore is an active research into the area of learning in robots. For instance Demiris and Hayes 2002 [5] and Maistros and Hayes 2001 [10] have devised approaches based on the mirror neuron concept to achieve robot learning through imitation. Demiris and Hayes 2002 [5] use behaviour and forward models in their approach where a demonstrator robot was observed by the imitator robot performing actions and then is required to predict what is being performed. Maistros and Hayes 2001 [10] use the Scheme Theory to express the function of the mirror neurons to achieve learning by imitation of grasping actions. This was done by using as demonstrator-imitation scenario in a similar manner to that by Demiris and Hayes 2002 [5]. Billard 2002 [2] also considers the use of imitation to aid autonomous robot communication learning of a proto-language by using an unsupervised approach based on a dynamic recurrent associative memory architecture. Language is learnt through a student robot recreating the actions of the teacher robot, through the teacher robot describing what it observes and the student robot having a similar perspective to the teacher. Gaussier et al. 2001 [9] have considered the use of a neural network approach that is able to achieve learning and communication through imitation. In doing so they concentrate on low-level imitations that recreate simple movements that are found in infants.

In our approach (e.g. Wermter and Panchev 2002 [22], Wermter et al. 2001 [20], Wermter and Elshaw 2003 [21], Weber and Wermter 2003 [18]) we study learning in intelligent robots based on some evidence from the brain since it obviously supports learning from demonstration and learning from verbal instructions in humans. In this particular study here in the context of the MirrorBot project we focus on two constraints: first multimodal learning and integration of action, vision and language and second the topological arrangement of actions and

their visual and language counterparts.

First, multimodal learning, recently, a class of neurons has been found in the rostral part of the ventral premotor cortex (area F5) in monkeys that are active both when a monkey handles an object and when it observes an experimenter performing similar actions [14]. More recently, PET studies have implicated these 'mirror neurons' in the gesture recognition system of humans. This system involves Broca's area, a language area in humans, which is generally believed to be the human homologue of area F5 in monkeys. Therefore, we explore the role of mirror neurons and cell assemblies for multimodal integration of action, vision, and language in the MirrorBot project.

Second, examining the processing of action verbs that relate to the leg, face and arm Pulvermüller et al. 2000 [13] found that cell assemblies are associated through semantic information with the appropriate body part. Furthermore, it was noted by Rizzolatti et al. 2001 [16] that when subjects were required to observe actions made by the mouth, hand and foot that the foot was represented dorsally and mouth and hand ventrally in the brain. This neurocognitive topological evidence motivates our approach for self-organising associative memory in multiple regions of the brain.

In our approach neural learning of multimodal association of motor actions, vision and language will be a key element for learning robots. We understand learning by demonstration and imitation in this general sense of learning multimodal internal topological representations. In an initial experiment and architecture described in this paper we show how representation of demonstrating motor actions and language instructions can be integrated. In a second step we will outline an architecture for integration of motor actions, vision and language representations.

II. ASSOCIATING MOTOR ACTIONS WITH ACTION VERBS

We have begun to associate internal representations of demonstrated actions with a word description. This system learns to associate the semantic features that are found in the sensor readings that represent the action with a representation of the word. As can be seen at the bottom of Fig 1 the architecture firstly contains a self-organising network to associate the action sensor readings with the appropriate body part by clustering the actions in different regions. At the next processing level there is a self-organising network for each body part that uses the sensor reading vectors to associate the actual action verbs with different regions. To the right in the architecture, the words that are represented using their phonemes are clustered in a self-organising network. The upper-most self-organising network associates the action representations by using the locations on the body part self-organising networks and their appropriate word form representation from the location on the word form self-organising network. Hence by associating the action representation with the word form the robot can describe the action with a word when it receives only the action representation and vice versa perform the action when it is given the word only.

In this system the input is used to produce the output by recreating the action from the sensor readings. The sensor readings provide information on the action such as the velocity of the separate wheels, the gripper activities and how the constituent sub-actions relate to the states of sensors such as break-beam and table sensors. If the robot receives the 'put' action sensor reading representation, it would be introduced into the trained body part network and activate the hand region of the output layer. The hand self-organising network would then position the sensor readings input in the 'put' region of the output layer. As the robot is describing the word form there is no necessary input from the word self-organising network into the association self-organising network. However, as the network has previously learnt to associate this action with the appropriate word form the 'put' region of the network is activated. The robot will then state using its language synthesis that the action semantic features provided are those for 'put'. On the association self-organising network, the winner-take-all mechanism removes ambiguities in the representation, allowing for only one action.

This describes the pathway from the internal action representation via the association area to the language description. It would be used to make the robot speak from observing actions. In a similar but opposite pathway the word input representation can lead via the association area - to the sensory robot action. This would be used to make the robot execute a verbal command.

This approach offers some brain-inspired regional modularity by having multiple self-organising networks each performing a subtask of the overall task. These networks are linked in a distributed overall memory organisation. Furthermore, this architecture includes components that are analogous to brain regions at a higher level. For instance, the SOMs that take the action representations and cluster these are related to the sensory motor cortical areas of the brain. The approach also takes into account the neurocognitive evidence of Pulvermüller (2003) [11] in that cell assemblies in different regions are associated with specific action verbs as a functional unit, with the association being based on the action verbs relationship with the appropriate body part.

This architecture links in some concepts of the mirror neuron theory. The relationship of mirror neurons to language was pointed out by Rizzolatti and Arbib 1998 [14] who found that neurons located in the F5 area of a primate's brain were activated by both the performance of the action and its observation. The recognition of motor actions comes from the presence of a goal and so the

motor system does not solely control movement [8]. The role of these mirror neurons is to associate action representations with vision or language representations. The mirror neuron system was a critical discovery as it shows the role played by the motor cortex in action depiction [16]. By using the sensor readings as input the mirror neuron concept is considered since the understanding of the action can come from either performing the action or a stored representation is linked to observing the action.

III. SELF-ORGANISATION ON THE ROBOT

In order to have greater objectivity and to incorporate self-organising maps into a robot control system, sensor readings were taken from the MIrror-neuron Robot Agent (MIRA) (see Fig 8). The MIRA robot is based on a Peoplebot and was set up to perform various actions that are associated in humans with the leg, head or hand. The leg verb actions were 'turn left', 'turn right', 'forward' and 'backward'; head action verbs were 'head up', 'head down', 'head right' and 'head left'; and finally the hand verbs were 'pick', 'put', 'lift', 'drop' and 'touch'. One action can be made of several basic actions. For instance, the hand verb action 'pick' included the following sub-actions (i) slowly move forward to the table; (ii) tilt camera downward to see table; (iii) lift gripper to table height; (iv) open gripper; (v) close gripper on object; (vi) stop forward motion; and (vii) lift gripper. The MIRA robot performing the 'pick' action is shown in Fig 8, top. This sequence of sub-actions corresponds in principle (although not in detail) to motor schemata since a complex action is represented as a sequence of basic actions. Sensor readings were taken for such sequences of basic actions.

In order to provide sufficient and varied training and test data the actions were performed 20 times (15 training and 5 test) under diverse conditions. For instance, the speed the robot was travelling at and the angle that the camera was tilted or panned to were varied. The sensor readings were taken 10 times a second while MIRA performed these actions including the state of the gripper, the velocity of the wheels and the angle that the robot's camera was at. The full list of the sensor readings is given in Table I.

To reduce the size of the input for the self-organising networks to a manageable level, 10 sets of the readings were taken over time to represent the action. This was achieved by taking the first, last and eight equi-distant sets of readings and combining them to create a single input for a sample. This procedure concatenates the whole time series to one data point and bypasses problems of short-term memory. We normalised the sensor readings for such variables as velocity of left wheel, velocity of right wheel, x coordinate of robot, y coordinate of robot, and the pan and tilt of the camera.

For the self-organising network to cluster actions based on the appropriate body part the input layer had 120 units,

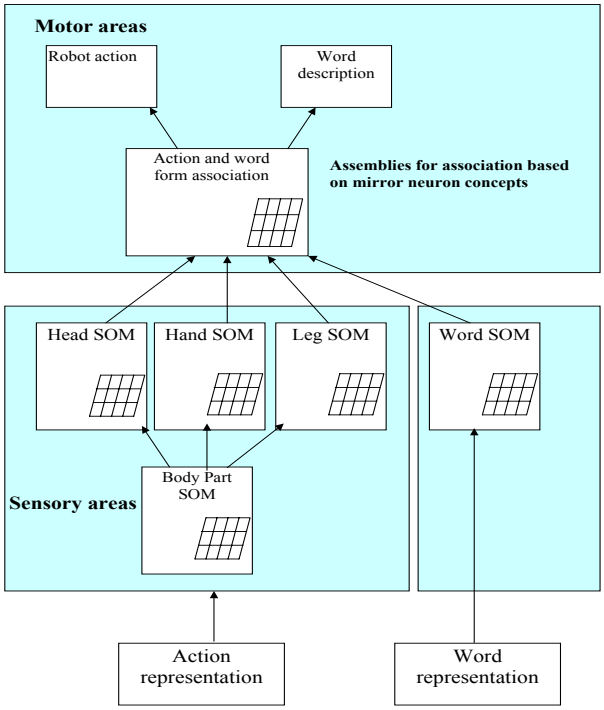


Fig. 1. The self-organising associative architecture.

one for each of the preprocessed sensor readings. The output layers had various sizes (from 8 by 8 units to 13 by 13 units) and the networks were trained between 50 to 500 epochs at intervals of 50 epochs. There were 260 samples in total, 195 for training and 65 for testing. The location of each of the training and test samples on the self-organising maps were identified based on the units that had the highest activation.

Fig 2 shows a self-organising network that was 12 by 12 units. Once this network architecture was trained for 50 epochs there was clear clustering into the three body parts (see Fig 2). The hand action words such as 'pick', 'touch', 'lift' were at the bottom of the training and test output layers in the hand body part region, with the head actions slightly below and to the right of the leg region. Although one unit within the head region contained both head and leg action samples with the highest activation, the percentage for head samples was much higher on both test and training data. For the training and test data the percentage of head action samples with the highest activation for that unit was over 60% for training samples and 70% for test samples. Due to the major difference between the head and leg action percentages for this unit, only the head percentage is shown on Fig 2.

For the training data 100% of the samples which correspond to the head and hand actions fell in the appropriate region and 88% of the leg data. For test data the percentage

TABLE I

SENSOR READINGS TAKEN BY ROBOT DURING ACTIONS.

Sensor Reading	Value
Left Wheel Velocity	Real number
Right Wheel Velocity	Real number
x coordinate of robot	Real number
y coordinate of robot	Real number
Break-beam state of gripper	No beam broken, Inner broken, Outer broken, Both broken
Gripper state	Fully open, closed, between open and closed
Gripper at highest or lowest position	Yes No
Gripper moving	Yes No
Table sensors activated	Yes No
Gripper opening	Yes No
Pan of camera	Integer
Tilt of camera	Integer

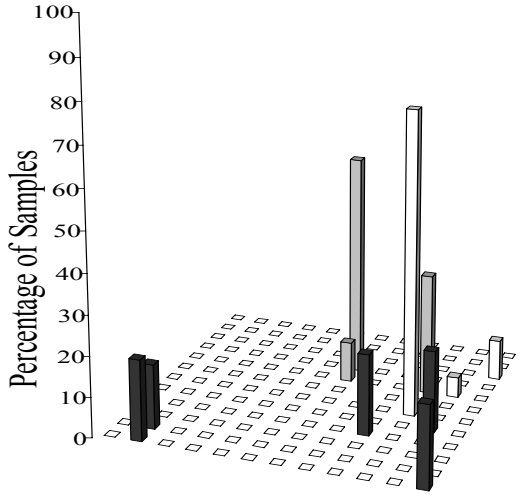


Fig. 2. The percentages for the test samples for the body parts that have highest activation for each unit on a 12 by 12 units network after a training time of 50 epochs. (Black - Hand, White - Head, Grey - Leg)

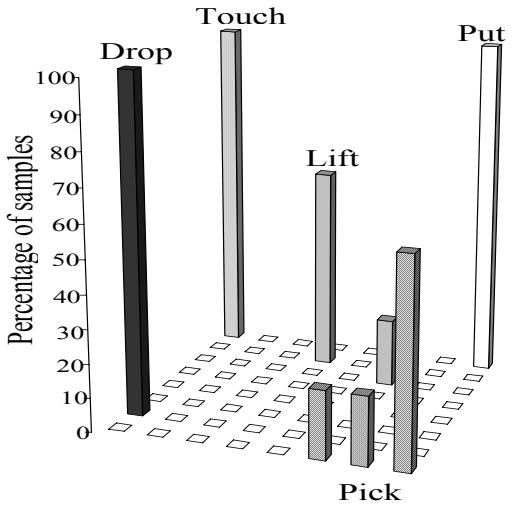


Fig. 3. The percentage for the test samples for the specific hand actions that have highest activation for each unit on a 8 by 8 units network.

was even better with 100% for hand and head and 90% for leg. It is interesting to note that within the hand verb region there was a good division into the actual action classes. In Fig 3 'pick' was located in the lower right of the map, 'put' in the lower left, 'drop' in the unit above 'pick', 'touch' at the top of the hand region and most of the 'lift' samples were located in a unit just below 'touch'. For the other two classes there was some splitting into the individual actions but not on the scale of the hand class (see Fig 4 and Fig 5).

For such an architecture on both training and test data the clusters were in very similar positions on the output layer, which points to the ability of the network to generalise on data it has not seen before. When considering the percentage of test data that fell in the regions identified by the training data the percentages were very high. For the hand actions 100%, head actions 95% and leg actions 88% of the test data fell into the appropriate training region. Therefore, if the self-organising network was used in the control of a robot it can perform successfully in an on-line manner clustering semantic features of the action to the appropriate region of the output layer.

Turning to the hand, head and leg self-organising networks, when considering the clustering of the specific body part actions for all three types of action, the size of network that performed best was 8 by 8. For the hand network the training time that produced the best clustering was 50 epochs, for the head network it was 150 epochs and for the leg self-organising network it was 100 epochs. As can be seen from Fig 3 to Fig 5 there was clear clustering into different regions for the hand, head and leg actions.

IV. ASSOCIATING VISION AND MOTOR REPRESENTATIONS

Our next step is to describe an associator neural network to localise a recognised object within the visual field. This is an essential basic skill for robotic learning by demonstration which we solve by a purely neuronal approach. The model, depicted in Fig 6, is thus a centrepiece of a larger model which can on the one hand perform actions like grasping and on the other hand is connected to neurally implemented language areas.

The idea extends the use of lateral associator connections within a single cortical area to their use between different areas [18]. The first cortical area is the visual area V1 which codes for an internal representation, "what", of images. The weights connecting it to the image are trained

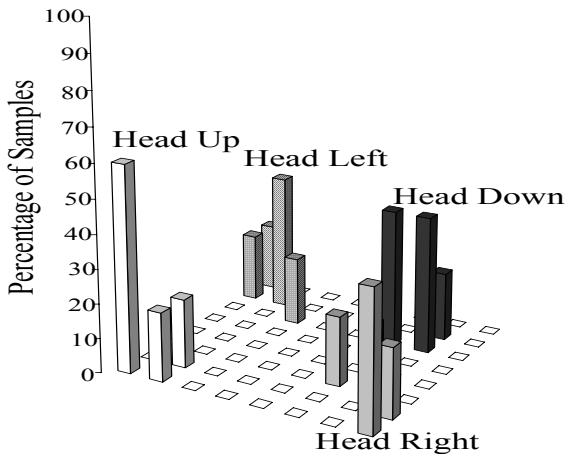


Fig. 4. The percentage for the test samples for the specific head actions that have highest activation for each unit on a 8 by 8 units network.

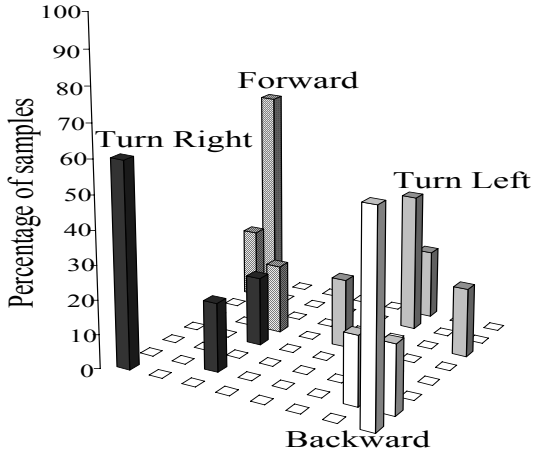


Fig. 5. The percentage for the test samples for the specific leg actions that have highest activation for each unit on a 8 by 8 units network.

by a sparse coding Helmholtz machine. Earlier, intra-area lateral connections have been implemented within V1 to endow the simple cells with biologically realistic orientation tuning curves as well as to generate complex cell properties. We extend the lateral connections to also span a second cortical area, the ”where” area which is laterally connected to the simulated V1. The lateral weights are trained to associate the V1 representation of the image to the location of an object of interest which is given on the ”where” area. The lateral weights are thus object specific associative weights which can complete a representation of an image with the location of the object of interest.

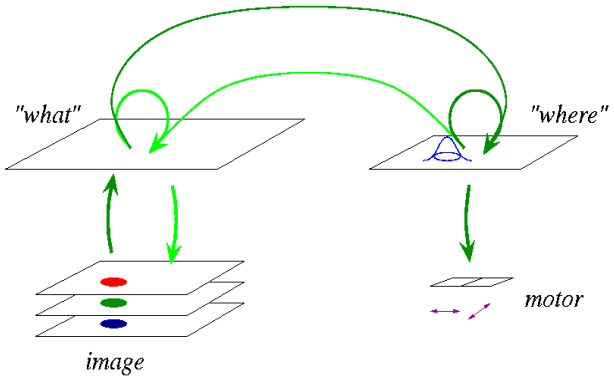


Fig. 6. Model architecture. The hidden representation ”what” of the image including the target object is associated to the location ”where” of the target which is relevant for motor action.

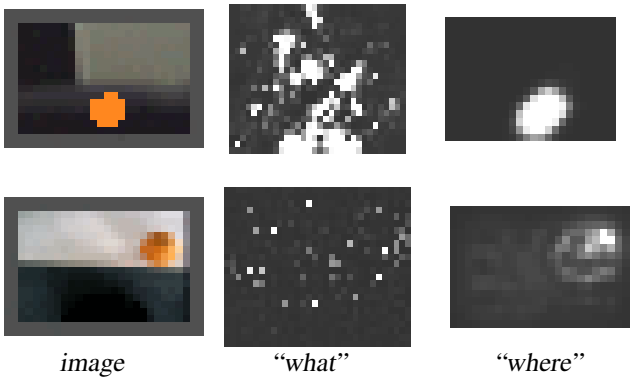


Fig. 7. Example representations on the image, ”what” and ”where” areas. The *image* is originally in color, where in the upper row, the orange fruit target is artificially generated. The networks of the upper and lower row were also trained and activated with different parameters.

Fig 7, shows the network activities after initialisation with sample stimuli of an orange and relaxation to a steady state. In both cases the ”where” area neuron’s activations were initialised to zero initially (not shown). The relaxation procedure which spans the ”what” and the ”where” area then completes the pattern by displaying the location of the object of interest as a Gaussian activity hill.

Once that an object of interest appears in the visual field, it is first necessary to localise its position within the visual field. Then, usually the centre of sight is moved toward it, and a grasping movement prototype will be activated which is related to the specific affordance [15].

We have made initial experiments connecting the ”where” area to motor neuron’s output which control the robot camera’s pan-tilt motors. The task is to move the camera so that the orange fruit is located in the centre of the “where” area (Figs. 6,7). This is achieved by a simple algorithm. Weights from every unit of the “where” area to the camera’s pan and tilt units were trained based on the

error of a movement: if after a tilt movement the camera would face, e.g., too much upward, then the unit which elicited that movement had its weight to the tilt motor unit changed, so that at the next trial it would face a little less upward. Fig 8, bottom, shows the camera pointing toward an orange which is moved across its ”visual field”. This implements the MIRA robot’s reaction to the command ”Bot show orange”.

Additionally, using reinforcement learning, we have very recently implemented the robot ”docking” at a table so that it can grasp an object which lies at the border of the table with its short grippers [19]. The input to the reinforcement-trained network is the perceived target location (from the “where” area) and the rotation angle of the robot w.r.t. the table. Outputs are the four motor units and a critic unit which carries a value function on the input space. A positive reinforcement signal is given if two conditions are met: *(i)* the target is perceived at the middle of the lower edge of the visual field (where also the gripper is perceived by the camera which is at a fixed position) and *(ii)* the rotation angle is zero (which is defined such that the robot is approaching the table perpendicularly). The weights to the value function unit and those to the motor units develop concurrently such that an optimal strategy toward reaching the target will by performed. Fig 8, top, shows the robot perpendicularly at the table, at the goal position. The data delivered during these actions will be used for the training and verification of mirror neurons.

Fig. 8. The MIRA robot performing the ’pick’ action (top) and recognising and tracking an orange with its pan-tilt camera (bottom).

V. ASSOCIATING VISION, LANGUAGE AND MOTOR REPRESENTATIONS FOR LEARNING BY DEMONSTRATION

As the next steps therefore the model needs to be extended to incorporate more complex motor tasks. This is not only desirable from a robotic application point of view, but also from the fact that mirror neurons are action-related, as they reside in motor associated cortical areas such as F5 and respond to performance, description and observation of actions (Rizzolatti and Arbib 1998) [14]. For this, we will integrate the language- and motor-sensory related ’pick’ action with the more vision relate the more vision related tracking action (Fig 8).

The model of Elshaw and Wermter (2002) [6] and Elshaw, Wermter and Watt (2003) [7] handles an organisation of a variety of actions on a self-organising layer of neurons as an avenue to include a larger number of motor tasks. Fig. 9 shows the plan of a proposed network.

In the envisaged network of Fig 9, mirror neuron properties are expected to evolve among some of the neurons in the top layer. They carry an internal representation $\vec{r}$ of all of the inputs, below. The inputs are from multiple modalities including higher level representations. The vector $\vec{l}$ contains language input information. This can include internal representations from language areas or the goal area of the cell assembly model for Broca/Wernicke areas. $\vec{pv}$ contains the visual perception which includes the identity and perceived location of a target to be grasped. $\vec{m}$ are the motor unit activations including wheels, gripper and pan-tilt camera. $\vec{ms}$ denotes motor sensory unit activations and may also include available idiothetic information such as the rotation angle of the robot. $\vec{i}$ are other internal states such as the goal related value function of the critic used in reinforcement learning.

Thick lines with arrow heads denote the weights. The vertical connections are trained with a sparse coding unsupervised learning scheme similar to the Helmholtz machine which we described for image processing (Fig 6). The inputs are collected from real robotic actions (after exercising with simulated data) which are performed interactively in the environment. The data contain only instantaneous information, i.e. the whole action sequence is not known. Therefore, neurons do not necessarily fire over a sustained period in time as do mirror neurons. However, since $\vec{r}$ is a distributed code, some of the units may specialise to code for longer sequences. The horizontal

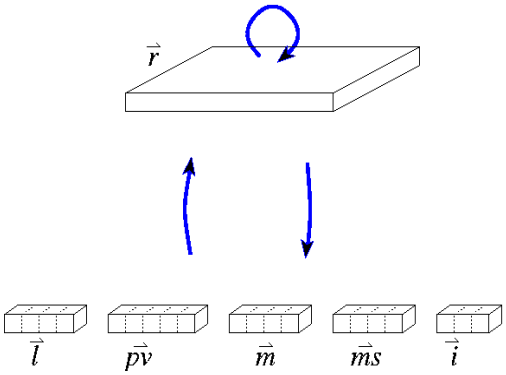


Fig. 9. The envisaged associative architecture

recurrent connections (depicted as open circle) are trained as an autoassociator neural network. They are used in a neural activation relaxation procedure which de-noises the representation $\vec{r}$ and may also encourage prolonged firing. As a possible extension, associator recurrent connections may also feed back to the input. This would be particularly interesting for the cortical feed back to the motor units, because of implications for motor control.

VI. DISCUSSION

We have developed biologically inspired solutions for tasks which are needed by a robot that should learn by demonstration and instruction.

The robot sensor inputs to the modular, self-organising network were partitioned in a way that they match the three body areas 'leg', 'head' and 'hand'. The match is intuitive, but equivalents of the robotic sensor readings (like "gripper opening") are likely to be represented at various locations on the cortex, as a visual or motor-sensory perception or distributed in the language system as a "word web" [11]. The network can in principle realise the findings of Pulvermüller et al. (2000) [13] on the processing of action verbs with different clusters representing the specific body parts. The network was able to identify the semantic features from the actual sensor readings for the individual action verb classes that were specific to the appropriate body part. These features were likely to include the degree of move, whether there was an object involved and the type and number of motors used.

The performance of the head, leg and hand self-organising networks are in principle suitable for use in a robot control system based on language instruction. This is because it is likely, based on the clear clustering demonstrated, that the sensor reading input will be accurately represented and mapped to the appropriate network region. As this location is the basis for the association between the action and the word this will contribute to the successful identification of the action and its description.

A recurrent associator network with distributed coding was applied to the visually related part of the task. Such associator networks form the neural basis for multimodal convergence and at the same time can supply a distributed representation across modalities as has been proposed for linguistic structures [12]. Multimodal representations furthermore allow for mirror neuron-like response properties which shall emerge in our application within a biomimetic mirror neuron-based robot, MirrorBot.

Two actions, interactively performed with the environment, shall supply input data to the envisaged mirror neurons. Since reinforcement learning which we used to train these actions is attributed to the basal ganglia, the model extends beyond the cerebral cortex, in a biologically plausible fashion.

VII. CONCLUSIONS

We have described some research toward integrating learning by demonstration and learning by instruction on a neural substrate on a robot. Our approach is not so much on imitating complex behaviour. Rather our focus is on testing mirror neuron concepts and other neurocognitive evidence like the topological arrangement of actions in order to provide a multimodal integration of the robots own actions, as well as visual observation and language instruction. We think that visual observation and language instructions are complementary forms of programming robots in a natural manner to perform and link their performance to their own underlying actions. An associative neural organisation of the internal memory may therefore be advantageous for a robot's learning of visually described actions or verbally instructed actions.

VIII. ACKNOWLEDGEMENTS

This work is part of the MirrorBot project supported by the EU in the FET-IST programme under grant IST-2001-35282.

IX. REFERENCES

[1] H. Asoh, S. Huyamizu, H. Isao, Y. Motomura, S. Akaho, and T. Matsu, "Socially embedded learning of office-conversant robot jijo-2", in *Proceedings of 1997 International Joint Conference on Artificial Intelligence, Nagoa*.

[2] A. Billard, "Imitation: a means to enhance learning of a synthetic proto-language in an autonomous robot", in Dautenhahn, K. and Nehaniv, C. (eds), *Imitation in Animals and Artifacts, Academic Press*, pp. 281-311, 2001.

[3] C. Breazeal and B. Scassellati, "A context-dependent attention system for a social robot", in *Proceedings of the 1999 Sixteenth International Joint Conference on Artificial Intelligence (IJCAI99), Stockholm, Sweden*, pp. 1146-1151.

[4] W. Burgard, A.B. Cremers, D. Fox, D. Hähnel, G. Lakemeye, D. Schulz, W. Steiner, and S. Thrun, ”Experiences with an interactive museum tour-guide robot”, *Artificial Intelligence*, Vol. 114, No. 1-2, 2000.

[5] Y. Demiris and G. Hayes, ”Imitation as a dual-route process featuring prediction and learning components: A biologically plausible computational model”, in Dautenhahn, K. and Nehaniv, C. (eds), *Imitation in Animals and Artifacts*, MIT Press, 2002.

[6] M. Elshaw and S. Wermter, ”A neurocognitive approach to self-organisation of verb actions”, in *Proceedings of 2002 International Joint Conference on Neural Networks, Honolulu, USA*, pp. 24-29.

[7] M. Elshaw, S. Wermter and P. Watt, ”Self-organisation of language instruction for robot action”, in *Proceedings of 2003 International Joint Conference on Neural Networks, Oregon, USA*.

[8] V. Gallese and A. Goldman, ”Mirror neurons and the simulation theory of mind-reading”, *Trends in Cognitive Science*, Vol. 2, No. 12, 1998, pp. 493-501.

[9] P. Gaussier, S. Moga, J.P. Banquet and J. Nadel, ”Learning and communication in imitation: An autonomous robot perspective”, *IEEE Transaction on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 31, No. 5, 2001, pp. 431-444.

[10] G. Maistros and G. Hayes, ”An imitation mechanism for goal-directed actions”, in *Proceedings of TIMR 2001 - Towards Intelligent Mobile Robots, Manchester*.

[11] F. Pulvermüller, *The neuroscience of language: On brain circuits of words and serial order*, Cambridge University Press, 2003.

[12] F. Pulvermüller, ”Words in the brain’s language”, *Behavioral and Brain Sciences*, Vol. 22, No. 2, 1999, pp. 253-336.

[13] F. Pulvermüller, M. Hare and F. Hummel, ”Neurophysiological distinction of verb categories”, *Cognitive Neuroscience*, Vol. 11, No. 12, 2000, pp. 2789-2793.

[14] G. Rizzolatti and M. Arbib ”Language within our grasp”, *Trends in Neuroscience*, Vol. 21, No. 5, 1998, pp. 188-194.

[15] G. Rizzolatti and G. Luppino, ”Cortical motor system”, *Neuron*, Vol. 31, 2001, pp. 889-901.

[16] G. Rizzolatti, L. Fogassi and V. Gallese, ”Neurophysiological mechanisms underlying the understanding and imitation of action”, *Nature Review*, Vol. 2, 2001, pp. 661-670.

[17] S. Thrun, M. Bennewitz, W. Burgard, F. Dellaert, D. Fox, D. Haehnel, C. Rosenberg, N. Roy, J. Schulte and D. Schulz, ”MINERVA: A second generation mobile tour-guide robot”, in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA’99)*.

[18] C. Weber and S. Wermter, ”Object localisation using laterally connected ”What” and ”Where” associator networks”, in *Proceedings of the 2003 International Conference on Artificial Neural Networks, Istanbul, Turkey*, pp. 813-820.

[19] C. Weber and S. Wermter, ”Robot Docking with Neural Vision and Reinforcement”, in *Proceedings of the AI-2003, Twenty-third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK*, (accepted).

[20] S. Wermter, J. Austin, D. Willshaw and M. Elshaw, ”Towards novel neuroscience-inspired computing”, in S. Wermter, J. Austin, and D. Willshaw, (Eds.), *Emergent Neural Computational Architectures based on Neuroscience*, Heidelberg, Germany: Springer-Verlag, 2001, pp. 1-19.

[21] S. Wermter and M. Elshaw, ”Learning robot actions based on self-organising language memory”, *Neural Networks*, Vol. 16 pp. 5-6, 2003, pp. 661-669.

[22] S. Wermter and C. Panchev, ”Hybrid preference machines based on inspiration from neuroscience”, *Cognitive Systems Research*, Vol. 3, No. 2, 2002, pp. 255-270.

Learning From Observation and Practice Using Primitives

Darrin C. Bentivegna (darrin@atr.co.jp)[1,2], Christopher G. Atkeson (cga@cmu.edu)[1,3],
Gordon Cheng (gordon@atr.co.jp)[1]

[1] ATR Computational Neuroscience Laboratories,
Department of Humanoid Robotics and Computational Neuroscience, Kyoto, Japan
[2] Georgia Institute of Technology, College of Computing, Atlanta, GA
[3] Carnegie Mellon University, Robotics Institute, Pittsburgh, PA, USA

Abstract

This paper focuses on learning from observation using primitives and improving performance of the task through practice. We have created a flexible framework that provides structure for this type of research. A novel algorithm that combines Q-learning and a locally weighted learning method to improve primitive selection and sub-goal generation has been created. We demonstrate this approach applied to the tilt maze task. Our robot initially learns to perform this task using learning from observation, and then increases performance through practice.

1 Introduction

Behavioral primitives are defined as solutions to small parts of a task that are combined to complete a task [1, 11]. We are exploring whether learning in terms of primitives speeds up learning of dynamic tasks. A task that is to be performed using primitives must first have a library of primitives to use. Research is being performed on having a robot discover primitives automatically after observing performances of a task [6] or while operating in a task environment [9]. We will use a manually defined library of primitives in this work, so we can focus on learning to *select* primitives and *generate* sub-goals.

A software and hardware version of the tilt maze task, Figure 1, have been created as testbeds in which to test our primitive learning framework and learning techniques. In this task, a player tilts a maze to roll a marble to a goal, avoiding hazards such as holes. The manually defined library of primitives is (Figure 2):

- **Roll To Corner**: The marble rolls along a wall and stops in a corner.
- **Roll Off Wall**: The ball marble rolls along a wall and then rolls off the end.
- **Guide**: The marble rolls without touching a wall.

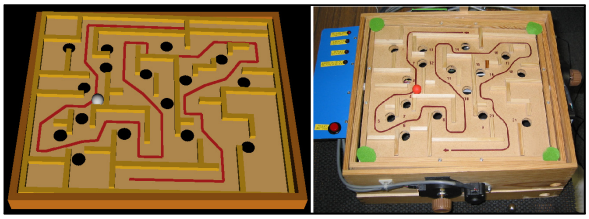


Figure 1: Software and hardware Marble Maze environments.

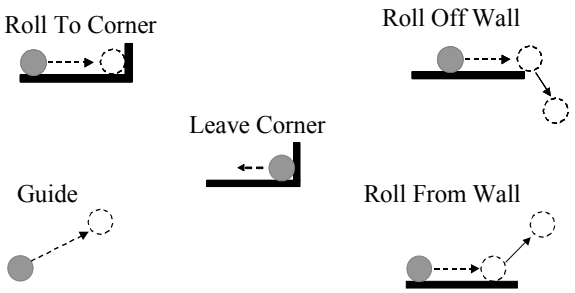


Figure 2: Primitives being explored in the Tilt Maze Environment.

- **Roll From Wall**: The marble rolls on a wall and then rolls away from it.
- **Leave Corner**: The marble is captured in a corner and then the board is positioned in preparation to move the marble from the corner location.

In both hardware and simulation versions of the tilt maze task the board and marble positions can be recorded as a human plays the game. The hardware maze has two motors that control the tilt of the board. A human can control the motors using knobs that are connected to encoders. The computer receives the encoder signals and creates the proper motor commands. The board orientation is also measured using two encoders. The computer is equipped with a vision system which estimates the position of the marble on the board

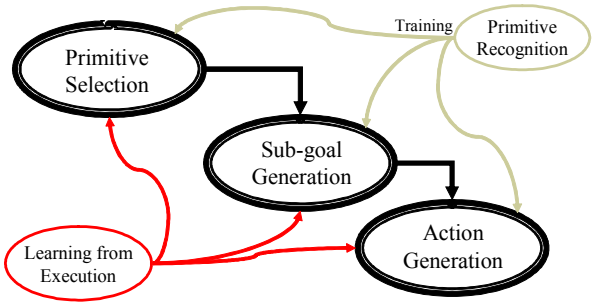


Figure 3: Our framework.

at 60Hz [5]. In the software version the human controls the board using a mouse. Noise is introduced into marble velocity calculation in the simulator to make the outcome of actions less deterministic.

2 Our Framework

Our behavioral framework has three main parts when operating in the environment using observed information (Figure 3) [3]. The first part is a classifier that uses the current location in state space to choose a behavioral primitive to execute. For tasks that have a set sequence of actions, such as dancing, primitive selection can be done by specifying the sequence of primitives to be performed [8]. Tung and Kak [13] show how a planning system can be used to specify the primitive execution sequence in an environment where objects are not moving during training and there is a high probability of successful performance of the primitives. Robots operating in a similar static environment have learned a primitive execution sequence from observed data [7]. We are interested in dynamic tasks where primitive execution sometimes fails, other agents interfere, and a fixed sequence of primitives is not adequate.

The second part of our framework is a module that specifies parameters for the behavioral primitive chosen, for example, how fast to go, or how much to turn. Often, these parameters can be interpreted as behavioral sub-goals. For the **Roll Off Wall** primitive, for example, this module specifies the velocity at which the marble will roll off the wall and the tilt of the board at the end of primitive execution.

The third part of our framework is a module that specifies the actuator commands to achieve a behavioral sub-goal, or behavioral target specified by the current primitive and parameters. There is a separate action generation module for each type of primitive. For the tilt maze task, models are created from observing the human that encode the actions the human takes during the performance of the primitive to move the environment from the current state to the sub-goal state. The models provide a policy that is used by the agent to control the tilt of the board to perform the selected primitive type to obtain the desired sub-goal.

The modules described above obtain training data from observing the task being performed by a human. The primitive recognition module segments the data collected from observing the task into the predefined primitive types. This segmented information is then structured as needed for the other modules in the system.

In learning from observation, the robot's goal is to behave like the teacher. No knowledge of higher level goals or how to improve its performance autonomously is needed. In order to learn from practice, there must be additional domain knowledge that provides information needed to evaluate progress towards task objectives. The learning from execution module contains that knowledge and provides feedback to the other modules so their behavior can be changed.

3 Obtaining Information from Observing Others

The observed data is continuous and is segmented into primitives by the primitive recognition module using critical events. Critical events are easily observable occurrences such as marble-wall contacts and the marble traveling along a wall. The data collected in the simulator includes the status of ball-wall contacts. This status provides information on the side of the wall (top, bottom, left, right) that the marble is currently in contact with. The primitive definition is used to create a sequence of critical events that allow the primitive to be observed in the data. The **Roll to Corner** primitive, for example, begins just before the ball makes contact with a wall that contains a corner and then rolls along that wall into the corner. An easy way to find an occurrence of this primitive is to look at each observed data point and find one where the status indicates that the marble is in contact with two connecting walls. This is the location of the end of the primitive and the state of the environment, S_e, is recorded for this point.

From this ending data point, the primitive recognition algorithm backs up through the data points and notes which wall the marble is rolling along. The algorithm continues to back up through the data searching for the first occurrence of when the marble is no longer on the wall. This point is the start of the primitive and the environment state, S_s, is recorded.

The parameters recorded for all environment states

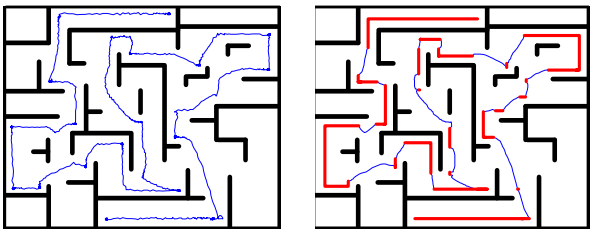


Figure 4: Left: Path of the marble while observing a human teacher. Right: The processed path with the thicker line (red) representing situations in which the marble is in contact with a wall.

in the tilt maze domain are the marble's position (x, y), velocity (Vx, Vy) and board tilt angles $(\theta x, \theta y)$. Using this obtained information a primitive data point can be created that represents an action, PT (one of the specified primitive types), taken by the human while they were operating in the environment. The goal of performing this primitive can be determined from S_e, the state of the environment at the completion of the primitive performance. This data point encodes what occurred during the observed performance in the following way: When the environment was in the state S_s, the human performed the primitive PT, and at the completion of that action the environment was in the state S_e.

The data collected from the hardware version of the tilt maze is noisy and does not include the wall contact status and velocity information provided by the simulator. To reduce the noise in the data it is filtered forward and backward through a Butterworth filter with a cutoff of $12Hz$. The velocities are then computed using the frame rate of $60Hz$ and the wall contacts are inferred from the observed ball position and board orientation. Figure 4 shows the raw collected data, left, of an observed game played by a human teacher. The right side of Figure 4 shows the processed data with a thicker line (red) showing where appropriate ball-wall contact status flags, described above, were set.

After the observed hardware data is processed and the appropriate status flags are set, it is applied to the recognition algorithms. The primitives recognized in the processed data are shown in Figure 5. The appropriate primitive symbol shows the beginning position of each recognized primitive. The line represents the path the marble took while that primitive was being performed. The primitive ends when the next primitive symbol is reached or the line ends. No action was classified in the areas where there are gaps between the end of one primitive and the start of another.

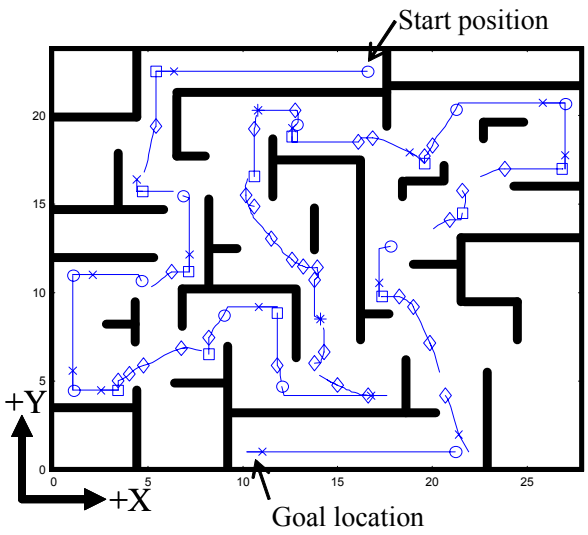


Figure 5: The primitives recognized while observing the human during one trial. The symbols show the start of the primitive as follows: ◯-**Roll To Corner,** □-**Roll Off Wall**, ◊-**Guide**, ∗-**Roll From Wall**, ×-**Leave Corner**.

4 Choosing Primitives and Associated Parameters

It is the responsibility of the primitive selection module, Figure 3, to choose the primitive type, based on the current state and prior observations of primitives that have been executed. In our implementation, during training the context or state in which the human has performed each primitive is extracted from the observed data, and during execution is used by a nearest neighbor lookup process to find the most appropriate primitive type as follows.

A database is created from the observed data that contains states of the environment and corresponding primitive types. A lookup is performed on this database to find the states that are closest to the query state. The distance of each data point from the query point is computed as $d(\boldsymbol{x}, \boldsymbol{q}) = \sqrt{\sum_j w_j \cdot (\boldsymbol{x}_j - \boldsymbol{q}_j)^2}$, where $\boldsymbol{x}$ and $\boldsymbol{q}$ are the locations of the data point and the query point in state space, and w allows each dimension to be weighted differently. A query to the database is the current state of the environment: marble position (x, y), velocity (Vx, Vy) and board tilt angles $(\theta x, \theta y)$.

A pure nearest neighbor lookup scheme would use the closest point to select the primitive type. The data point also contains the observed outcome of the human's performance of that primitive, which can be used

as the desired sub-goal and used to compute the parameters needed to perform that primitive type.

A more robust approach is to use several nearby points, and implement some sort of voting scheme. Multiple data points in the vicinity of the current state are retrieved from the database. The primitive type (a discrete choice) can be chosen by selecting the primitive type that occurs most often within the closest N data points, for example. We are currently selecting the primitive type indicated by the nearest data point.

4.1 Computing the Desired Sub-goal

Once the primitive type has been chosen the sub-goal can be computed. It is important to first choose the primitive type because the sub-goals of different primitive types can not be combined. For example it would not make sense to use the sub-goal of the **Roll Off Wall** primitive with the **Roll Into Corner** primitive. The **Roll Into Corner** primitive will be expecting a corner for the marble to land in as a sub-goal location and the **Roll Off Wall** primitive will be specifying a sub-goal location at the end of a wall.

The n closest points of the same primitive type are used to compute the sub-goal using a locally weighted learning (LWL) model[2]. n has been chosen as 5. A kernel function, $K(d) = \exp^{-\alpha d^2}$, uses the distance to compute the weight of each data point. [2] discusses the effect of other kernel functions on the weighting of the data points. The output components needed at the query point are computed using the equation $y(\boldsymbol{q}) = \frac{\sum y_i K(d(\boldsymbol{x}_i, \boldsymbol{q}))}{\sum K(d(\boldsymbol{x}_i, \boldsymbol{q}))}$ where i ranges from 1 to n. The values of the vector w and α were set globally and were chosen by trial and error.

4.2 Results of Learning from Observation

Our learning from observation using primitives framework has been used to create agents that operate in the hardware and software tilt maze environments. These agents followed four basic steps: 1. Observe the state of the environment; 2. Decide what primitive to perform; 3. Compute the parameters to use with the selected primitive type. 4. Perform the primitive until it has terminated. Steps 2 through 4 use the information obtained from observing the human. The agent's intention is to act as the human did, or as the human would, for the observed state. But if the agent incorrectly predicts the human's action, or can not correctly perform the chosen action, it has no way of knowing if the outcome is desirable for completing the task. A primitive

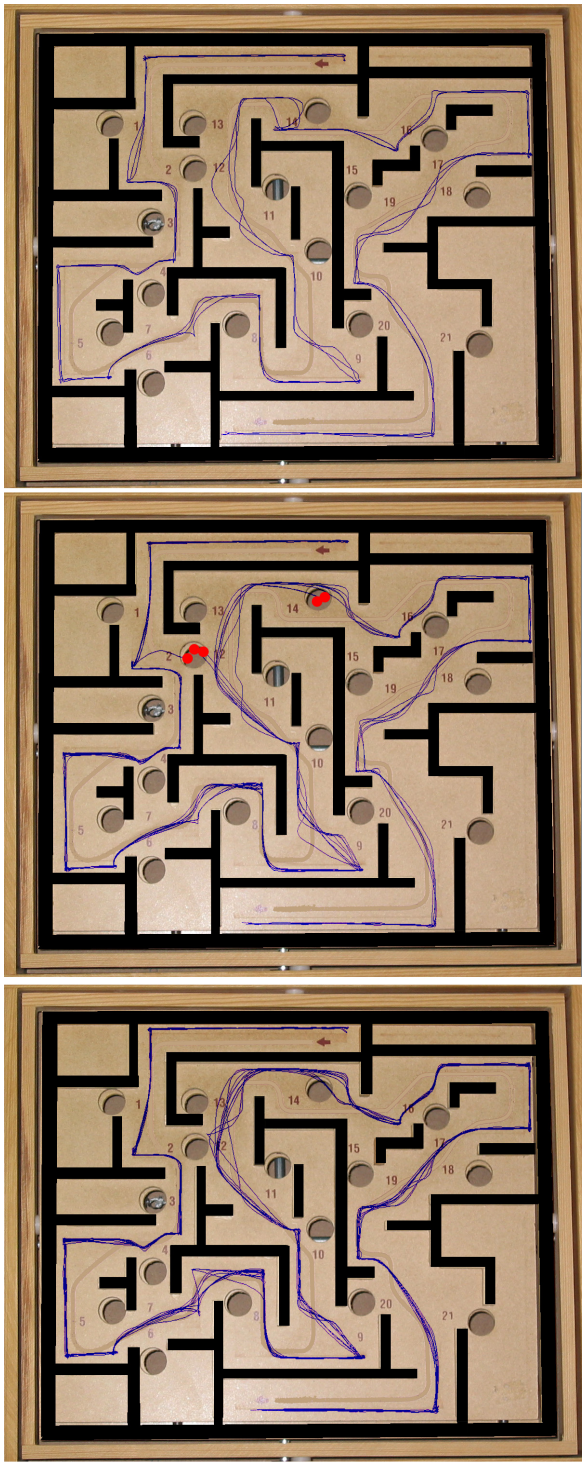


Figure 6: Marble paths during hardware maze learning: Top: The 3 training games. Middle: Performance on 10 games based on learning from observing the 3 training games. The marble falls into holes 2, 12, and 14. Bottom: Performance on 10 games based on learning from practice after 30 practice games.

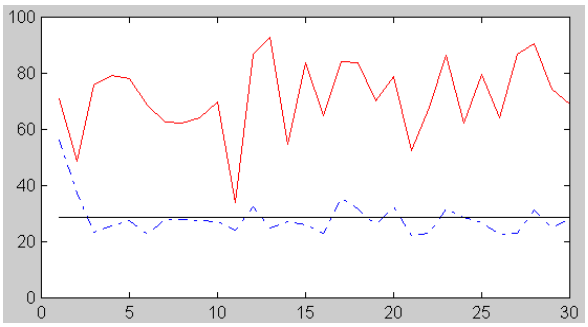


Figure 7: Performance on the software marble maze. Top solid (red) line: Agent using only observed information. Dashed (blue) line: Agent also learning while practicing. Bottom solid (black) line: Average time of the three observed games performed by the human.

ends when the marble reaches the chosen sub-goal location, rolls outside a bounding box containing the start and end position, rolls into a hole, or does not make progress in a timely manner.

Figure 6 shows the performance of the hardware agent. The top picture shows the path of the marble during the three observed games. The middle picture shows the ten paths the agent took while using this information to control the marble through the maze from the start position (the middle of the top of the board). The marble falls into holes 2, 12, and 14 five times and completes the maze five times.

An agent was also tested in the software environment. In the software environment play continues, even after failures, until the end location is reached. If a failure occurs the marble is re-placed on the board ahead of the failure location and the player is given a 10 second penalty on the time that is taken to traverse the maze. The agent can fail because the marble has fallen into a hole or does not make progress toward the goal within 15 seconds. This agent observed the human perform the task three times in the simulator. The human completed the maze in 23.3, 30.3, and 32.7 seconds, never fell into a hole and was never penalized for not making progress. The top solid (red) line on the graph in Figure 7 shows the performance of the agent during 30 trials in the simulation environment using the observed information. Each trial consist of playing three consecutive games and the graph shows the running average of the time to complete the three games. The skilled human's average of 28.77 seconds is shown by the bottom solid (black) line in the graph.

These results show the agents quickly learned how to perform in many parts of the maze from only observing the human and have approximately the same performance from one trial to the next. Differences between trials are mostly due to the noise that is inherent in, or introduced into, the environment. The observed information provides the agent with a good indication of what is the right thing to do for a given state, but it does not have the ability to change its policy from one trial to the next. A common error made by the agent is choosing a primitive that it can not perform from the current state of the environment.

These results also show that the agent's performance does not match that of the human and that the agents can perform better if they had the ability to change their policy while playing the game. To change its policy, the agent must have knowledge of the overall task objectives and some way to evaluate its performance toward the accomplishment of those objectives. Using this information it must also have a way to change its behavior as it practices to increase its performance toward completing the task. The remainder of this paper describes a method in which the agent can learn, through practice, to become more skilled at choosing a primitive and sub-goals for an observed state.

5 Learning to Select Experiences

As described above, the primitive to be performed and the desired sub-goal is determined by the data points selected. The results of performing a primitive can be observed and evaluated at its completion. If the chosen primitive and sub-goal result in the agent not making progress in the task, such as the marble falling into a hole, there must be some way to indicate this so that the same action is not performed again in the future from this same state.

To obtain this functionality, the LWL algorithm has been combined with a Q-learning algorithm [12] to give the agent an indication of the value of using data points from an observed state. The basics of the algorithm are to incorporate a multiplier into the distance function $d(\mathbf{x}_i, \mathbf{q})$ in the kernel regression equation in section 4. The multiplier has the effect of moving the data point in relation to the query point. A multiplier greater than 1.0 will have the effect of moving the data point further away from the query point and a multiplier less than 1.0 have the effect of moving the data point closer to the query point. For example, if the marble falls into a hole after a selected primitive is performed, the multipliers associated with the set of data points that were used to decide on that primitive can be increased. The next time the agent finds itself in the same state, those data points will appear further away and will therefore have less or no effect on the new chosen action.

5.1 Associating Multipliers With Data Points

There is a problem with this naive approach of only associating one multiplier to each data point. For one query point, the chosen data points may be inappropriate. But from a different query point, these data points may work very well. If the same multiplier is used at both query points, the system will not be able to distinguish between these two situations. From the first query point the agent may choose a more desirable action, but from the second query point the agent may choose a less desirable action. To overcome this limitation our algorithm provides the ability to use a different multiplier for different query points around the data point. Obviously there cannot be an infinite number of multipliers so the state space is quantized around a small area of the state space in the vicinity of the data point. In the tilt maze environment, for example, the state space has six dimensions. Each dimension is quantized into five cells. Therefore each data point in the database has a table of size 5^6. For any query point in the state space, its position relative to the data point is used to find the cell that is associated with that query point.

Only a small fraction of the cells are visited. Therefore the tables are stored as sparse arrays and only when the value in a cell is initially updated is the cell actually created. For example, if a set of chosen data points provides a good result, those data points will be chosen every time for that environment state. Other data points in the area will not be visited and therefore all the cells that are associated with those data points will not be created.

5.2 Using the Multipliers When Selecting Primitives and Sub-goals

In the tilt maze implementation the numbers in the cells encode the value of using that data point in relation to the query point. When selecting the nearest data points from the data base, the following equation is now used to provide the distance between the data point $\mathbf{x}_i$ and the query point $\mathbf{q}$: $\hat{d} = d(\mathbf{x}_i, \mathbf{q}) \cdot f(\mathbf{x}_i, \mathbf{q})$. The function $f(\mathbf{x}_i, \mathbf{q})$ finds the number in the cell of the data point $\mathbf{x}_i$ associated with this query point $\mathbf{q}$ and uses that number to compute a multiplier.

The multiplier returned by the function $f(\mathbf{x}_i, \mathbf{q})$ has a direct impact on the apparent distance of the data point in relation to the query point. Section 4.1 shows how multiple data points of the same type are combined to compute the parameters. The new distance $\hat{d}$ is now used to compute the parameters in place of $d(\mathbf{x}_i, \mathbf{q})$ in the LWL algorithm described previously.

Currently $f(\mathbf{x}_i, \mathbf{q})$ is C/V where C is the value that the cells are initialized to and V is the current value in the cell. As the cell value is increased, $f(\mathbf{x}_i, \mathbf{q})$ decreases and the data point's apparent distance is reduced. In this implementation V must be restricted from reaching 0.

5.3 Updating Cell Values

When the results of performing a primitive are observed, the values associated with the data points used to select the primitive type and parameters can be adjusted to reflect the performance. If the values are set in isolation for each primitive performed, the agent will not consider what can be done from the state the environment is left in when the primitive completes. The agent may make large progress during a primitive, for example, but when it completes it leaves the marble heading very quickly towards a hole. There are no possible actions that can now be taken to recover and the marble will always fall into the hole. There must be some way to propagate this information back to previous primitive selections.

The Q-learning function lends itself very well to updating the values while taking into account the result of actions taken in the future. The Q values in the Q-learning algorithm, $Q(s, a)$, are normally used to provide the expected future reward that can be obtained by taking action a from state s. The results are observed when an action is taken and the values are updated. Rewards are given to guide the values to provide the desired outcome. In our implementation, the Q values, $Q(\mathbf{q}, \mathbf{x}_m)$, are located in the cells of each data point and provide an indication of the expected reward that can be obtained by selecting data point $\mathbf{x}_m$ from query point $\mathbf{q}$. The value V, described in the previous section, becomes the Q value. Since the selected data points and their distance from the query point determine the action that is taken, the Q values have a direct impact on manipulating the action that is chosen at query point $\mathbf{q}$.

The cells are initialized with a constant, C, and then updated using a modified version of the Q-learning function. Since there are N data points that are used, N Q-values must be updated at each evaluation step For each data point chosen, $\mathbf{x}_m$, $m = 1, N$, the Q-values $Q_t(\mathbf{q}_t, \mathbf{x}_m)$ are updated as follows:

$$Q_t(\mathbf{q}_t, \mathbf{x}_m) = Q_t(\mathbf{q}_t, \mathbf{x}_m) + \alpha \cdot \left[\begin{array}{c} r + Q_{t+1}(\mathbf{q_{t+1}}, \hat{\mathbf{x}}_\mathbf{m}) - \\ Q_t(\mathbf{q}_t, \mathbf{x}_m) \end{array} \right]$$

- $Q_t()$ represents the Q-value of the data points that were chosen at time t.

- α is the learning rate, since multiple data points are used, the distance given by $\frac{K(\hat{d}(\mathbf{x}_i,\mathbf{q}_t))}{\sum_{j=1}^{N} K(\hat{d}(\mathbf{x}_j,\mathbf{q}_t))}$ is used as the learning rate. The distance returned by this equation for each data point used is between 0 and 1.0 and distances of all the points used add up to 1.0. This has the effect of having the points that contributed the most toward selecting the primitive and parameters having the highest learning rate.
- r is the reward observed after the primitive has been performed.
- $Q_{t+1}(\mathbf{q_{t+1}}, \hat{\mathbf{x}})$ is the future reward that can be expected from the new state $\mathbf{q_{t+1}}$ and selecting the data points $\hat{\mathbf{x}}_\mathbf{m}$ at the next time step. This value is given by: $\sum_{i=1}^{N} \left[Q_{t+1}(\mathbf{q_{t+1}}, \hat{\mathbf{x}}_i) \cdot \frac{K(\hat{d}(\hat{\mathbf{x}}_i,\mathbf{q_{t+1}}))}{\sum_{j=1}^{N} K(\hat{d}(\hat{\mathbf{x}}_j,\mathbf{q_{t+1}}))} \right]$

A primitive ends when the sub-goal location is reached, if the marble rolls outside a bounding box that contains the start and end location, or if there are small or no changes in the environment state during the execution of the primitive. When the primitive begins and ends the state of the environment is recorded and rewards are assigned to communicate to the agent which actions are most effective in completing the task. In the tilt maze task the agent receives positive rewards for making progress towards the goal location and negative rewards for taking up time.

5.4 Result of Learning Through Practice

The same agents described in Section 4.2 are now given the ability to change their policy while operating in the environment. The bottom picture in Figure 6 shows the path of the marble for ten games played by the hardware agent after it has practiced for 30 games. This agent has completed the maze ten consecutive times without falling into a hole. The dashed (blue) line on the graph in Figure 7 shows the performance of the software agent playing the game with the ability to update its primitive and parameter selection policy. Again the graph shows 30 trials with each trial being the average of three games. This agent immediately decreased the time it takes to complete the maze and its performance is as good or better than that of the observed human's average of 28.77 seconds.

6 Discussion

In learning from observation, the performance of the learning agent is dependent on the performance of the observed agent. The actions available to the learning agent are limited by what is observed. The goal of this research is not to create agents that operate optimally in the environment, but to create agents that can learn to perform proficiently in a short time and have the ability to increase their performance through practice. Even though the agent's first attempt is to perform the same actions as the observed teacher, it is not a requirement to do so. If the agent, while practicing, performs an action that was not previously observed, but moves the marble through the maze very quickly, the agent will try to perform the same action in the future. This can be seen in Figure 6 around hole 14. In the three observed games the human maneuvers the marble below hole 14. During practice, middle picture, the agent falls into hole 14 and learns that it can more easily maneuver the marble around the top of this hole. The human player did not know this action was possible until they observed the action discovered by the agent.

The size of the cells associated with a data point were chosen manually through trial and error with the cells near the center being smaller then those further away. Using this system assumes that the data points will be chosen close to their origin in the state space and that by making the closer cells smaller will allow a finer distinction of the data point's effect on the outcome when chosen in this area. But if the data point is chosen far from its origin, it will not be possible to make fine distinctions. There is a trade-off between generalization, learning speed, and ability to make fine distinctions.

From the results of this initial implementation it can be seen that choosing a Q-value in relation to the query point is very useful. But there needs to be simpler and more effective methods to associate Q-values with a data point. We are currently exploring a method that encodes the value function in a locally weighted projection regression (LWPR) model [15] that is created for each data point. The LWPR approach was chosen because new data can be added very easily and the new information is available for use immediately without having to go off line to train the model on the new information. The problem with most locally weighted learning methods is that each data point added to the model increases the time needed to compute a solution [2]. LWPR maintains a reasonably stable lookup time so data may continuously be added. It is a nonparametric local learning system that uses locally linear models, spanned by a small number of univariate regressions in selected directions in the input space. LWPR is proving its usefulness in such tasks as inverse-dynamics learning [10] and inverse kinematics learning [14].

7 Conclusions

Choosing actions to perform when operating in a dynamic environment, such as the Tilt Maze environment described in this paper, is a difficult task. Because the state space is large and continuous, expecting to learn entirely from random actions is not realistic. An initial policy can be created using knowledge of primitive

actions performed in the environment and information obtained from observing others. Within our research we find that the performance of the initial policy is quite high but there is still room for improvement. This initial policy provides a very good starting point from which to practice to further increase competence at the task.

Our learning from observation using primitives framework described in Section 2 provides flexibility in conducting research in learning from observing others. The framework uses the observed data in a systematic way, and provides the ability to learn while practicing. The organization of the data allows lookups to be performed using LWL techniques. The algorithm described in Section 5 is effective for updating the policy used during primitive selection and sub-goal generation.

Agents using this framework have learned an initial policy to use in the hardware and software Tilt Maze environments. The agents can traverse most of the maze after only observing a few games performed by a human. The agents go on to increase performance at the task by updating their primitive and parameter selection policy while practicing the task. Further testing of our learning from execution method is being conducted to gain a better understanding of the effect the various parameters have on the learning rate. Learning methods that reduce the number of parameters are also being explored. We are also extending this research to an Air Hockey environment [4].

8 Acknowledgments

Support for all authors was provided by ATR Computational Neuroscience Laboratories, Department of Humanoid Robotics and Computational Neuroscience, and the Communications Research Laboratory (CRL). It was also supported in part by the National Science Foundation Award 0325383.

References

[1] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, MA, 1998.

[2] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11:11–73, 1997.

[3] D. C. Bentivegna and C. G. Atkeson. A framework for learning from observation using primitives. In *Proceedings of the RoboCup 2002 International Symposium.*, Fukuoka, Japan, 2002.

[4] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning from observation and practice at the action generation level. In *IEEE-RAS International Conference on Humanoid Robotics (Humanoids 2003)*, Karlsruhe, Germany, 2003.

[5] D. C. Bentivegna, A. Ude, C. G. Atkeson, and G. Cheng. Humanoid robot learning and game playing using pc-based vision. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems.*, Switzerland, 2002.

[6] A. Fod, M. Mataric, and O. Jenkins. Automated derivation of primitives for movement classification. In *First IEEE-RAS International Conference on Humanoid Robotics (Humanoids-2000)*, MIT, Cambridge, MA, 2000.

[7] Y. Kuniyoshi, M. Inaba, and H. Inoue. Learning by watching: Extracting reusable task knowledge from visual observation of human performance. In *IEEE Transactions on Robotics and Automation*, pages 799–822, 1994.

[8] M. J. Mataric, M. Williamson, J. Demiris, and A. Mohan. Behavior-based primitives for articulated control. In *Fifth International Conference on Simulation of Adaptive Behavior (SAB-98)*, pages 165–170. MIT Press, 1998.

[9] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.

[10] S. Schaal, C. Atkeson, and S. Vijayakumar. Scalable locally weighted statistical techniques for real time robot learning. In *Applied Intelligence - Special issue on Scalable Robotic Applications of Neural Networks*, volume 17, pages 49–60, 2002.

[11] R. A. Schmidt. *Motor Learning and Control*. Human Kinetics Publishers, Champaign, IL, 1988.

[12] R. Sutton and A. Barto. *Reinforcment Learning: An Introduction*. MIT Press, 1998.

[13] C. Tung and A. Kak. Automatic learning of assembly tasks using a dataglove system. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, 1995.

[14] S. Vijayakumar, A. DŚouza, T. Shibata, J. Conradt, and S. Schaal. Statistical learning for humanoid robots. In *Autonomous Robot*, volume 12, pages 55–69, 2002.

[15] S. Vijayakumar and S. Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000)*, Stanford, CA, 2000.

Teaching Bayesian Behaviours to Video Game Characters

Ronan Le Hy Anthony Arrigoni Pierre Bessière Olivier Lebeltel

GRAVIR/IMAG
INRIA Rhône-Alpes, ZIRST
38330 Montbonnot, France
lehy@imag.fr

***Abstract*— This article explores an application of Bayesian Programming to behaviours for synthetic video games characters. We address the problem of real-time reactive selection of elementary behaviours for an agent playing a first person shooter game. We show how Bayesian Programming can lead to condensed and easier formalisation of finite state machine-like behaviour selection, and lend itself to learning by imitation, in a fully transparent way for the player.**

I. Introduction

Today's video games feature synthetic characters involved in complex interactions with human players. As John Laird sums it up [10], a synthetic character may have one of many different roles: tactical enemy, partner for the human, strategic opponent, simple unit amongst many, commenter... In all of these cases, the game developer's ultimate objective is for the synthetic character to act like a human player.

We are interested in a particular type of synthetic character, which we call a *bot* in the rest of this paper. It is a player for a first person shooter game named *Unreal Tournament* augmented with the Gamebots control framework [6] (see figure 1). This framework provides a tridimensional environment in which players have to fight each other, taking advantage of resources such as weapons and health bonuses available in the arena. We believe, with Laird [10], [8], that this kind of computer game provides a challenging ground for the development of human-level AI.

After listing our practical objectives, we will present our bayesian model. We will show how we use it to specify by hand a behaviour, and how we use it to learn a behaviour. We will tackle learning by example using a high-level interface, and then the natural controls of the game. We will show that it is possible to map the player's actions onto bot states, and use this reconstruction to learn our model. Finally, we will come back to our objectives as a conclusion.

A. Objectives

Our core objective is to propose an efficient way to specify a behaviour for our bot. This can be broken down into several criteria that hold either for the developers or for the player.

Fig. 1. Unreal Tournament and the Gamebots environment.

1) Development Team's Viewpoint:

a) Programming efficiency: One crucial concern for the programmer is productivity: he needs both expressivity and simplicity of the behaviour programming system.

b) Limited computation requirements: The processing time alloted to AI in games is typically between 10% and 20% of the total processing time [13]; therefore it is important for the behaviour system to be light in terms of computation time.

c) Design / development separation: The industrial development scheme often draws a separation between game designers and engine developers. The system should allow the designers to describe behaviours at a high conceptual level, without any knowledge of the engine's internals.

d) Behaviour tunability: The ability to program a variety of different behaviours, and to adjust each of them without having to modify the system's back end is essential to the designer.

2) Player's Viewpoint:

a) "Humanness": As defined by Laird [7], this implies the illusion of spatial reasoning, memory, common sense reasoning, using goals, tactics, planning, communication and coordination, adaptation, unpredictability... One important criterion for the player is that the synthetic character does not cheat; its perceptions and actions should be as much as possible like a human player's.

b) Behaviour learning: This feature is gradually finding its place in modern games: the player can adjust

its synthetic partners' behaviour. The behaviour system should therefore support learning.

B. Technical Framework

As mentioned earlier, we used the Gamebots framework to conduct our experiments. This implies that our bot communicates with Unreal Tournament via a text protocol on a Unix socket. It receives messages covering its perceptions: its position and speed, health level, ammunition, visible opponents and objects, etc. In return, it sends actions: move to a given point, rotate, change weapon...

The environment is perceived by the bot as a graph, of which nodes are characteristic points of the topology and various objects. The bot perceives only what is in its field of vision.

As our objectives and framework have been exposed, we shall now proceed to explicit our model of behaviour selection, and discuss its interest for the specification and learning of behaviours.

II. Bayesian Model

Before examining our particular bot model, we review in the next section the principles of Bayesian Programming [11].

A. Bayesian Programming

Rational reasoning with incomplete and uncertain information is quite a challenge. Bayesian Programming addresses this challenge, and relies upon a well established formal theory: the probability theory [4]. As a modeling tool, it encompasses the framework of Bayesian Networks [5].

$$Program \begin{cases} Description \begin{cases} Relevant\ Variables \\ Decomposition \\ Parametric\ Forms \end{cases} \\ Question \end{cases}$$

Fig. 2. structure of a Bayesian Program

In our framework, a Bayesian Program is made of two parts: a *description* and a *question*.

The description can be viewed as a knowledge base containing the a priori information available about the problem at hand. It is essentially a joint probability distribution. The description is made up of three components: 1) A set of *relevant variables* on which the joint distribution is defined. Typically, variables are motor, sensory or internal. 2) A *decomposition* of the joint distribution as a product of simpler terms. It is obtained by applying Bayes theorem and taking advantage of the conditional independencies that may exist between variables. 3) The *parametric forms* assigned to each of the terms appearing in the decomposition (they are required to compute the joint distribution).

Given a distribution, it is possible to ask *questions*. Questions are obtained first by partitioning the set of variables into three sets: (1) *Searched*: the searched variables, (2) *Known*: the known variables, and (3) *Free*: the free variables. A question is then defined as the distribution:

$$P(Searched \mid Known) \tag{1}$$

Given the description, it is always possible to answer a question, *i.e.* to compute the probability distribution $P(Searched \mid Known)$. To do so, the following general inference is used:

$$\begin{aligned} & P(Searched \mid Known) \\ &= \frac{\sum_{Free} P(Searched\ Free\ Known)}{P(Known)} \\ &= \frac{1}{Z} \times \sum_{Free} P(Searched\ Free\ Known) \end{aligned} \tag{2}$$

where Z is a normalisation term.

As such, the inference is computationally expensive (Bayesian inference in general has been shown to be NP-Hard [2]). A symbolic simplification phase can reduce drastically the number of sums necessary to compute a given distribution. However the decomposition of the preliminary knowledge, which expresses the conditional independencies of variables, still plays a crucial role in keeping the computation tractable.

B. Modelling our Bot

1) Bayesian Program: Our particular bot behaviour uses the following bayesian program.

a) Relevant Variables:

- S_t: the bot's state at time t. One of *Attack*, *SearchWeapon*, *SearchHealth*, *Explore*, *Flee*, *DetectDanger*. These states correspond to elementary behaviours, in our example programmed in a classic procedural fashion.
- S_{t+1}: the bot's state at time $t+1$.
- H: the bot's health level at t.
- W: the bot's weapon at t.
- OW: the opponent's weapon at t.
- HN: indicates whether a noise has been heard recently at t.
- NE: the number of close enemies at t.
- PW: indicates whether a weapon is close at t.
- PH: indicate whether a health pack is close at t.

The elementary motor commands of the bot are the values of variables S_{t+1} and S_t. They include an attack behaviour, in which the bot shoots at an opponent while keeping a distance to him and strafing; a fleeing behaviour, which consists in trying to escape (locally) an opponent; behaviours to fetch a weapon or a health bonus the bot noticed in its environment; a behaviour to detect possible

opponents outside the current field of view of the bot; and behaviour to navigate around the environment and discover unexplored parts of it.

b) Decomposition: The joint distribution is decomposed as:

$$\begin{aligned} & P(S_t\ S_{t+1}\ H\ W\ OW\ HN\ NE\ PW\ PH) \\ = \quad & P(S_t) \\ & P(S_{t+1}|S_t) \\ & P(H|S_{t+1}) \\ & P(W|S_{t+1}) \\ & P(OW|S_{t+1}) \\ & P(HN|S_{t+1}) \\ & P(NE|S_{t+1}) \\ & P(PW|S_{t+1}) \\ & P(PH|S_{t+1}) \end{aligned}$$

To write the above, we make the hypothesis that knowing S_{t+1}, any sensory variable is independent to each other sensory variable. Although it may seem to reduce the expressivity of our model, it allows to specify it in a very condensed way; this point will be emphasised upon in section II-B.2.

c) Parametric Forms:

- $P(S_t)$: unknown (unspecified)
- $P(S_{t+1}|S_t)$: table (this table will be defined in section II-B.2)
- $P(Sensor|S_{t+1})$ with $Sensor$ each of the sensory variables: tables

d) Identification: Identification of the parametric forms is done either by manually writing the tables, or by learning them. We describe these two processes in sections III (*Specifying a Behaviour*) and IV (*Learning a Behaviour*).

e) Question: Every time our bot has to take a decision, the question we ask to our model is:

$$P(S_{t+1}|S_t\ H\ W\ OW\ HN\ NE\ PW\ PH)$$

Knowing the current state and the values of the sensors, we want to know the new state the bot should switch into. This question leads a probability distribution, on which we draw a value to decide the actual new state. This state translates directly into an elementary behaviour which is applied to the bot.

2) Inverse Programming: We shall now emphasise the peculiarities of our method to specify behaviours, compared to one using simple finite state machines (FSMs). The problem we address is, knowing the current state and the sensors' values, to determine the next state: this is actually naturally accomplished using an FSM.

Let us consider the case where each of our n sensory variables has m_i ($1 \leq i \leq n$) possible values.

In an FSM modelling a behaviour [3], [14], we would have to specify, for each state, a transition to each state, in the form of a logical condition on the sensory variables.

It means that the programmer has to discriminate amongst the $\prod_i m_i$ possible sensory combinations to describe the state transitions. Not only does this pose the difficult problem of determining the appropriate transitions, but it raises the question of convenient formalised representation. This approach could actually lead to several implementations, but will possibly [1] result in a script resembling the following:

```
if S_t =A and W =None and OW =None then
   if HN =False and NE!=None
      or NE =TwoOrMore then
         S_{t+1} ← F
   else if HN =True or NE =One
      and PW =True then
         S_{t+1} ← A
   else ...
```

This kind of script is hard to write and hard to maintain.

In contrast, our approach consists in giving, for each sensory variable, for each possible state, a distribution (i.e. m_i numbers summing to 1). In practice, we write tables like table 3, which represents $P(H|S_{t+1})$. Values of H are enumerated in the first column, those of S_{t+1} in the first line; cells marked x are computed so that each column sums to 1.

Moreover, instead of specifying the conditions that make the bot switch from one state to another, we specify the (probability distribution of the) sensors' values when the bot goes into a given state. This way of specifying a sensor under the hypothesis that we know the state is what makes us call our method "inverse programming".

Although somewhat confusing at first, this is the core advantage of our way to specify a behaviour. As a matter of fact, we have to describe separately the influence of each sensor on the bot's state, thereby reducing drastically the quantity of needed information. Furthermore, it becomes very easy to incorporate a new sensory variable into our model: it just requires to write an additional table, without modifying the existing ones.

Finally, the number of figures we need in order to specify a behaviour is $s^2 + snm$, where s is the number of states, n the number of sensory variables, and m the average number of possible values for the sensory variables. It is therefore linear in the number of variables (assuming m constant).

$H \setminus S_{t+1}$	A	SW	SH	Ex	F	DD
Low	0.001	0.1	x_3	0.1	0.7	0.1
$Medium$	0.1	x_2	0.01	x_4	0.2	x_5
$High$	x_1	x_2	0.001	x_4	0.1	x_5

Fig. 3. $P(H|S_{t+1})$

III. Specifying a Behaviour

A. Basic specification

A behaviour can be specified by writing the tables corresponding to $P(S_{t+1}|S_t)$ and $P(Sensor|S_{t+1})$ (for each sensory variable). Let us consider for instance table 3, which gives the probability distribution for H knowing S_{t+1}. We read the first column this way: given the fact that the bot is going to be in state *Attack*, we know that it has a very low probability (0.001) to have a low health level, a medium probability (0.1) to have a medium health level, and a strong chance ($x = 1 - 0.001 - 0.1$) to have a high health level.

This form of specification allows us to formalise conveniently the constraints we want to impose on the behaviour, in a condensed format, and separately on each sensory variable. For instance, table 3 formalises the relation of the bot's health level to its state: if it starts attacking, then its health is rather high; if it starts searching for a health pack, then its health is very probably low; if it starts fleeing, then its health is probably rather low, but with a high degree of uncertainty.

All tables on the sensory variables are built on the same pattern; the one giving $P(S_{t+1}|S_t)$ (see table 4) is special. It gives some sort of basic transition table; i.e. it answers in a probabilistic way the question: knowing nothing but the current state, what will be the next state?

$S_{t+1} \setminus S_t$	A	SW	SH	Ex	F	DD
A	x_1	x_2	x_3	x_4	x_5	x_6
SW	10^{-5}	x_2	10^{-5}	10^{-5}	10^{-5}	10^{-5}
SH	10^{-5}	10^{-5}	x_3	10^{-5}	10^{-5}	10^{-5}
Ex	10^{-5}	10^{-5}	10^{-5}	x_4	10^{-5}	10^{-5}
F	10^{-5}	10^{-5}	10^{-5}	10^{-5}	x_5	10^{-5}
DD	10^{-5}	10^{-5}	10^{-5}	10^{-5}	10^{-5}	x_6

Fig. 4. $P(S_{t+1}|S_t)$

The answer our sample table gives is: tend to stay in your current state (notice the xs on the diagonal) or switch to attack (notice the xs on the first line) with the same high probability; switch to other states with a very low probability (10^{-5} – which in our example we found to be representative of "very low").

Again, this makes a parallel with an FSM with probabilistic transitions: with our transition table $P(S_{t+1}|S_t)$, we give a basic automaton upon which we build our behaviour by fusing the tendencies given separately on each sensory variable.

B. Tuning the behaviour

Tuning our behaviour amounts to tuning our probability distributions. For instance, to create a *berserk* character that is insensible to its health level, we put only uniform distributions (i.e. in our notation, only xs) in table $P(H|S_{t+1})$. A *berserk* is also very aggressive, so the transition table we proposed in table 4 is quite adapted. A transition table for a more prudent character would not have those xs on the first line, so that the state A would not be particular.

To create a unique behaviour, we therefore have to review all our tables, i.e. the influence of each sensory variable on the character according to the said behaviour.

C. Results

Several observations can be made when out bots are playing the game. The first is that their behaviour corresponds to what we want: the behaviour switching occurs reasonably, given the evolution of the sensory variables. The second is that they can't compete with humans playing the game. Noting this allows to pinpoint the fact that our method's interest mostly resides in the gain of ease and power in the design of behaviours. It does not pretend to overcome the limitations of the elementary behaviours we are switching between, nor can it do more than what the Gamebots framework allow, in terms of perception and action. Therefore, what we aimed for, and finally obtained, is a reliable, practical and efficient way to specify the real-time selection of elementary behaviours.

Our attempt to tune the behaviour shows that the differences between our 'reasonable' bot and our 'aggressive' bot are visible, and correspond to what we tried to specify in the tables. For instance, the aggressive bot is excited by the presence of several opponents, whereas this situation repels the reasonable bot; and the aggressive bot is not discouraged to attack when its health level goes low.

IV. Learning a Behaviour

Our goal now is to teach the bot a behaviour, instead of specifying all the probability distributions by hand. It requires to be able to measure at each instant sensory and motor variables of the controlled bot. In particular, it is necessary to determine the state S_t at each instant. It can be done by letting the player specify it directly in real time, or by inferring it from his natural actions in the game.

A. Selecting behaviours

This form of learning by example presents a simple interface to the player, shown on figure 5.

The player controls in real time the elementary behaviour that the bot executes, by using buttons that allow to switch to each state with a mouse click. In addition to the ordinary Unreal Tournament window on the right, part of the internal state of the bot is summed up in the learning interface on the left.

Fig. 5. Interface used to teach the bot: on the right is the normal Unreal Tournament window showing our bot; on the left is our interface to control the bot.

B. Recognising behaviours

While the previous method of teaching a behaviour works, it deprives the player of the interface he is used to; his perceptions and motor capabilities are mostly adjusted to the bot's. In order to solve this problem, it is possible to give the player the natural interface of the game, and try to recognise in real time the behaviour he is following.

To recognise the human's behaviour from his low-level actions, we use a heuristic programmed in a classical imperative fashion. It involves identifying each behaviour's critical variables (for instance, attack is characterised by distance and speed of the bot to characters in the centre of his field of view), and triggering recognition at several timescales.

Behaviours are recognised after a delay (back-propagating a state to past, yet unrecognised events, thanks to a critical event like picking a health bonus), after examining critical variables over a fixed period (to identify danger checking behaviours, for instance), or immediately on special events (like attacking and fleeing). Exploration is a default state, when no other seems to match the observations.

We do this recognition off-line, on data representing ten to fifteen minutes of game-play; processing this data and producing the tables that represent our behaviour takes five to ten seconds.

C. Results

recognition learned, aggressive	4.4
recognition learned, cautious	13.9
selection learned, aggressive	45.7
manual specification, aggressive	8.0
manual specification, cautious	12.2
manual specification, uniform	43.2
native (level 3/8) UT bot	11.0

Fig. 6. Performance comparison on learned, hand-specifi ed, and native Unreal bots (lower is better)

Table on figure 6 shows a comparison between different specification methods. Numbers are the average difference to the winning bot, over ten games won by the first bot reaching 100 kills. Our bots compare well to the native UT bot, whose skill corresponds to an average human player. Aggressive bots (gray lines) perform significantly better, and learning by recognition does much better than learning by selection, along with hand specification.

Lessons from these results can be summed up in the following way (we will refer here to the table on figure 7, which is the same as figure 3, but learnt by recognition):

1) learnt tables share common aspects with hand-written tables (as for the transition table $P(S_{t+1}|S_t)$); for instance,in the fleeing state F, health level is much more probably low or average than high;
2) differences in behaviour of the teacher influence the learnt behaviour: aggressivity (or the lack of it) is found in the learnt behaviour, and translates into performance variations (in our setup, aggressive behaviours seem to be more successful);
3) nevertheless, differences between hand-specified and learnt models are noticeable; they can explained by:
   a) player-specific behaviours: humans almost always attack and do not retreat; another example is the low probability of $P(H = High|S_{t+1} = SW)$ in the learnt table (dark gray cell on figure 7): it can be explained by the fact that human players give a much higher priority to searching a good weapon over searching for health bonuses;
   b) additional information: some parts of the hand-written tables are specified as uniform (as a result from a refusal or impossibility to specify theoretically a link between two events, like the value of the opponent's weapon knowing that the bot is exploring), whereas their learnt counterparts include information;
   c) perceptive differences: a human player and a bot have a different perception of sound (the human perceives direction combined with the origin of sound, like an impact on a wall or the sound of the shooting itself, whereas the bot senses only direction);
   d) bias induced by data quantity: a human player has almost always an average health level (which is due to a poor choice of discretisation for the health level variable), which explains higher values in the learnt table in figure 7 (line of light gray cells);
4) our learning methods lead to functioning behaviours; learning using behaviour recognition scores best, and allows to reach the level of an average native UT bot.

$H \setminus S_{t+1}$	A	SW	SH	Ex	F	DD
Low	0.179	0.342	0.307	0.191	0.457	0.033
Average	0.478	0.647	0.508	0.486	0.395	0.933
High	0.343	0.011	0.185	0.323	0.148	0.033

Fig. 7. learnt $P(H|S_{t+1})$

V. Discussion

A. Evaluation

We shall now come back to the objectives we listed at the beginning, to try and assess our method in practical terms.

1) Development Team's Viewpoint:

a) Programming efficiency: Our method of behaviour design relies upon a clear theoretical ground. Programming the basic model can use a generic bayesian programming library, and needs afterwards little more than the translation into C++ (for instance) of the mathematical model. Design is really expressed in terms of practical questions to the expertise of the designer, like "if the bot is attacking, how high is his health level?"; it does not require a preliminary formalisation of the expected behaviour to program. Moreover, in our model behaviours are data (our tables). It means that they can easily be loaded and saved while the behaviour is running, or exchanged amongst a community of players or developers.

b) Limited computation requirements: The computation time needed for a decision under our model can be shown to be linear in both the number of sensory variables and the number of states.

c) Design / Development separation: Development amounts to incorporating the bayesian framework into the control architecture, and establishing the bayesian model; design consists in establishing relations between the variables in the form of probability distributions. A designer really has to know about what the bot should do, but does not need any knowledge of the implementation details; he needs but a light background on probabilities, and no scripting or programming at all.

d) Behaviour tunability: We have seen that our way of specifying behaviours gives a natural way to formalise human expertise about behaviours, and that it implies that tuning a behaviour is possible, as they are expressed in natural terms and not in artificial logical or scripting terms. Moreover, the quantity of data needed to describe a behaviour is kept small compared to an FSM, and this helps keeping the analysis and control of a behaviour tractable for the designer.

2) Player's Viewpoint:

a) "Humanness": This criterion is hard to assess, although it can be done [9] in ways comparable to the Turing test [12]. Our method of specifying a behaviour helps the designer translate his expertise easily, and therefore gives him a chance to build a believable bot.

b) Behaviour learning: We have seen that learning under our model is natural: it amounts to measuring frequencies. This is a chance for the player to teach its teammate bots how to play. Recognising high-level states on the basis of low-level commands is possible, and allows a player to adjust a behaviour completely transparently, with the original controls of the game.

B. Perspectives

We have shown a way to specify FSM-like action selection models for virtual robots, and to learn these models by example. The recognition involved in learning from the natural actions of a player in the game remains a classicaly programmed heuristic; an obvious perspective is to formalise it within the bayesian framework, in order to perform probabilistic behaviour recognition. This would grant more adaptability to variations in the behaviour model.

Acknowledgments

This work was partially funded by the ROBEA-CNRS project "Modeles Bayesiens pour la Generation de Mouvement", the BIBA project funded by the European Community, and a grant from the French Ministry of Research.

VI. References

[1] Unrealscript language reference. Website. `http://unreal.epicgames.com/UnrealScript.htm`.

[2] G. Cooper. The computational complexity of probabilistic inference using bayesian belief network. *Artificial Intelligence*, 42(2-3), 1990.

[3] E. Dysband. A finite-state machine class. In M. Deloura, editor, *Game Programming Gems*, pages 237–248. Charles River Media, 2000.

[4] E. T. Jaynes. Probability theory: the logic of science. Unprinted book, available on-line at `http://bayes.wustl.edu/`, 1995.

[5] M. Jordan, editor. *Learning in Graphical Models*. MIT Press, 1998.

[6] G. A. Kaminka, M. Veloso, S. Schaffer, C. Sollitto, R. Adobbati, Andrew N. Marshal, S. Scholer, Andrew, and S. Tejada. Gamebots: the ever-challenging multi-agent research test-bed. *Communications of the ACM*, January 2002.

[7] J. Laird. Design goals for autonomous synthetic characters. Draft, 2000.

[8] J. E. Laird. It knows what you're going to do : Adding anticipation to a quakebot. In *AAAI Spring Symposium Technical Report*, March 2000.

[9] J. E. Laird and J. C. Duchi. Creating human-like synthetic characters with multiple skill-levels : A case study using the Soar quakebot. In *AAAI Fall Symposium Technical Report*, August 2000.

[10] J. E. Laird and M. Van Lent. Human-level AI's killer application : Interactive computer games. In *AAAI Fall Symposium Technical Report*, August 2000.

[11] O. Lebeltel, P. Bessière, J. Diard, and E. Mazer. Bayesian Robot Programming. *Autonomous Robots*, 2003. In press.

[12] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):433–460, 1950.

[13] S. Woodcock. Game AI : The state of the industry 2000-2001. *Game Developer*, August 2001.

[14] M. Zarozinski. Imploding combinatorial explosion in a fuzzy system. In M. Deloura, editor, *Game Programming Gems 2*, pages 342–350. Charles River Media, 2001.

Towards Robot Intermodal Matching Using Spiking Neurons

Emachi Eneje & Yiannis Demiris
Intelligent & Interactive Systems Group
Department of Electrical and Electronic Engineering
Imperial College London
Exhibition Road, London, SW7 2BT
UK

***Abstract*— For a robot to successfully learn from demonstration it must posses the ability to reproduce the actions of a teacher. For this to happen, the robot must generate motor signals to match its proprioceptively perceived state with that of the visually perceived state of a teacher. In this paper we describe a real time matching model at a neural level of description. Experimental results from matching of arm movements, using dynamically simulated articulated robots, are presented.**

I. INTRODUCTION

It is beneficial for biological organisms to copy the actions of their conspecifics, as this greatly accelerates their cognitive and sensorimotor development. Learning by imitation has recently attracted that attention of the roboticist (see review in [5]), since devising mechanisms that allow robots to imitate will open possibilities for learning through demonstration. Its been suggested that imitation occurs on several different levels [10].
1) Stimuli level imitation, that is reproducing exact copies of perceived stimuli. 2) Functional level imitation, and 3) at a social level imitation, this is thought to be roots of empathy. Viewing imitation at the stimuli level, a question remains as to how it is possible to match physical movements without the use of the visual modality in generating a matching evaluation critic in perceptually opaque cases, for example facial gesture imitation. This type of imitation raises a couple of interesting questions. First, the metrics used to detect cross-modal equivalences in human acts. Second, the process by which they correct their imitative errors [3]. Investigations by Meltzoff [2], [3] focus on this kind of imitation process. His experiments in facial gesture imitation suggests that infants are capable of imitating, even hours after birth. In accounting for his finding, Meltzoff proposed an *Active Intermodal Matching* (AIM) hypothesis. In AIM, the infant is able to match facial gestures by performing intermodal matching. That is translating visual perceived stimuli from an external coordinate frame to a viewer centered representation, which can be used along with the viewers proprioceptive state to drive the matching process.

In this paper we implement a matching model at a neural level of description. In the model, physical state of the imitator and demonstrator are encoded in the temporal characteristics of the spiking neuron. Preliminary experimental results in a real time matching task using simulated humanoids are reported.

II. BIOLOGICAL INSPIRATION

Biological organisms need a means of acquiring information about their environment, and sensory receptors provide that interface. The topographic organisation of the model is inspired by the somatosensory cortex (SC). The SC if the primary site for somatic information processing. Of the sensations processed in the SC, the proprioceptive sensation is crucial in providing a bodily sense, i.e. the static location of limbs, and the sense of kinesthesia [8]. This bodily sense if facilitated by a class of sensory receptors known as mechanoreceptors, providing information such as limb velocity, position, muscle force and direction of movement. These parameters are encoded by the temporal characteristics of spiking activity. For illustrative purposes, figure 1 shows a shoulder joint mechanoreceptor with a spatially modulated receptive field sensitive in one of three degrees of freedom. Evidence of this kind of spatial modulation of activity is presented in [6]. Given mechanoreceptor operation, limb distribution can be represented by the collective spiking activity of mechanoreceptors located in the body. We also implement a set of neural nodes that encode demonstrator parameters in spiking activity. In the model presented, we assume all relevant demonstrator parameters, joint angles, can be extracted. A review of vision approaches to perceiving animate motion can be found in [4].

III. MATCHING MODEL

The model presented in this paper is shown in figure 2. The joint angles of the demonstrator are extracted and encoded. The neural activity generated in the visual transformation stage is fed to the neural comparator. The imitators somatic information, encoded by mechanoreceptors, is also fed to the neural comparator. The two signals are then combined to the generate control signals that drive the matching process.

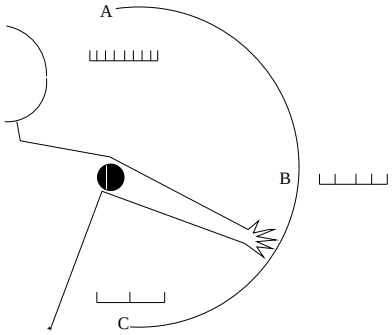


Fig. 1. Shows a single mechanoreceptor attached to the shoulder joint. The activity of this joint mechanoreceptor is spatially modulated

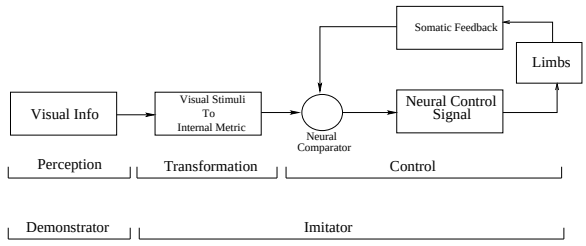


Fig. 2. Schematic of basic matching model presented

A. Neuron Model

The spike accumulator model (SAM) of the neuron is implemented. The equations governing the behaviour of the model we use are given below and are based on an implementation by Tijsseling & Berthouze [7].

$$u_i(t) = \sum_j w_{ij}(t) o_j + \tau v_i(t-1) \tag{1}$$

where $u_i(t)$ is the accumulated potential of neuron i at time t, $v_i(t)$ is the internal, or membrane, potential of neuron i, $o_j(t)$ is the output of connected neuron j, w_{ij} - represents the afferent connection weight between neuron i and j and τ is the decay rate of the of the internal potential $v_i(t)$. The internal potential is given by:

$$v_i(t) = u_i(t) - \rho o_i(t) \tag{2}$$

where ρ is the subtraction constant. SAM output is determined by:

$$o_i(t) = \begin{cases} 1 & \text{if } u_i(t) > T \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where T is the internal threshold level.

B. Mechanoreceptors

We implement a class of position sensitive mechanoreceptors. The equations governing it behaviour is given below:

$$e_i(t) = K_1 \theta_i + b_1 \tag{4}$$

$$a_i(t) = a_i(t-1) + e_i(t) \tag{5}$$

where $e_i(t)$ is the mechanoreceptor excitor, the value of which over the course of time varies in proportion to the angle, θ, of the joint to which the receptor is attached. Constant, b_1, accounts for the resting frequency of the mechanoreceptor. K_1 is a scaling constant. $a_i(t)$ is the accumulated potential of the mechanoreceptor. The output of the mechanoreceptor, $o_i(t)$ at a given time instant is given by;

$$o_i(t) = \begin{cases} 1 & \text{if } a_i(t) > T \ ; \ a_i(t) = 0; \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

C. Neural Comparator

The neural comparator, shown in figure 3, is a three node arrangement. The comparator is a three node arrangement. Nodes *one* and *two* are implemented as SAM nodes, where as the node *Osc*. is an oscillating unit. Nodes one and two receive excitatory and inhibitory afferents of both visual and somatic origin. Nodes *one* and *two* are connected to node *Osc* both via inhibitory connections. Given a trajectory matching task, the activities of nodes *one* and *two* reflect the disparity between the joint state of the corresponding imitator and demonstrator limbs. For a limb correspondence there will be little disparity between the spiking activity of visual neurons and mechanoreceptors. This in turn results in no inhibiting signal being sent to node *Osc*. A neural comparator unit is implemented for each of the imitators degrees of freedom. Using this simple arrangement, appropriate matching torques can be generated.

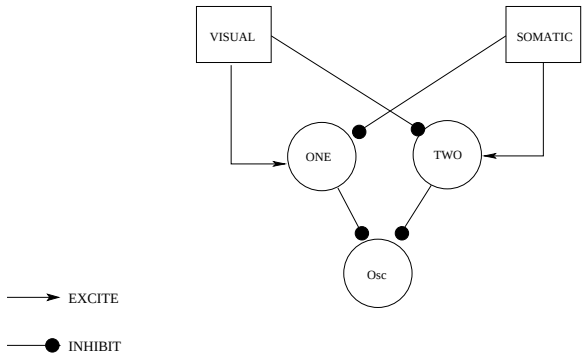


Fig. 3. Comparator assembly implemented on imitator. Three node arrangement acting as a comparator of imitators somatic state (S) of a single degree of freedom of a single limb, with the state of the corresponding limb DOF of the demonstrator (V)

D. Torque Generation

The motor control signals driving the imitator are derived from the node activities of the neural assembly. We define *activity* as the number of spikes within a given

time interval. We denote this time interval, Ts, which may also be viewed as a spike *summation interval*. There are a total of N summation intervals during any given matching simulation run. Thus *activities* of nodes *one*, *two* and *Osc*. during a given summation interval, n, are denoted by A_{one}, A_{two} and A_{Osc} respectively. Thus equations describing torque generation process are given below:

$$\beta_n = A_{one} - A_{two} \tag{7}$$

where β_n is the activity difference between nodes *one* and *two* during summation interval n. From β_n we calculate torque value, λ, for a particular interval n as:

$$\lambda_n = \gamma(\beta_n + base_{Ts}) \tag{8}$$

where $base_{Ts}$ is a base torque constant. γ is the torque coefficient and is calculated as follows:

$$\gamma = \frac{2}{s\sqrt{2\pi}}(\exp^{-\frac{1}{2}(\alpha_1)} + \exp^{-\frac{1}{2}(\alpha_2)}) + F \tag{9}$$

where F is constant to ensure a non-zero γ value. Since γ is a mixed Gaussian curve, its function parameters are based on the difference between joint angles in corresponding imitator and demonstrator limbs. α_1 and α_2 are given by:

$$\alpha_1 = \left(\frac{(\theta_d - \theta_i) + Z}{s}\right)^2 \tag{10}$$

$$\alpha_2 = \left(\frac{(\theta_d - \theta_i) - Z}{s}\right)^2 \tag{11}$$

where Z is is a constant which, along with s, determines the shape of the γ function. θ_d and θ_i are the joint angles of the demonstrator and imitator respectively.

IV. Experiments

A. Platform & Postural Task

Two simulated humanoids, demonstrator and imitator, were implemented using the Dynamechs simulation library [1], a rigid body simulator based on real time physics engine. Each of the two simulated humanoids has 8 degrees of freedom. A modular Proportional-Integral-Derivative control scheme is used to control each DOF of the demonstrator. The matching model described in the last section is implemented on the imitator, thus all the imitators limb control signals are generated by the matching model.

During an imitation trial, spatial targets are fed to the demonstrators controller and a motion trajectory is executed. As the demonstrator executes a movement, the imitator generate torques in order to reproduce the demonstrated trajectory. Results and analysis of model performance is presented in the following section.

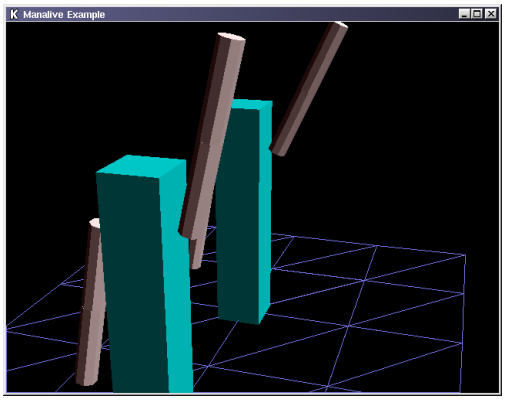


Fig. 4. Shows the two simulated humanoids, imitator closest, during a posture matching task

B. Results

Two different trajectory matching experiments were conducted to assess the performance of our model. First, the imitators ability to match a single demonstrated trajectory. Figures 5 and 6 correspond to the results of this trial. Figure 5 shows how joint angle of demonstrator and imitator varies during the matching trial. At the bottom of the figure the β generated by the imitator during the trial is shown, the polarity of the β graph determines the polarity of the torque signals issued to drive the matching process. Figure 6 shows the activities of the three nodes in the neural comparator during the same single trajectory matching task. This deviation between the demonstrated and the imitated trajectories is due to several factors. First, the humanoids are implemented in a dynamic simulation environment where gravity and friction apply. Second, neural nodes implemented that perform angle transformations have a finite spatial resolution that result in both a delay in imitators response and the final static target offset that are seen in figure 5. Figure 7 shows the results of a matching trial where motion is through multiple via points. Neural assembly parameter A_{Osc} is shown at the bottom of the figure. During the trajectory motion path of the imitator limb is not smooth. This highlights a problem of using spike signals for control. This problem can be overcome by using a distributed population averaged control scheme that utilises multiple receptors per degree of freedom.

V. Conclusions and Future Work

In this paper we present a model dealing with intermodal matching, at a neural level of description, for the purposes of movement imitation. Our results demonstrate the models ability to imitate demonstrated trajectories in real time. We achieve this by representing visual and proprioceptive information using the activity of spiking neurons, and a neural comparator circuit to drive the torque generation process that results in

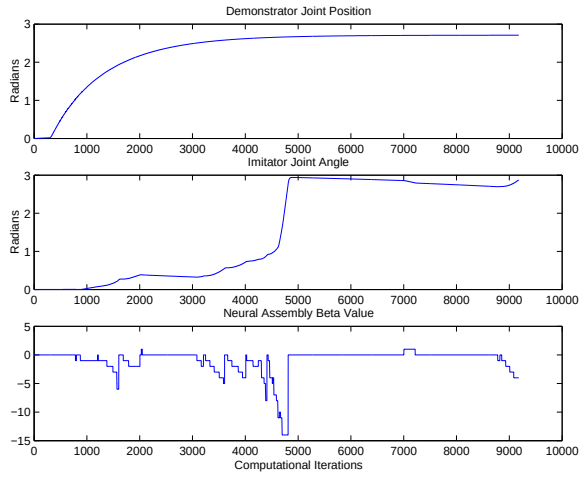


Fig. 5. Shows demonstrator, imitator joint position and generated torque

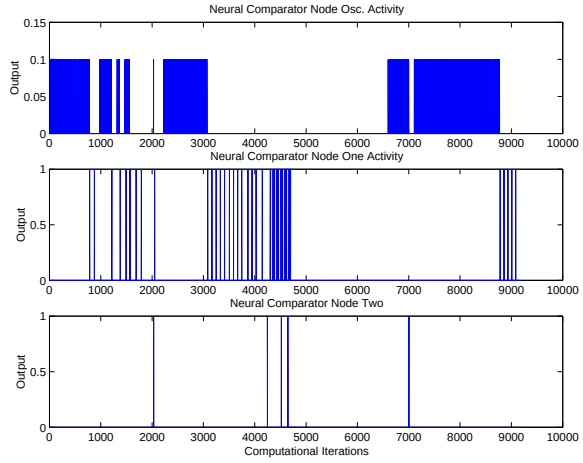


Fig. 6. Shows the activities of neural assembly nodes Osc, One and Two, during a simple trajectory matching task

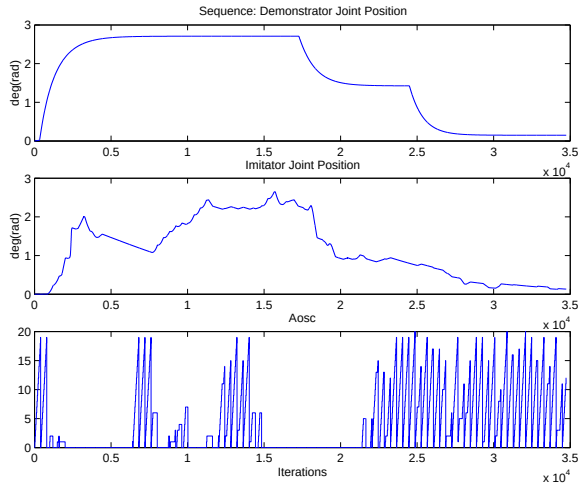


Fig. 7. Figures shows how joint position of demonstrator and imitator varies over time for matching a complex trajectory

intermodal posture matching. By transforming visual and proprioceptive information into spiking neuron activation, we can compare the representations between the two modalities without resolving into geometrical representations of posture, which have been used in previous implementations of the AIM hypothesis [9].

There are several further issues that would be interesting to investigate. Our experiments currently do not perform any elaborate coordinate transformations between the demonstrated movements and the imitated movement. Our next step will add a mental translation and rotation stage, so the imitator is able to imitate demonstrated movements irrespectively of the viewing angle, and we will test the timing issues that will arise from adding such a stage. The robustness of the model to noise also needs to be investigated. We have started investigating this issue through the systematic addition of noise to the visual and proprioceptive values. We are currently also making the first steps towards implementing the model on humanoid robots.

VI. ACKNOWLEDGMENTS

Emachi Eneje is supported by an EPSRC doctoral training award.

VII. REFERENCES

[1] S. McMillan, ”A Computational Framework for Simulation of Underwater Robotic Vehicle System”, *Special Issue of the Journal of Autonomous Robots*, vol. 3, 1996, pp. 253-268.

[2] A.N. Meltzoff, M. K. Moore, ”Imitation, Memory and the Representation of Persons”, *Infant Behavior and Development*, vol. 17, 1994.

[3] A. N. Meltzoff, M. K. Moore, ”Explaining Facial Imitation: A Theoretical Model”, *Early Development and Parenting*, vol. 6, 1997, pp. 179-192.

[4] T. B. Moeslund, E. Granum, ”A Survey of Computer Vision-Based Human Motion Capture”, *Int. Journal of Computer Vision and Image Understanding*, vol. 81, no. 3, 2001

[5] S. Schaal, ”Computational Approaches to Motor Learning by Imitation”, *Trends in Cognitive Science*, vol. 3, 1999, pp. 233-242.

[6] S.I. Helms Tillery, J.F. Soechting, T.J. Ebner, ”Somatosensory Cortical Activity in Relation to Arm Posture: Nonuniform Spatial Tuning”, *Journal of Neurophysiology*, vol. 76, October 1996, pp. 2423-2438.

[7] A. Tijsseling, L. Berthouze, ”A neural network architecture for the categorization of temporal information”, Submitted, 2002

[8] E. R. Kandel, J. H. Schwartz, T. M. Jessell, *Principles of Neural Science* McGraw-Hill, 2000

[9] J. Demiris, S. Rougeaux, G. M. Hayes, L. Berthouze, Y. Kuniyoshi ”Deferred Imitation of Human Head Movements by an Active Stereo Vision Head”, in *Proceedings of the 6th IEEE International Workshop on Robot Human Communication*, pp.88-93,

[10] J. Demiris, *Movement Imitatation Mechanisms in Robots and Humans*, thesis, University of Edinburgh, 1999.