2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

An Intuitive Interface for a Cognitive Programming By Demonstration System

David Brageul, Slobodan Vukanovic, Bruce A. MacDonald, *Senior Member, IEEE*

Abstract— **A significant challenge in programming robots by demonstration is to accurately capture the user's intentions, so that sensor differences can be managed during playback. Sensor difference can be caused by: natural sensory data variations, minor variations in the task conditions, significant changes in the task scenario, or because the task requires a new set of actions to be executed. This paper presents a design for a programming by demonstration system that focuses on the important goal of capturing the intentions of the user during the demonstration. A gesture interface for a large touch screen is used during demonstration, to capture more clearly the user's intentions for robot movements, and also during a pre-playback session to capture the user's intentions regarding sensor data.**

I. INTRODUCTION

UNDENIABLY in a not too distant future, service robots will be part of our everyday life. They will help us to accomplish many diverse tasks, at work and in the home. The great variety of tasks they may have to accomplish means that it will not be possible for robot tasks to be pre-programmed [1]. Robot assistants will constantly have to learn new tasks from their human users. However, most end users are not programmers and cannot be expected to programme their robots using common programming languages. End users must be able to make robots do what they want in an intuitive manner. The Programming by Demonstration (PbD) paradigm is a possible solution [2]-[4].

A. The PbD Principle

PbD has two main phases: the demonstration of the task by the human user, where the task is recorded from the robot's sensor data and actuator controls, and task playback. Arc welding and assembly line paint spraying are examples: a human operator can first perform the task while a motion sensor records the trajectories for later playback [5]. Fetching objects can also by taught by PbD [6].

Research in PbD faces diverse problems, such as identifying what to imitate in a task and how to imitate it [7]. It is important to identify the user's intentions in the demonstration and then provide reasoning methods to ensure the robot achieves the intended goals during playback. In general this requires a third important aspect to PbD, which is the representation and generalisation of the recorded task in a form that is robust to the diverse range of environmental conditions that may prevail during playback.

B. Key Problem: Capturing User Intentions

Capturing the user intention is the first key point for a robust PbD system: without knowledge of the user intention, the PbD system performs pure imitation and thus, in most cases, the robot will not be able to repeat the task if the environment changes [8], [9]. For example, for a fetching task, the position of the objects to fetch might change between the demonstration and the playback of the task. If the user intention could be understood from a demonstration, then this information could be used to generate a programme representation that responds to environment changes in an appropriate manner. The programs generated could then be used to perform the demonstrated tasks in different environments. Possible solutions to take into account the user intention include averaging over several demonstrations [4] and good user interaction [8]. The next key step for a robust PbD system is to reason about the intentions during the playback phase. This paper focuses on the capture of user intentions when demonstrating navigational tasks.

C. Overview of Our Approach

In this paper, we introduce the concept of a PbD system based on "cognitive" PbD, and spbd, its implementation. By "cognitive" we mean that our system tries to respond to environment changes during the playback of a task by reasoning over the knowledge learnt about the user intentions during the demonstration, and from pre-playback interactions with the user that further clarify what they want. Teaching a task by demonstration is thus achieved by an addition to the PbD process, so there are now three main phases:

1) The demonstration. The user controls the robot's trajectories by pointing at the desired locations in a graphical user interface, which displays a view of the robot's working space on a large touch screen. Meanwhile, spbd records the robot's moves and its perceptions of the environment through sensors and actuators. This knowledge is represented by user understandable predicates, such as **clear(front)**, or **obstacle(left, far)**. Our work enhances the

Manuscript received September 14, 2007.

D. Brageul, S. Vukanovic and B. MacDonald are with the Department of Electrical and Computer Engineering, University of Auckland, New Zealand (phone: +6493737599x88157; fax: +6493737461; e-mail: b.macdonald@auckland.ac.nz).

demonstration phase by creating additional ways for the user to express his or her intentions, e.g. the robot waypoints can be edited.

2) **Identification of the user intention.** This stage relies heavily on user interaction and follows the demonstration. It is a pre-playback session, where the task is visualised step by step so that the user can clarify their intention by selecting predicates relevant to the task. High-level tasks can also be taught by combining simpler tasks in this step.

3) The playback. During this phase, the selected predicates are considered as goals to be achieved and basic planning techniques are applied to make the robot work toward those goals in order to repeat the task.

The rest of this paper is organised as follows: Section II gives an overview of relevant PbD research. Section III introduces our proposed model of cognitive PbD, and. Section IV details the demonstration process. Section V explains how programmes are constructed, whereas Section VI shows how the user's intention is captured. Experiments are shown in Section VII. Section VIII analyses our system and finally Section IX summarises this paper and proposes future work.

II. Literature Survey

A task is usually considered as a sequence of low level skills, such as grasping or moving to a given position. For example, a task can be pouring the content from one container to another [10] or laying a table [3]. The sequence of skills can be represented using STRIPS-like pre and post-conditions, such as in [1] and [10]. Demonstrations can be internally represented as effects on the environment, trajectories, operations and object positions [2].

The demonstration can be performed by the user themselves, while the robot gathers data through powerful sensors such as data-gloves [10], cameras or haptic devices. The robot can also be controlled through an external master system such as keyboards, graphical interfaces or teach pendants and experiences the demonstration through its internal and external sensors [4], [10].

Correctly inferring the user intention (see section I) has been recognised as a key problem [8], [9]. This is a difficult process, especially when the user only performs one demonstration. In those systems, the only demonstration given is inherently specific to the environment in which it was performed. A possible solution is to give multiple demonstrations of the same task ([10] and [4]); but it could possibly annoy the user [1].

However, the responsibility to determine what is relevant or irrelevant in a demonstration should not be left entirely to the imitator [4]. Interactive PbD systems are more able to avoid errors and generate the programmes that will best perform the tasks as expected by the user [11]. [11] and [8] use 3D-icons to help the user specifying their intention in a 3D simulation window. In [1], after a demonstration the user is asked to check whether the segmentation of a manipulation task is correct and has the possibility to modify the generated programme. The newly acquired task can be simulated in a 3D interface before being accepted

Multi modal interfaces, which interact with the user through several input modalities such as speech, gesture, vision, or contact with the robot, allow a more intuitive user interaction [9], [4]. In [9], speech and hand gestures are used to provide feedback both during and after the demonstration of a task with a vacuum cleaning robot. In [4], a robot learns and generalises a task over multiple demonstrations, with the additional help of vocal cues during the demonstrations and feedback cues after demonstrations, which allow the robot to refine the task representation.

III. Presentation of Spbd

The simple programming by demonstration system (spbd) implemented at the University of Auckland Robotics Lab has methods for demonstrating tasks and playing back previously recorded tasks. The typical usage of the system is shown in Fig. 1. The data from the demonstration phase is captured, possibly transformed, and stored in a given programme representation. This programme can then be further refined by the user manually selecting the important detail and discarding possible mistakes. During the playback phase, a programmed task is selected for execution, its programme representation is interpreted, and it is played back on the robot. Spbd uses the Player software interface to robot hardware [12].

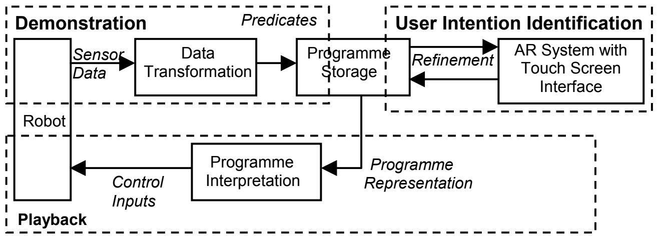


Fig. 1. Components of spbd involved in the three main phases of our PbD approach.

In addition to task recording and playback, spbd allows different data transformations and different programme representations to be defined (programme representation is discussed in Section V). Transformation of sensor data (which is often real-valued numerical data) into a more suitable form is important since robots' perceptions must be rendered using the augmented reality system and shown on the touch screen. Furthermore, the process of data transformation enables the removal of natural variation in the robot's sensors. Cognitive decisions can then be made based on accurate data during playback.

Spbd transforms the sensor data into logical predicates, which specify the properties of objects in the robot's environment, or a relation between a number of objects. For example, the distance from a robot to some object **x** in its environment can be encoded as a predicate **near(robot, x)**. Using this symbolic representation, the robot's perceptions can be described concisely and shown graphically and textually on the touch screen.

Methods for obtaining symbolic knowledge (such as predicates) from numerical sensor data usually involve sorting the incoming sensor data into predefined labeled categories [13], [14], and [15]. In our approach, models of significant sensor data patterns were designed manually, to make them more understandable. Each sensor model has a predicate associated with it. During demonstrations and playback, the defined models are continuously queried with sensor readings from the environment. If the readings match a particular model, a predicate corresponding to that model is produced. Each model contains parameters that guide the matching process. For example, the parameter of the model producing the **near(robot, x)** predicate can be a range of real-valued distances. The robustness of a given predicate to sensor variations can thus be tuned by adjusting the parameters of the model associated with that predicate.

IV. Demonstration and Robot Control

During a demonstration, the user interacts with the robot via a touch-screen interface. This interface shows a view of the robot's working space from an overhead camera and provides the user with basic trajectory planning commands. Virtual aids, such as a grid and a visualisation of the trajectories. overlay the camera image to assist the user (see Fig. 2).

To guide the robot through a demonstration, the user simply points on the touch screen to where the robot should go. The robot's trajectory can be modified by dragging the visualised trajectories at any time, even during execution.

The robot’s pose is regularly corrected using image processing to track a marker on its back (see Fig. 2). This camera tracker has been implemented by [16] and makes use of ARToolkitPlus [17].

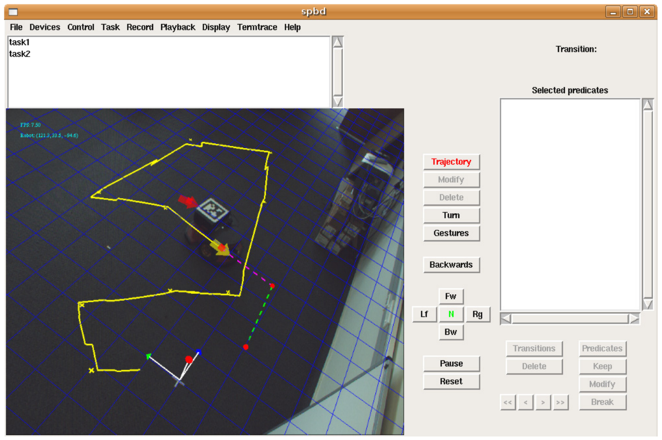
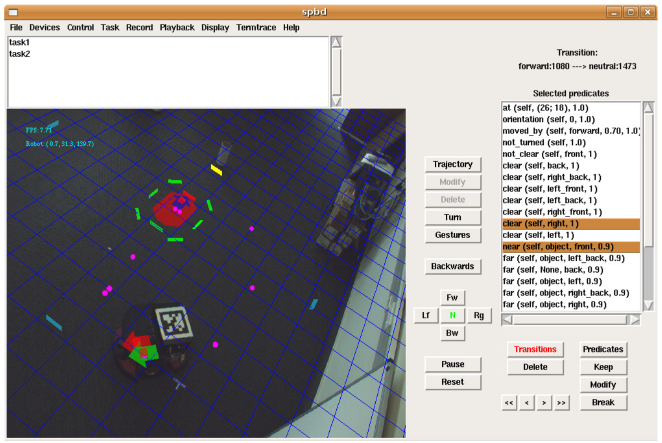


Fig. 2. The robot control interface. The trajectories given by the user are represented by augmented reality aids.



Transition:
forward:1080 ---> neutral:1473

Selected predicates

```
at (self, (26; 18), 1.0)
orientation (self, 0, 1.0)
moved_by (self, forward, 0.70, 1.0)
not_turned (self, 1.0)
not_clear (self, front, 1)
clear (self, back, 1)
clear (self, right_back, 1)
clear (self, left_front, 1)
clear (self, left_back, 1)
clear (self, right_front, 1)
clear (self, right, 1)
clear (self, left, 1)
near (self, object, front, 0.9)
far (self, object, left_back, 0.9)
far (self, None, back, 0.9)
far (self, object, left, 0.9)
far (self, object, right_back, 0.9)
far (self, object, right, 0.9)
```

Fig. 3. Intention capture phase. The purple dots represent different transitions. For the selected transition, the predicates are displayed textually and graphically.

V. Programme Construction

A programme is constructed as a sequence of states. The robot's motor states can be: not moving (neutral), moving forward, moving backward, turning left, and turning right. An assumption is made that significant events during demonstration occur when a change in the robot's state takes place. Hence, predicates that occur at the transition of the robot's states are recorded during demonstration.

When a demonstration concludes, the user is prompted to select which predicates best describe the experiences that the system should retain. For example, consider a task *move-forward-until-obstacle*, in Fig. 4. The robot moves forward from rest and continues until it reaches an obstacle. The sequence of states that the robot experiences is *neutral*, *forward*, and *neutral*, where the first *neutral* state indicates the initial rest state, and the second *neutral* state indicates the state that the robot enters when is senses the obstacle.

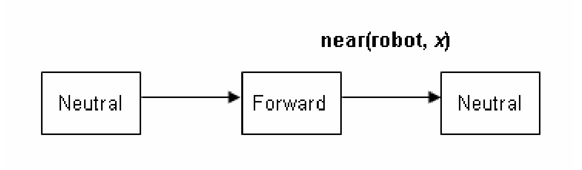


Fig. 4. States of the *move-forward-until-obstacle* task.

To complete the *move-forward-until-obstacle* task, predicates are added to transitions between its states. In spbd, the robot remains in its current state until all the predicates on the transition to its next state are satisfied. In the *move-forward-until-obstacle* task, the transition between the first *neutral* state and the *forward* state should contain no predicates, as the robot moves forward from rest unconditionally. The transition between the *forward* state and the second *neutral* state may contain the predicate **near(robot, x)**, indicating that the robot is near some obstacle **x**.

If the system knows how to achieve a predicate during playback, a higher-level command is issued instead of repeating the demonstrated motor states. For instance, if the user wants the robot to go from a point A to another point B, then they need to keep only the last transition (between the *forward* and *neutral* states), select the predicate **at(B)**, and discard the other, now irrelevant, transitions on the trajectory between A and B. In this instance, a higher-level **goto(B)** command will be used during playback.

Once several small tasks have been demonstrated, they can be manually combined into higher-level tasks by arranging them into programme structures. Currently, spbd allows three programme structures to be constructed: an "and" structure executes its component tasks in sequence, one after the other; an "or" structure selects one and only one of its component tasks for execution; and a "loop" structure, which repeatedly executes the component task.

After a demonstration, the user can also specify some predicates (or their negations) as *breaking conditions*. During the playback, spbd constantly checks if any of the specified breaking conditions is satisfied. If this is the case, the playback is interrupted. This also interrupts the playback of any parent task looping on this task.

VI. Capturing The User Intent

A. Motivation

Aside from providing an easy and intuitive robot control method, our interface aims to assist the user during the interaction phase, in which they have to give 'clues' to spbd, so that the system is able to understand the user intention and thus play back the task with a maximum fidelity. This phase mainly consists of selecting the relevant predicates for each transition recorded during the demonstration of the task. To help the user have a better understanding of the predicates generated by spbd and assist them during the selection process, the interface displays a visual representation of those transitions and predicates (see Fig. 3).

B. Work Flow

When the demonstration of a task ends, coloured dots pop up on the interface at the locations where all the different transitions happened during the recording. By graphically navigating through all the transitions on the touch screen, the user is visually "replaying" the task that has just been demonstrated and thus has a better understanding of what happened. Transitions are selected by touching the coloured dots. When a dot is selected, the predicates for the associated transition are displayed in two ways: textually in a listbox and graphically on top of the camera image. The graphical representation shows the robot's pose during the transition as well as the predicates representing the robot's perception of its environment at this time. The graphic representation of the predicates is quite basic: it consists of simple geometric shapes. Amongst the predicates represented are the robot's pose and nearby obstacles (see Fig. 3). The user selects relevant predicates by touching either the graphical presentation or the list in the listbox. Spbd allows the selected predicates to be modified. The user also has the option to discard the selected transition altogether. Finally, when several tasks have been demonstrated, the user can select and combine them using "and", "or", and "loop" structures.

VII. Experiments

Experiments have been carried out using both real robots (ActivRobots' 3-DX Pioneers) and Stage [12], a robot simulator. Stage and spbd ran on a 3GHz Pentium machine with 1GB of RAM. The following predicates were available in the example tasks:

at(x,y): the robot's location is (**x**,**y**)
orientation(angle): the robot's orientation is **angle**
moved by(xx): the robot has moved by **xx** meters since its last neutral state
turned by(xx): the robot has turned by an angle of **xx** degrees since its last neutral state
objectSpotted(xx, center): the object **xx** is straight in front of the robot
adjacent(front): an obstacle is adjacent close to the robot's front

A. Trajectory Following

This is a basic task in which the robot repeats a trajectory consisting of absolute and relative points. Absolute points typically correspond to commands such as **goto(*x*, *y*)** or **turnToOrientation(90°)**. Relative points can be assimilated to **move by(x cm)** or **turn by(xx°)** type commands.

As an example, consider the case where the robot has to follow a pattern representing the number "4," on the floor and then go to a particular location, say (0,0). This task is easily demonstrated just by pointing at the desired locations, e.g. the vertices of the pattern and the point (0,0). The pattern is an example of a relative trajectory, so to repeat this first part of the task the user has to select predicates reflecting this relative character (**moved by**, **turned by**). The second part of the task being an absolute trajectory, the user can discard all the transitions that might have been recorded between the end of the demonstration of the pattern and the moment the robot reached the location (0,0). Only the last transition, when the robot passes from a 'forward' state to a 'neutral' state has to be kept, and the predicate **at(0,0)** has to be selected (see Fig. 5).

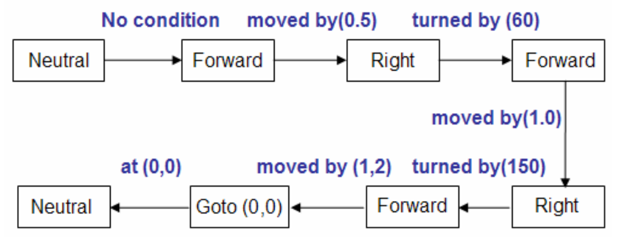


Fig. 5. Finite state machine representation of the *trajectory following* task. The boxes represent the commands that will be sent to the robot. The predicates in blue represent the conditions to go to the next instruction.

B. *Naïve Object Tracking*

For this task the robot follows a given object, by repeatedly turning to face the object and then moving forward. The robot also has to stop the whole execution of the task when it touches the object. To simulate object recognition, we used image processing to recognise ARToolkitPlus markers. To demonstrate this task, the user can first demonstrate *a tracking step* subtask, in which they just make the robot turn until it faces the object to track and then make it move forward a little, say 20 cm. Then, the user has to select the adequate predicates: the robot has to stop turning when it faces the object (**objectSpotted(xx, center)**) and then it stops moving forward after 20 cm (**moved by(0.20)**). To force the robot to stop the playback when it touches the object, the user can define the predicate **adjacent(front)** as a breaking condition (see Section VI). (Note that if this predicate has not been generated during the demonstration, the user has to create it themselves.) This *tracking step* subtask has then to be combined into a *naïve tracking* task using a "Loop" operator. Fig. 6 shows a representation of the *naïve tracking* task.

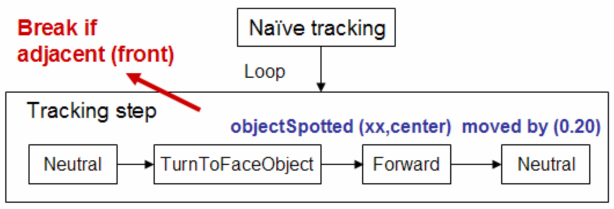


Fig. 6. Tree like representation of the *naïve tracking* task. The breaking condition is illustrated in red.

C. *Brave Patrolling*

In this task, the robot repeatedly patrols, i.e. follows a given trajectory, until it detects an intruder object. Then, the robot begins chasing the intruder. Consider the case where the robot has to patrol following a triangle pattern which absolute coordinates are (0,0), (10,0) and (5,5).

To demonstrate this task, the user can first demonstrate a *patrol step* subtask, in which the robot is shown how to patrol along the wanted path. After specifying the predicate **objectSpotted(xx, center)** as a breaking condition for this task, it is then combined into a *patrol* subtask using a "Loop" operator. Finally, the *patrol* and the *naive tracking* are combined together using an "And" operator. The result is that when the predicate **objectSpotted(xx, center)** is generated, during an iteration of the *patrol* loop, then both *patrol step* and *patrol* are interrupted, *and naïve tracking* is executed (see Fig. 7).

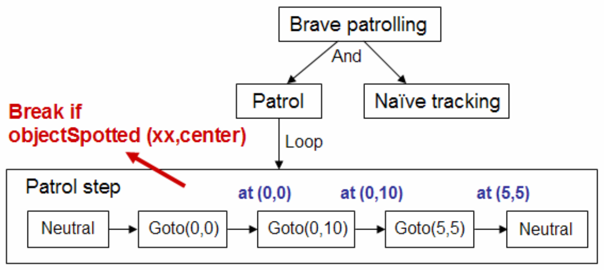


Fig. 7. Representation of the *brave patrolling* task.

VIII. ANALYSIS

A. *Strengths*

The representation of tasks is flexible and allows subtasks to be combined together, and allows the user to modify the task step by step without being involved in programming details. The gesture interface allows the user to interact with the representation to a greater degree than a traditional demonstration method, because the medium of demonstration is the same as the medium of interaction about intentions. The pre-playback run throughs are effective because of (a) the flexible, concrete representation given to the user, and (b) the gesture interface, enabling the user to effectively help build the programme step by step. The playback process is flexible in that it focuses on sensory data matches to make sure the playback actions are appropriate. The gesture interface itself is an easy way to specify way points, and to create a set of waypoints that are easy to see and can be manipulated afterwards. The gesture interface also leads naturally to the pre-playback interface.

B. *Weaknesses*

In spbd, the user must compose high level tasks by themselves. On one hand, this ensures an accurate control over execution during playback, especially for tasks that require different actions to be executed depending on the environment state. On the other hand, this can be inconvenient for the user. Maybe the segmentation should be done by the system and supervised by the user.

Currently it is not possible for the user to create a branch *during* one subtask, to another subtask. Branches can only be created at the *end* of a subtask; a breaking condition can be specified but not a branch for normal execution. What is needed in our future development is to provide an exception structure that the user can invoke graphically.

Another problem is that the demonstrated tasks cannot be reused with different parameters. Consider the example where the user wants to teach the robot to follow a square pattern. This can be done easily using our system; however this task can be repeated only to follow a square of the precisely the same dimensions as the one in the demonstration. To make the robot execute a square pattern with different dimensions, another task must be taught. A solution may be to introduce variables. After the demonstration of a task, the user could for instance indicate to the system that certain of the predicates' arguments are actually modifiable when the task is loaded into the system for playback. However end users who are not familiar with the concept of variables could perhaps be confused.

IX. Conclusion

In this paper, we introduced spbd, a PbD system for navigational tasks, as well as a touch screen gesture interface that uses augmented reality techniques to ease the user interaction with spbd. An important contribution of the work is that the user is able to easily interact with the task representation, to refine it before playback. Our system has successfully been used to demonstrate simple tasks by demonstration, without using a low level programming language. The main contribution of this work is a technique for capturing much more of the user's intentions than a PbD system without the gesture interface. The underlying representation helps capture the user intentions and we expect future work to include more reasoning about the intentions during playback.

Additional future work includes introducing variables and other features, such as obstacle avoidance during the playback and identification of the user's intentions with multiple demonstrations of the same task. A simulation environment, such as Stage, could also be used to allow the user to check safely whether the selected predicates reflect their intentions before accepting them. Scaling up the approach (e.g. involving manipulation tasks) remains an open issue.

Acknowledgment

Dr Toby Collett provided the essential tools for the augmented reality used in our work, and also provided much appreciated assistance during the development.

References

[1] M. Ehrenmann, R. Zöllner, O. Rogalla, and R. Dillmann. Programming service tasks in household environments by human demonstration. In Robot and Human Interactive Communication, Sept 2002. Proc. 11th IEEE Int. Workshop on, pp. 460–467.

[2] R. Dillmann, O. Rogalla, M. Ehrenmann, R. Zöllner, and M. Bordegoni (1999, Octobre). Learning robot behaviour and skills based on human demonstration and advice: the machine learning paradigm. In 9th Int. Symp. of Robotics Research (ISSR'99), pp. 229–238.

[3] R. Zöllner, M. Pardowitz, S. Knoop, and R. Dillmann (2005, 18-22 April). Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration. In Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on, Spain, pp. 1535–1540.

[4] M. N. Nicolescu and M. J. Matarić (2003). Natural methods for robot task learning: instructive demonstrations, generalization and practice. In AAMAS '03: Proc. of the second international joint conference on Autonomous agents and multiagent systems, New York, NY, USA, pp. 241–248. ACM Press.

[5] R. Voyles R. and P. Khosla (1999, 10-15 May). Gesture-based programming: a preliminary demonstration. In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on, Volume 1, Detroit, Michigan, USA, pp. 708–713.

[6] Y. Kuno, M. Yoshizaki, and A. Nakamura. Vision-Speech System Becoming Efficient and Friendly through Experience. 2003.

[7] A. Billard and R. Siegwart,, "Robot learning from demonstration", Robotics and Autonomous Systems, 2004, 47, 65-67

[8] H. Friedrich, R. Dillmann, and O. Rogalla (1999). Interactive robot programming based on human demonstration and advice. In H. I. Christensen, H. Bunke, and H. Noltemeier (Eds.), Sensor Based Intelligent Robots, Volume 1724 of LCNS, pp. 96–119. Springer.

[9] S. Iba, C. J. J. Paredis, and P. K. Khosla (2005). Interactive multimodal robot programming. The International Journal of Robotics Research 24 (1), 83–104.

[10] K. Ogawara, J. Takamatsu, H. Kimura, and K. Ikeuchi (2002, 11-15 May). Generation of a task model by integrating multiple observations of human demonstrations. In Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE Int. Conf.on, Vol. 2, pp. 1545–1550

[11] H. Friedrich, H. Hofmann, and R. Dillmann (1997, 10-11 July). 3d-icon based user interaction for robot programming by demonstration. In Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proc., 1997 IEEE Int. Symposium on, pp. 240–245.

[12] B. P. Gerkey, R. T. Vaughan, and A. Howard. "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems". In Proc. Int. Conf. on Advanced Robotics, pages 317–323, 2003.

[13] M. T. Rosenstein and P. R. Cohen, "Concepts From Time Series". In Proc. of the 15th National Conf. on Artificial Intelligence, pp 739–745, Menlo Park, CA, USA, 1998. AAAI.

[14] T. Oates, M. D. Schmill, and P. R. Cohen, "Identifying Qualitatively Different Experiences: Experiments with a Mobile Robot". In Proc. of the 16th Int. Joint Conf. on AI, 1999.

[15] L. Firoiu and P. Cohen "Abstracting from Robot Sensor Data using Hidden Markov Models". In Proc. of the 16th Int. Conf. on Machine Learning, pp 106–114, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[16] T. Collett and B. MacDonald. "Augmented reality visualisation for player". In Proc. Int. Conf. on Advanced Robotics, pp. 3954–3959, Orlando, May 15–19 2006.

[17] D. Wagner and D. Schmalstieg. ARToolKitPlus for pose tracking on mobile devices. In M. G. H. Grabner (Ed.), Proc. of 12th Computer Vision Winter Workshop (CVWW07), St. Lambrecht, Austria, 2007.