Preprint of: Robotics and Autonomous Systems, 38 (3-4): 171-181(2002)

Mobile Robot Programming Using Natural Language.

Stanislao Lauria, Guido Bugmann[1], Theocharis Kyriacou, Ewan Klein*

Centre for Neural and Adaptive Systems, School of Computing, University of Plymouth
Drake Circus, Plymouth PL4 8AA, United Kingdom.

*Institute for Communicating and Collaborative Systems, Division of Informatics, University of Edinburgh, 2 Buccleuch Place, Edinburgh EH8 9LW, Scotland, United Kingdom.

http://www.tech.plym.ac.uk/soc/staff/guidbugm/ibl/index.html

KEYWORDS: Natural Language, Human-robot dialogue, mobile robots learning, corpus collection, route-description.

Abstract
How will naive users program domestic robots? This paper describes the design of a practical system that uses natural language to teach a vision-based robot how to navigate in a miniature town. To enable unconstrained speech the robot is provided with a set of primitive procedures derived from a corpus of route instructions. When the user refers to a route that is not known to the robot, the system will learn it by combining primitives as instructed by the user. This paper describes the components of the Instruction Based Learning architecture and discusses issues of knowledge representation, the selection of primitives and the conversion of natural language into robot-understandable procedures.

1 Introduction

Intelligent robots must be capable of action in reasonably complicated domains with some degree of autonomy. This requires adaptivity to a dynamic environment, ability to plan and also speed of execution. In the case of helper robots, or domestic robots, the ability to adapt to the special needs of their users is crucial. The problem addressed here is one of how a user could instruct the robot to perform tasks which manufacturers cannot completely program in advance. In such case the system would not work at all if it cannot learn.
Such learning requires interaction and collaboration between the user and the robot. But, as most users are computer-language-naïve, they cannot personalise their robot using standard programming methods. Indirect methods, such as learning by reinforcement or learning by imitation, are also not appropriate for acquiring user-specific knowledge. For instance, learning by reinforcement is a lengthy process that is best used for refining low-level motor controls, but becomes impractical for complex tasks. Further, both methods do not readily generate knowledge representations that the user can interrogate.
Instruction-Based Learning (IBL), which uses unconstrained speech, has several potential advantages. Natural language can express rules and sequences of commands in a very concise way. Natural language uses symbols and syntactic rules and is well suited to interact with robot knowledge represented at the symbolic level. It has been shown that learning in robots is much more effective if it operates at the symbolic level [2]. This is to be contrasted with the much slower learning at the level of direct sensory-motor associations.
Chunking, sequencing and repair are the aspects, related to natural language interactions, shaping the design of IBL systems discussed here. Chunking is a principle that applies to the communication of information. Chunking is meant here as the human characteristic to divide, during explanations, tasks into sub-tasks so that all information should be presented in small 'basic' units of actions. As shown in [12], chunking is done spontaneously by humans and we expect that conversions from natural language instruction to robot program will be facilitated if the robot knows a set of primitive procedures corresponding to the action-chunks natural to the user.
Regarding repair, natural language explanations are notoriously underspecified, and the robot must be able to verify the consistency of the acquired program. For example, in a sequence of instructions given by the user, the final state of an action may not correspond to the expected state for the next action. In this case, the system

[1] To whom correspondence should be addressed.

would not be able to perform its task due to a missing chunk. For this reason, it is necessary to define a proper internal knowledge representation allowing the system to detect the missing information. In this way, the system would be able to make predictions about future events so that the problem can be solved while the system is still interacting with the user.
The system not only has to pay attention to user knowledge and dialogue goals, but it also has to adapt its dialogue behaviour to current limitations of the user's cognitive processing capabilities. Assistance is then expected from the system, so that the interaction may naturally flow over the course of several dialogue turns. Finally, a dialogue manager should take care of identifying, and recovering from, speech recognition and understanding errors.
This paper describes initial steps and considerations towards a practical realisation of an IBL system. The experimental environment is that of a miniature town in which a robot provided with video camera executes route instructions. The robot has a set of pre-programmed sensory-motor action primitives, such as "turn left" or "follow the road". The task of the user is to teach the robot new routes by combining action primitives. That task should reveal all the constraints described above, and enable testing of the developed methodology.
The closer the correspondence between primitives and chunks expressing the very basic actions (such as "turn left") is, the less difficult the learning is, since, in this way, the number of repair dialogue between the user and system is kept to the minimum. For this reason, it is necessary to select these primitives that corresponds as closely as possible to the action expressed in the chunks (see section 4).
A complete IBL requires several steps to transform a spoken chunk into a robot action (Table 1). First, the system must be able to convert speech into text. After that, some syntactic parsing and semantic analysis is carried out. Then at the functional mapping level, the system must be able to transform the user utterance into internal symbols that the robot can understand. By understanding we mean here that there is a correspondence between symbols and actions or real-world objects. In this way, the appropriate procedure can be called to act on the sensors and motors according to the user intentions.

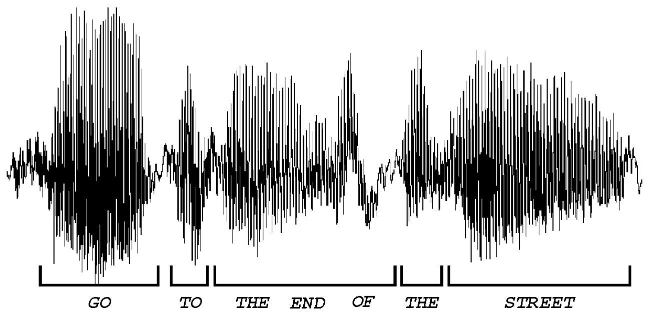
Analysis			Repair
↓	Speech recognition		↑
↓	Tagging	**Go**/VB **to**/TO **the**/DT **end**/NN **of**/IN **the**/DT **street**/NN	↑
	Syntactic Parsing	[VG go] [to to] [NG the end of the street]	
↓	Semantic Analysis	Robot (x), end_of_street (y), request_go (x, y)	↑
↓	Functional Mapping	Goto("end_of_street")	↑
	Robot program	Until found(end_of_street) follow_the_road()	

Table 1. From speech to action. *The various steps involved in the transformation of a user command into the corresponding action are shown here.*

Section 2 clarifies how symbol-level description and low-level sensory motor action procedures are integrated. The proposed representation of procedural knowledge is also described. In section 3 the system architecture is described.
The problems of considering the appropriate selection of action primitives is described in section 4 by analyzing recorded route instructions, and establishing a list of actions that are natural to users. The results of this

investigation are also discussed. These implications and other findings are discussed in section 5, along with the question of how the proposed system compares to other approaches. The conclusion follows in section 6.

2 The IBL model

2.1 Symbolic learning

The learning process is based on predefined initial knowledge. This "innate" knowledge consists of primitive sensori-motor procedures with names, such as "turn left", "follow the road" (the choice of these primitives is explained in sections 2.3 and 4). The name is what we call here a "symbol", and the piece of computer program that controls the execution of the corresponding procedure is called the "action" (Figure 1A). As each symbol is associated with an action, it is said to be "grounded".

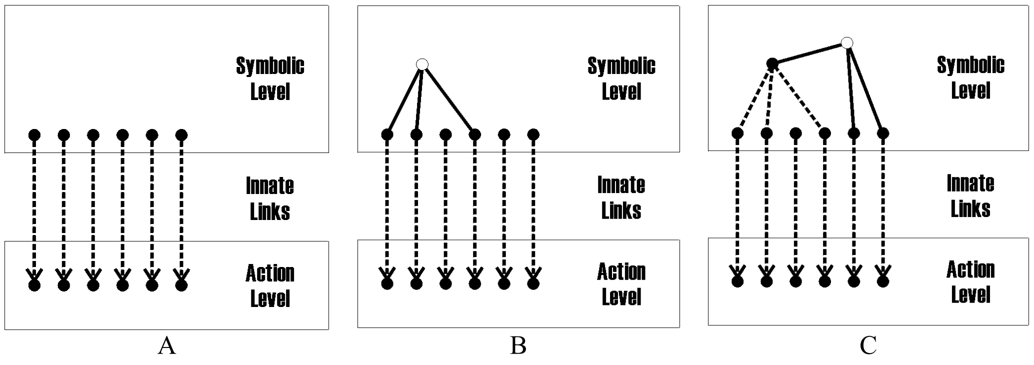


Figure 1. Symbolic learning. *(A) is a schematic representation of the initial system, comprising symbols associated with pre-programmed (innate) primitive action procedures. In (B) the user has defined a new procedure (open circle) as a combination of symbols. The new symbol is grounded because it is a construct of grounded symbols. In (C), the user has defined a new procedure that combines a procedure previously defined by himself with primitive action procedures.*

When a user explains a new procedure to the robot, say a route from A to B that involves a number of primitive actions, the IBL system, on the one hand, creates a new name for the procedure, and, on the other hand, writes a new piece of program code that executes that procedure and links the code with the name (see section 2.2 for details). The code refers to primitive actions by name. It does not duplicate the low-level code defining these primitives. For that reason, the new program can be seen as a combination of symbols rather than a combination of actions (figure 1B). As all new procedures are constructed from grounded primitives, they become also grounded by inheritance and are "understandable" by the system when referred to in natural language.
When explaining a new procedure, the user can also refer to old procedures previously defined by himself. In that way the complexity of the robot's symbolic knowledge increases (fig. 1C).

2.2 Knowledge representation

The internal representation needs to support three functions: (i) formal modeling of NL route descriptions; (ii) internal route planning for determining whether a given route description is sufficiently specified; and (iii) the generation of procedures for navigation at execution time. These three functions require different representations that will be described in turn.

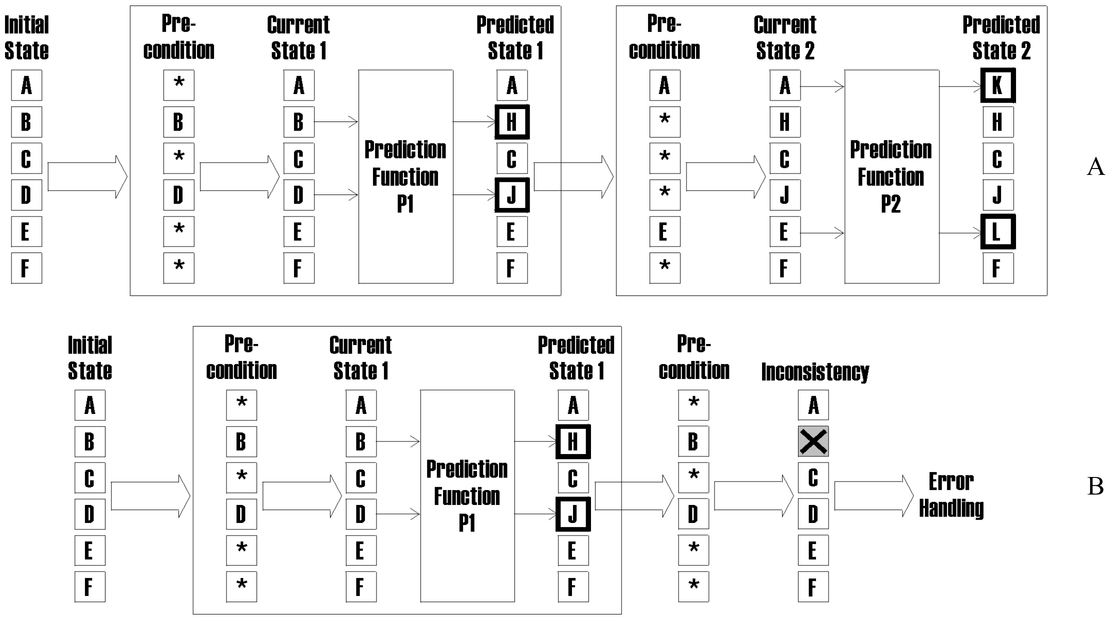


Figure 2. Route instruction verification. ***(A)*** *For each procedure there is a prediction function that transforms a state vector into its future value. The function first determines if the input state satisfied the minimal criteria ("pre-condition") to enable the procedure to be executed. An action is executable only if selected elements of the state vector have required values. If this is the case, the next state is predicted and processed by the prediction function associated with the next procedure in the instruction. Each action modifies certain components of the state vector, and leaves the other unchanged.* ***(B)*** *If the predicted state produced by one procedure does not allow the next procedure to be executed, an error handling process is initiated. (Note: the "initial state" in the text corresponds to the "current state" in the figure).*

(i) The utterances of the user are represented using the Discourse Representation Structure (DRS) [9]. This is then translated into symbols representing procedures or is used to initiate internal functions such as execution of a command or learning of a series of commands (section 3).
(ii) When the user describes a route as a sequence of actions, it is important for the robot to verify if this sequence is executable. The approach proposed here associate each procedure with a triplet SiAijSj with properties similar to productions in SOAR [8]. The state Si is the initial state in which the action Aij can take place. It is the pre-condition for action Aij. The state Sj is the final state, resulting of the action of Aij applied to the initial state (figure 2 clarifies the difference between "initial state" and "pre-condition"). For a sequence of actions to be realisable, the final state of one action must be compatible with the pre-condition of the next one. To enable this verification, the robot must be able to "imagine" the consequence of an action. For that purpose, a PREDICTION function is associated with each primitive action, and with each newly created procedure. Figure 2 illustrates the use of the prediction function during verification of the consistency of the sequence of instructions from the user. It should be noted that this process also helps detecting some of the errors in natural language processing.

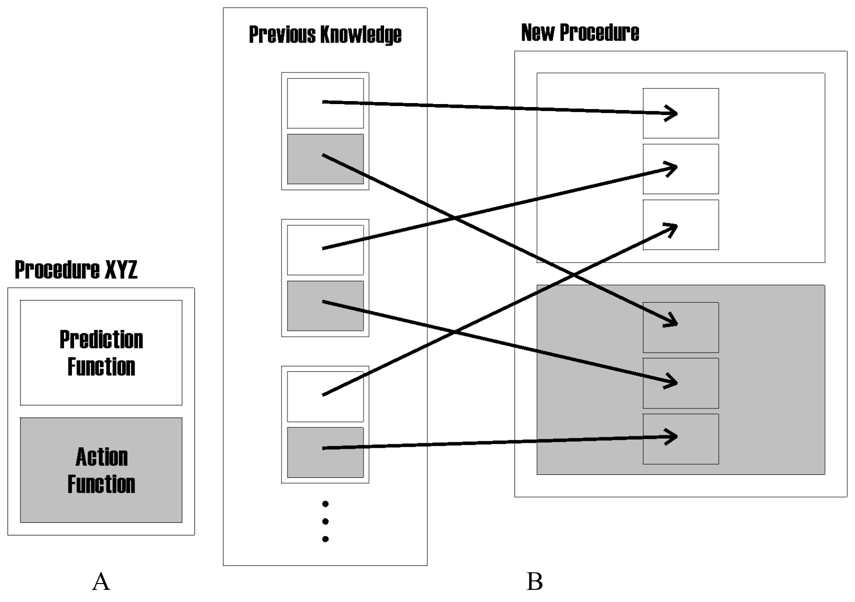


Figure 3. Procedural knowledge representation. ***(A)*** *A procedure file contains an ACTION function that causes the physical displacement of the robot, and a PREDICTION function that calculates the future state of the robot resulting from the action. The ACTION is used during execution of a command, and the PREDICTION is used for consistency checking during the learning process.* ***(B)*** *An instruction by the user results in a "New Procedure" file being written. In this file, the actions components of the requested primitive procedures are combined (in the form of function calls) to create the new ACTION function, and the prediction components are combined to create the new PREDICTION function. This includes an additional procedure-specific pre-condition.*

(iii) When a robot executes a command, it executes a piece of program code that contains the sequence of primitive procedures to be executed. Thus, a key part of IBL is the generation of a program code. This is enabled by the use of a scripting language (section 3). This program is called the ACTION function. Both ACTION and PREDICTION functions are physically located in the same file that contains all information specific to a procedure. This is schematised in figure 3.

2.3 Sensory-Motor primitives

Sensory-motor primitives are defined as action-chunks that users usually refer to in unconstrained speech. These could be low-level procedures referring, for example, to robot wheel turns, distance vectors etc. or they can be high-level procedures like for example "turn left after the church" or "take the second exit off the roundabout". In natural language route instructions, low-level specification of actions generally does not appear. Instead, higher-level procedures are mentioned which will have to be pre-programmed and thus become the sensory-motor primitives in this context.
In this project we have defined primitives as procedures which take parameters. For example the action "take the second right after the post-office", maps to the primitive *turn* with parameters *second*, *right*, *after* and *post-office*. It is then a matter of correctly mapping user utterances to the right primitives and passing the right parameters to them.

3 System Architecture

The architecture is comprised of several functional processing modules (figure 4). These are divided into two major units: the Dialogue Manager (DM) and the Robot Manager (RM).

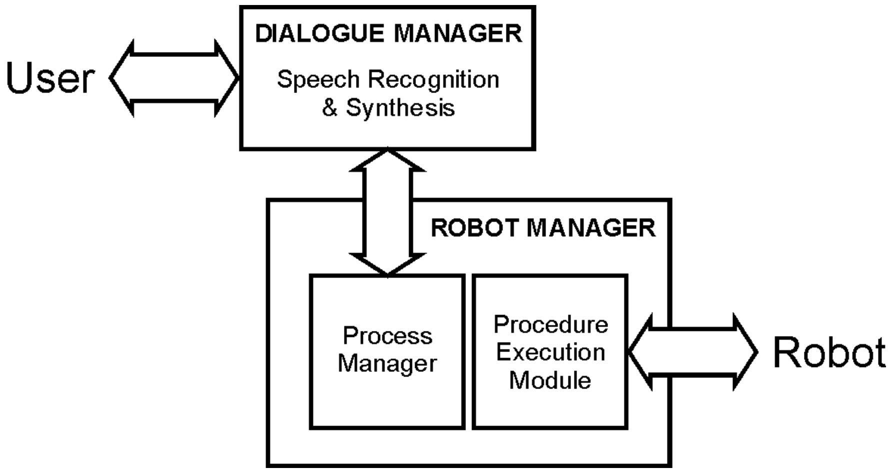


Figure 4. *IBL system's architecture (see text for description).*

The Dialogue Manager is a bi-directional interface between the Robot Manager and the user, either converting speech input into a DRS semantic representation [16], or converting requests from the Robot Manager into dialogues with the user. Its components are described in [9].
The RM deals with the DM's output and also with the learning and execution of the commands from the user. As shown in figure 4 the RM includes two modules: the Process Manager (PM) and the Procedure Execution Module (PEM). The PEM is responsible for carrying out the commands by the user. It executes procedures called by the Process Manager module.
The PM transforms the semantic representation produced by the DM into the internal language of the robot that includes learning and execution functions. Mapping symbols from the DRS onto the corresponding entities in the internal representation allows converting user requests into robot procedures with the right parameters. When successful, the PM starts the appropriate process either to execute the requested task by a call to the PEM or alternatively to build a new user-defined procedure explained by the user. When such mapping is not successful the RM must inform the DM, which starts a clarification dialogue with the user. Such mapping process is supported by a new specification language that expresses the relations between the symbols used in the DRS and the corresponding primitives. Thus to introduce new primitives, it is sufficient for the designer of an IBL systemto change the grammar of the specification language without having to recompile any of the RM modules.
The Robot Manager is written using the Python[2] scripting language. C language extensions to Python are also used in case where speed is a constraint (for example in vision routines). An important feature of scripting languages such as Python is their ability to write their own code. For instance, a route instruction given by the user will be saved by the Robot Manager as a Python script that then becomes part of the procedure set available to the robot for execution or future learning.
It is important that the RM must listen to the DM and try to process its output but at the same time it should be able to send messages to the DM. The DM and the RM are designed as two different processes based on asynchronous communication protocols. These processes run concurrently on different processors. In this way, the system can handle, at the same time, both the dialogue aspects of an incoming request from the user (i.e. speech recognition and semantic analysis) and the execution of a previous user request (i.e. check if the request is in the system knowledge domain, and execute vision-based navigation procedures).
Two aspects are essential with this concurrent-processes approach. The first is to define an appropriate communication protocol between the two processes. The second is to define an appropriate architecture for the RM and DM allowing the two processes to both communicate with each other while performing other tasks. At present the use of context-tagged messages within a communication based on the Open Agent Architecture (OAA) [13] is evaluated.
Moreover, the system must also dynamically adapt itself to new user requests or to new internal changes, by being able to temporarily suspend or permanently interrupt some previous activity. For example the user may want to prevent the robot crashing against a wall and must therefore be able to stop the robot while the robot is

[2] http://www.python.org

driving towards the wall. Hence, the importance of a concurrent approach where the system constantly listens to the user while performing other tasks and at the same time is able to adjust the task if necessary.

4 Corpus Collection and Data Analysis

To evaluate the potential and limitations of IBL, a real-world instructions task is used, that is simple enough to be realisable, and generic enough to warrant conclusions that hold also for other task domains. A simple route scenario has been selected, using real speech input and a robot using vision to execute the instructed route (see 4.1 below for more details). The first task in the project is to define the innate actions and symbols in the route instruction domain. For this reason, a corpus of route descriptions has been collected from students and staff at the University of Plymouth. In section 4.2 and 4.3 corpus collection and data analysis are presented.

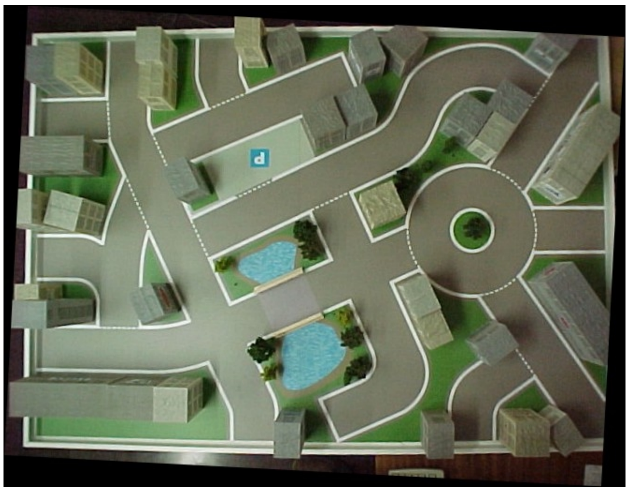


Figure 5. *Miniature town in which a robot will navigate according to route instructions given by users. Letters indicate the destinations and origins of various routes used in the experiment.*

4.1 Experimental Environment

The environment is a miniature town covering an area of size 170cm x 120cm (figure 5). The robot is a modified RobotFootball robot[3] with an 8cm x 8cm base (figure 6A). The robot carries a CCD colour TV camera[4] (628 (H) x 582 (V) pixels) and a TV VHF transmitter. Images are processed by a PC that acquires them via with a TV capture card[5] (an example of such image is shown in figure 6B). The PC then sends motion commands by FM radio to the robot. During corpus collection, the PC is also used to record instructions given by subjects.

[3] Provided by Merlin Systems (http://www.merlinsystemscorp.com/)
[4] Provided by Allthings Sales and Services (http://www.allthings.com.au/)
[5] TV Card: Hauppage WinTV GO

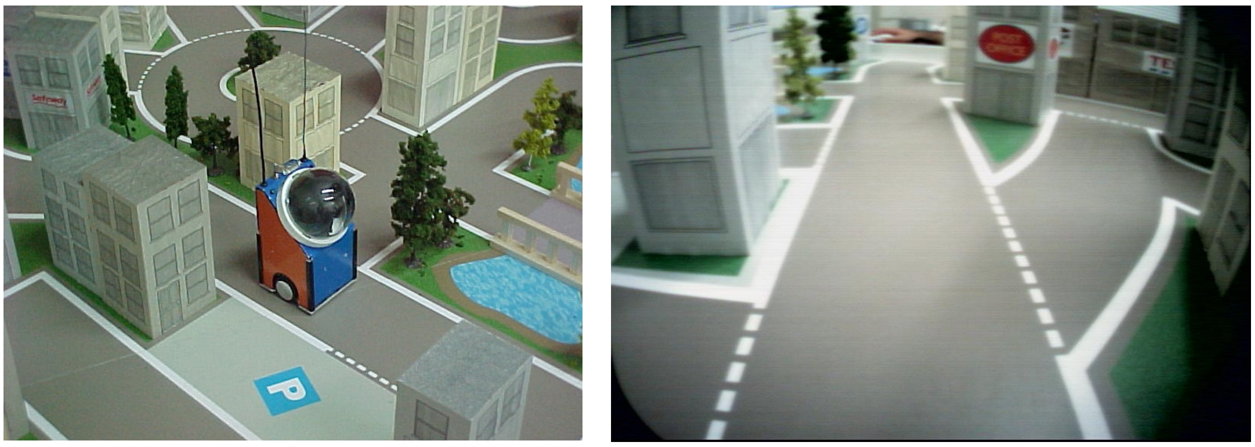


Figure 6 A. *Miniature robot (base 8cm x 8cm).* ***B****. View from the on-board colour camera*

The advantage of a miniature environment is the ability to build a complex route structure in the limited space of a laboratory. The design is as realistic as possible, to enable subjects to use expressions natural for the outdoor real-size environment. Buildings have signs taken from real life to indicate given shops or utilities such as the post-office. However, the environment lacks some elements such as traffic lights that may normally be used in route instructions. Hence the collected corpus is likely to be more restricted than for outdoor route instructions. The advantage of using a robot with a remote-brain architecture [7] is that the robot does not require huge on-board computing and hence can be small, fitting the dimensions of the environment.

4.2 Collection of a corpus of route instructions

To collect linguistic and functional data specific to route learning, 24 subjects were recorded as they gave route instructions for the robot in the environment. Subjects were divided into three groups of 8. The first two groups (A and B) used free flow speech, to provide a performance baseline. It was assumed that a robot that can understand these instructions as well as a human operator would represent the ideal standard. Subjects from group C were induced in producing shorter utterances by a remote operator taking notes.
The groups A and B were told that the robot was remote-controlled and that, at a later date, a human operator would use their instructions to drive the robot to its destination. It was specified that the human operator would be located in another room, seeing only the image from the wireless on-board video camera. This induced subjects to use a camera-centred point of view relevant for robot procedure primitives and to use expressions proper for human communication. Subjects were also told to reuse previously defined routes whenever possible, instead of re-explaining them in detail. Each subject had 6 routes to describe among which 3 were "short" and 3 were "long". The long routes included a short one, so that users could refer to the short one when describing the long one, instead of re-describing all segments of the short one. This was to reveal the type of expressions used by users to link taught procedures with primitive ones. Each subject described 6 routes having the same starting point and six different destinations. Starting points were changed after every two subjects. A total of 144 route descriptions were collected. For more details about collection and analysis of the corpus see [1]

4.3 Corpus Analysis: The functional vocabulary

The aim of the corpus analysis is to twofold. First, to define the vocabulary used by the users in this application, in order to tune the speech recognition system for an optimal performance in the task. Secondly, to establish a list of primitive procedures that users refer to in their instructions. The aim is to pre-program these procedures so that a direct translation from the natural language to grounded symbols can take place. In principle, if the robot does not know a primitive procedure, the user could teach it. Hereafter, we report on the functional analysis of the corpus. The reader interested in the task vocabulary can refer to [1]. The functional vocabulary is a list of primitive navigation procedures found in route descriptions.
The initial annotation of instructions in terms or procedures, as reported here, is somehow subjective, and influenced by two considerations. (i) The defined primitives will eventually be produced as C and Python Programs. It was hoped that only a few generic procedures would have to be written. Therefore, the corpus has been transcribed into rather general procedures characterised by several parameters (table 2). (ii) An important issue is knowledge representation. According to the SAS representation discussed in section 2.2, the

executability of primitives can only be evaluated if their initial and final states are defined. Subjects however rarely specified explicitly the starting point of an action and sometimes did not define the final state in the same utterance. Nevertheless, it was assumed that the system would be able to infer the missing information from the context. Therefore, procedures without initial or final state were considered to be complete, and were annotated as such. The specifications of primitive procedures are likely to evolve during the project.

	Count	**Primitive Procedures**
1	308	MOVE FORWARD UNTIL [(past |over |across) <landmark>] | [(half_way_of | end_of) street] | [after <number><landmark> [left | right]] | [road_bend]
2	183	TAKE THE [<number>] turn [(left | right)] | [(before | after | at) <landmark>]
3	147	<landmark> IS LOCATED [left | right |ahead] | [(at | next_to | left_of | right_of | in_front_of | past | behind | on | opposite | near) < landmark >] | [(half_way_of | end_of | beginning_of | across) street] | [between <landmark> and <landmark>] | [on <number> turning (left | right)]
4	62	GO (before | after | to) <landmark>
5	49	GO ROUND ROUNDABOUT [left | right] | [(after | before | at) <landmark>]
6	42	TAKE THE <number> EXIT [(before | after | at) <landmark>]
7	12	FOLLOW KNOWN ROUTE TO <landmark> UNTIL (before | after | at) <landmark>
8	4	TAKE ROADBEND (left | right)
9	4	STATIONARY TURN [left | right | around] | [at | from <landmark>]
10	2	CROSS ROAD
11	2	TAKE THE ROAD in_front
12	2	GO ROUND <landmark> TO [front | back | left_side | right_side]
13	1	PARK AT <location>
14	1	EXIT [car_park | park]

Table 2. *Primitive navigation procedures found in the route descriptions collected from groups A and C. Procedure 3 is used by most subjects to indicate the last leg of a route, when the goal is in sight.*

This analysis methodology differs slightly from the one in [4]. In our analysis, there are no statements describing landmarks, as these are made part of procedures specifications, and consequently there are also no actions without reference to landmarks. Even when a subject specified a non-terminated action, such as "keep going", it was classified as "MOVE FORWARD UNTIL", assuming that a termination point would be inferred from the next specified action. The list of actions found in the route descriptions of groups A and C is given in table 2. It has been shown in [9] that the number of distinct procedures is increasing with the number of sampled instructions, but at a rate much smaller than the number of distinct words. Here we discover on average one new procedure for every 38 route instructions, while with words, we discovered in average one new word for each instruction. New procedures typically are the least frequent in table 2.

5 Discussions

Teaching a route to a robot using natural language is an application of a more general instruction-based learning methodology. The corpus-based approach described here aims at providing users with the possibility of using unconstrained speech, whilst creating an efficient natural language processing system using a restricted lexicon. As mentioned in section 2.3, primitives are quite complex procedures. Section 4.3 describes how the primitives where extracted from a corpus recorded by a group of people, mostly students, from various fields of study. They spoke freely to the robot using human-like expressions and therefore the primitives extracted from what they said reflect the amount of "knowledge" naive users would expect the robot to have. The level of complexity of the primitives therefore depends, not only on the nature of the natural language application but also on its users and their expectations of the robot. If the subjects of our corpus were robot engineers, for example, and were told that the robot does not know how to move or turn prior to their route instructions they may have produced a different corpus from which different primitives would have been extracted.

An important finding in [9] was that functional vocabulary is not closed. Hence, at some point in a robot's life, the user may have to teach it new primitives. For that purpose, the robot would need to posses an additional set of low level primitives, which correspond to lower level robot actions. Examples of such primitive learning are found in [5] and [14]. With our approach, this would require the collection of a new corpus to determine the necessary additional primitive procedures. Another solution could lie in an appropriate dialogue management to suggest a reformulation of the instruction. It is expected that with the corpus-based method used here, the frequency of such "repair dialogues" will be minimised. An open question is the detection of new functions in the user's utterance, as the lexicon may not contain the required vocabulary.

The approach to robot control described may be seen as an attempt to integrate the good properties of Behaviour-based control and classical AI. Behaviour-based control is an effective method for designing low-level primitives that can cope with real-world uncertainties, and AI has developed effective tools for symbol manipulation and reasoning (for a more detailed discussion about hybrid systems see for example [10]). However, the system differs in several ways from both methods. Here, the corpus defines what symbols and primitives to use. Consequently, some of the primitives are rather complex functions, involving representations of the environment and planning. These primitives are not always compatible with the representation-less philosophy of behaviour-based systems. On the AI side, the system does not use the full range of reasoning capabilities offered by systems such as SOAR. There are no other aims in symbolic processing than verifying the consistency of instructions, and the construction of new procedure specifications.

Other previous work on verbal communication with robots has mainly focused on issuing commands, i.e. activating pre-programmed procedures using a limited vocabulary. Only a few research groups have considered learning, i.e. the stable and reusable acquisition of new procedural knowledge. [6] used textual input into a simulation of a manipulator with a discrete state and action space. [3] used voice input to teach displacements within a room and mathematical operations, but with no reusability. In [15] textual input was used to build a graph representation of spatial knowledge. This system was brittle due to place recognition from odometric data and use of IR sensors for reactive motion control. Knowledge acquisition was concurrent with navigation, not prior to it. Whereas in [11], the system could learn new actions through natural language dialogues but only while the robot was performing them (i.e. it could only learn a new route from A to B while it was actually moving from A to B and dialoguing with the user).

In the IBL system described here, learning operates purely at the symbolic level; hence it can be done prior to performance. The ability to predict future states enables to engage in a verification dialogue before execution errors occur. If environmental conditions change such that an instruction is not valid anymore, this can be detected from the mismatch between the expected result and the actual one. Learning however is not autonomous. The system requires interaction with a human user to learn new symbols and their meaning. This simplifies the design of the robot due to the transfer of part of the cognitive load to the user. Future experiment will reveal if this approach results in effective and socially acceptable helper robots.

The design of an IBL system requires, as expected, specialists in NL processing and speech recognition, as well as specialists in artificial vision and robot control. Here we found that significant work was also required in extracting from the semantic representation of the user's utterance the corresponding robot-executable procedures. It is hoped that this process will be simplified in the future by using the new specification language currently developed as part of the project.

6 Conclusions

In this paper, it was noted that domestic robots, which cannot learn from their users will be of limited use. The Instruction-Based Learning method (IBL) has been presented in the special case of route instructions.

A key task in an IBL system is the translation from Natural Language (NL) instructions to robot-understandable procedures. The corpus-based approach has been proposed here to optimise such translation. It defines a task domain specific lexicon and set of primitives. This results in the implementation of a constrained language and limited task capabilities. However, it is expected that within a given task domain this will maximise the use of spontaneous speech and NL conversion efficiency. Only 14 primitives have been, but these are complex robotics procedures, involving visual search and planning. We believe that this is required to ensure efficient communication with a naive user. But the set probably is not closed. In other words, users at some time are likely to refer to primitives for which there is not preprogrammed counterpart in the robot's repertoire. It is likely that the dialogue management will play a key role in handling such situations.

Acknowledgement: This work is supported by EPSRC grants GR/M90023 and GR/M90160. The authors are grateful to A. Cangelosi and K. Coventry for enlightening discussions.

References:

[1] Bugmann G., Lauria S., Kyriacou T., Klein E., Bos J. and Coventry K. (2001) "Using Verbal Instruction for Route Learning", Proc. of 3rd British Conf. on Auton. Mobile Robots and Autonom. Systems: Towards Intelligent Mobile Robots (TIMR'2001), Manchester, 5 April.

[2] Cangelosi A., Harnad S. (2001) The adaptive advantage of symbolic theft over sensorimotor toil: Grounding language in perceptual categories. Evolution Communication. (in press)

[3] Crangle C. and Suppes P. (1994) Language and Learning for Robots, CSLI Lecture notes No. 41, Centre for the Study of Language and Communication, Stanford, CA.

[4] Denis M. (1997) "The description of routes: A cognitive approach to the production of spatial discourse", CPC, 16:4, pp.409-458.

[5] FLAKEY: www.ai.sri.com/people/flakey/integration.html

[6]Huffman S.B. and Laird J.E. (1995) "Flexibly Instructable Agents", Journal of Artificial Intelligence Research, 3, pp. 271-324.

[7] Inaba M., Kagami S., Kanehiro F., Hoshino Y., Inoue H. (2000) "A platform for robotics research based on the remote-brained robot approach", International Journal of Robotics Research, 19:10, pp. 933-954.

[8] Laird J.E., Newell A. and Rosenbloom P.S. (1987) "Soar: An architecture for general Intelligence" Artificial Intelligence, 33:1, pp.1-64.

[9] Lauria S., Bugmann G., Kyriacou T., Bos J., Klein E. (2001) "Personal Robot Training via Natural-Language Instructions" , IEEE Intelligent Systems, 16:3, pp. 38-45.

[10] Malcom C. M. (1995), The SOMASS system: a hybrid symbolic and behavioured-based system to plan and execute assemblies by robot. In J. Hallam, et al. (Eds), Hybrid problems and Hybrid solutions pp 157-168. Oxford: ISO-press.

[11] Matsui T., Asoh H., Fry J.,et al..(1999) Integrated Natural Spoken Dialogue System of Jijo-2 Mobile Robot for Office Services, http://citeseer.nj.nec.com/matsui99integrated.html

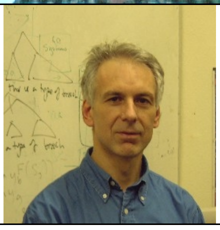
[12] Miller G. (1956)'The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity Processing Information'. The Psychol. Review, v. 63, p. 81-97

[13] Open Agent Architecture http://www.ai.sri.com/~oaa/

[14] Seabra Lopes, L. and A.J.S. Teixeira (2000) Human-Robot Interaction through Spoken Language Dialogue, Proceedings IEEE/RSJ International Conf. on Intelligent Robots and Systems, Japan.

[15] Torrance M.C. (1994) Natural Communication with Robots, MSc Thesis submitted to MIT Dept of Electrical Engin. and Comp. Science.

[16] Traum, D., J. Bos, R. Cooper, S. Larsson, I.Lewin, C. Matheson and M.Poesio (1999) A model of dialogue moves and information state revision. Trindi Report D2.1. www.ling.gu.se/projekt/trindi/publications.html

	Stanislao Lauria is currently a research fellow at the University of Plymouth. He received a Laurea in Physics from the Universita' di Napoli and a PhD degree in Cybernetics from the University of Reading. He has been research fellow at the University of Reading. His research interests are in the area of Neural Networks, Artificial Intelligence and robot vehicles. He can be contacted at stasha@soc.plym.ac.uk
	Guido Bugmann is a senior research fellow in the University of Plymouth's School of Computing, where he develops vision-based navigation systems for robots and investigates biological planning and spatial memory. He previously worked at the Swiss Federal Institute of Technology in Lausanne, NEC's Fundamental Research Laboratories in Japan and King's College London. He has three patents and more than 90 publications. Bugmann studied physics at the University of Geneva and received his PhD in physics at the Swiss Federal Institute of Technology in Lausanne. He is a member of the Swiss Physical Society, the Neuroscience Society, and the British Machine Vision Association. Contact him at the Centre for Neural and Adaptive Systems, School of Computing, University of Plymouth, Drake Circus, Plymouth PL4 8AA, UK; gbugmann@soc.plym.ac.uk.
	Theocharis Kyriacou is currently studying for a Ph.D. in Instruction Based Learning for Mobile Robots in the School of Computing of the University of Plymouth. He earned his B.Eng. (Honours) degree in Electronic Engineering Systems from the University of Sheffield in 2000.
	Ewan Klein is a reader in the Division of Informatics at the University of Edinburgh and director of Natural Language Research at Edify Corporation. His research interests include computational approaches to phonology, syntax, and semantics; multimodal interfaces; natural language specification of hardware design; and dialogue with intelligent systems. He received a BS in social and political science and a PhD in formal semantics from the University of Cambridge and an MS in general linguistics from Reading University