Intuitive Industrial Robot Programming Through Incremental Multimodal Language and Augmented Reality

Batu Akan, Afshin Ameri, Baran Çürüklü, Lars Asplund
Mälardalens Högskola, Box 883, 721 23 Västeras, Sweden
batu.akan@mdh.se, aai08001@studet.mdh.se, baran.curuklu@mdh.se, lars.asplund@mdh.se

Abstract— **Developing easy to use, intuitive interfaces is crucial to introduce robotic automation to many small medium sized enterprises (SMEs). Due to their continuously changing product lines, reprogramming costs exceed installation costs by a large margin. In addition, traditional programming methods for industrial robots is too complex for an inexperienced robot programmer, thus external assistance is often needed. In this paper a new incremental multimodal language, which uses augmented reality (AR) environment, is presented. The proposed language architecture makes it possible to manipulate, pick or place the objects in the scene. This approach shifts the focus of industrial robot programming from coordinate based programming paradigm, to object based programming scheme. This makes it possible for non-experts to program the robot in an intuitive way, without going through rigorous training in robot programming.**

I. INTRODUCTION

For several decades large companies producing mass-market products have used industrial robots in, e.g. machine tending, welding, and palletizing. Car manufacturers have led this trend, and thus have been an important market for industrial robots. Today industrial robots are perceived as being mass-market products on their own. Prices for industrial robots are pressed down, similar to other mass-market products. At the same time the technology behind them is getting better and better for every new generation. Despite this trend, in small medium enterprises (SMEs) robots are not commonly found. Even though the hardware cost of industrial robots has decreased, the integration and programming costs make them unfavorable for many SMEs. In order to make industrial robots more common within the SME sector, industrial robots should easily be (re)programmable by engineers that work in the production line at a manufacturing plant. Our goal is to give an industrial robot the ability to communicate with its human colleagues in the way that humans communicate with each other, thus making the programming of industrial robots more intuitive and easy. Consequently, a human-like interaction interface for robots will lead to a richer communication between humans and robots.

Traditional way of programming industrial robots is to use the teach pendant to sequentially move the robots tool center point (TCP) through the desired points. However the traditional programming method suffers in three ways: (i) Jogging an industrial robot with 6 degrees of freedom with a joystick with two degrees of freedom is very time consuming and cumbersome; (ii) the operator doesn't get any visual feedback of the process result before the program has been generated and executed by the robot; (iii) many iterations are needed for even the simplest task [17].

A view of the working environment is presented to user through a unified system. The system overlays visuals through augmented reality to the user and also it receives inputs and commands through a high level multi modal language. Such an approach would speed up the programming phase of the industrial robot and also would utilize the intuitive process knowledge of the operator.

Augmented reality (AR) is a term used for overlaying computer generated graphics, text and three dimensional (3D) models over real video stream. Virtual information is embedded into the real world, thereby augmenting the real scene with additional information. Augmented reality proved to be useful in several industrial cases, for visualizations. Olwal et al. [14] used 3D optical visualization techniques to visualize the process of a CNC machine to the operator. AR also provides great opportunities for Human Robot Interaction (HRI), and has been widely used in tele-robotics because AR allows the operator to work as if he is present at the remote working environment [7], [13], [9]. However AR can be very beneficial for programming industrial robots as well whether it is remote or local. Through wearable computers and head mounted displays it is possible to visualize and generate paths through a pointing device [17]. In their work Chong et al. [6] visually tracked marker to define collision-free paths for the industrial robot to follow. Once the path is generated a virtual robot simulates the behavior of the robot on the screen.

Communication between humans is a multimodal and incremental process [5]. Multi modality is believed to produce more reliable semantic meanings out of error-prone input modes, since the inputs contain complementary information, which can be used to remove vague information [16]. For example, if the speech recognition system has "blue object" and "brown object" as its two best hypotheses, the wrong hypothesis can be easily ruled out if there is support from vision system that there is no blue object in the scene.

It is also accepted that some means of communication are more error-prone to special type of information than the others [16]. For example, in an industrial environment,

saying ”weld this to that” while pointing at the two objects, is more reliable than saying “weld the 3cm-wide 5cm-long piece to the cylinder which is close to the red cube”. That’s because the speech channel is more error-prone when it comes to defining spatial information, while visual channel is more reliable in this case. Studies have shown that humans tend to use visual channel more often when spatial data is involved [15].

Multimodal nature of human communication and benefits of using such systems in HRI has made it an interesting field of research for years. Since the introduction of “Media Room” in Richard A. Bolts paper [4], many other systems have been implemented which are based on multimodal interaction with the user. Researchers have employed different methods in implementation of such systems [5], [8], [11]. All multimodal systems are common in the sense that they receive inputs from different modalities and combine the information to build a joint semantic meaning of the inputs.

On a higher lever of abstraction, speech modality alone has been used by Pires [18] to command an industrial robot through switching between preprogrammed asks. Also Marin et al. [13], [12] combined both augmented and virtual reality environments together with higher level voice commands to remote operate an industrial robot.

In this paper a context dependent multi modal language which is backed up by an augmented reality interface that enables the operator to interact with an industrial robot is proposed. The proposed language architecture makes it possible to manipulate, pick or place the objects in the scene. Such a language shifts the focus of industrial robot programming from coordinate based programming paradigm to object based programming scheme.

The rest of the paper is organized as follows: In Section II we give an overview explanation of proposed system architecture; augmented reality module, reasoning system and the multi modal language. In Section III we present test cases and Section IV provides discussion and conclusion.

II. Architecture

The hardware components of the proposed system is shown in Figure 1 and include a 6-Degree of Freedom (DoF) ABB IRB140 arm, a pneumatic-gripper, a robot controller, a desktop-PC, a desktop based display and a stereo camera. The software components behave as follows; The augmented reality module acts as a front and to the user together with synthesized speech, the incremental multimodal language module is responsible for receiving inputs and semantically analyzing them, and finally the reasoning system is responsible for turning the given commands into actions.

A. Augmented and Virtual Reality environments

The augmented reality (AR) module presented in this paper is an extension of the virtual reality (VR) and simulation environment described in [3]. The augmented reality environment consists of the virtual models of the physical entities that are around the robots operating range,

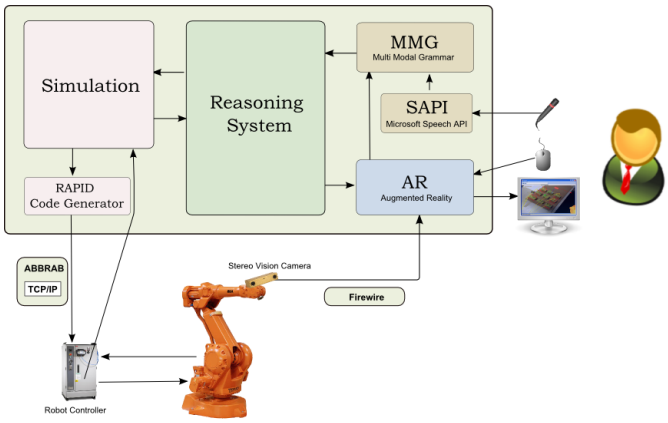

Fig. 1. Hardware ad software components of the system.

such as other robots, work-objects, tools, work-benches, etc. The objects and the robots are defined in local coordinate systems, therefore allowing the system to represent drop zones and gripping locations in object coordinates. Parent child relations are also kept for every object and their sub-objects. When an object is put into another object or over another object, it becomes its child object, and its coordinates and orientation are represented in its parents coordinate system, which enables the system to track the objects location. If the parent object is moved then all the child objects follow the parent object.

By putting the camera on the gripper of the robot and making the OpenGL camera follow the real camera, the virtual simulation environment can be used as an augmented fronted to the operator. As will be explained in the multimodal language section, the augmented reality environment allows the user to select the objects as well as drop-locations in the scene. Also by having the camera on top of the robot, the full working range of the robot can be supervised through a single camera. While reducing the installation costs, the single-camera solution also adds flexibility to the system. In this work the camera is used by the user to monitor the workbench. Based on the view from the camera the user gives instructions to the robot. Note that, the camera is not used for object recognition even though this possibility exists, since the primary objective of this work is to evaluate the multimodal language.

Augment reality can also be used as a mean for visual feedback to the user. It can be used to display text information about selected objects such as coordinates and orientations as well as display meta information associated with that object such as shape, color, weight, etc. Also the path planned by the robot is overlaid with the scene image from the camera. Figure 2 shows a screenshot of the augmented reality front end. On the left side of the image the AR part could be seen. Yellow lines shows the path to be taken by the robot, red cubes hints the user that there will be a gripper operation and the selected objected is highlighted with red color. On the right side of the image, system messages and automatically generated RAPID code

is shown.

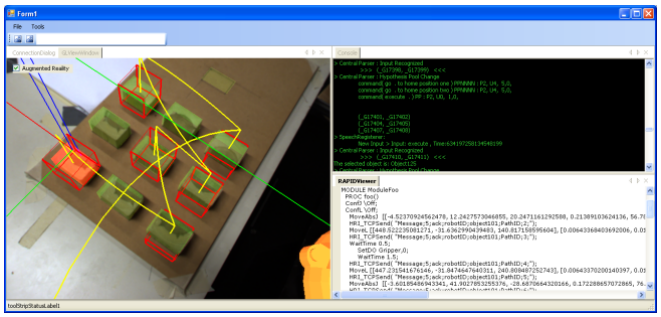

Fig. 2. Screenshot of the system. Yellow lines represents the path the to be taken by the robot and the red cubes represent a gripper action, whether grip or release.

Since the camera is mounted on the gripper of the robot, it is not always possible to visualize the paths to be taken by the robot, because the robot is not fully visible in the AR view. In such cases it is easier for the user to switch to virtual reality view, observe the path to be taken by the robot and switch back to AR view. Figure 3 shows two screenshots of the system, one in AR and the other in VR view.

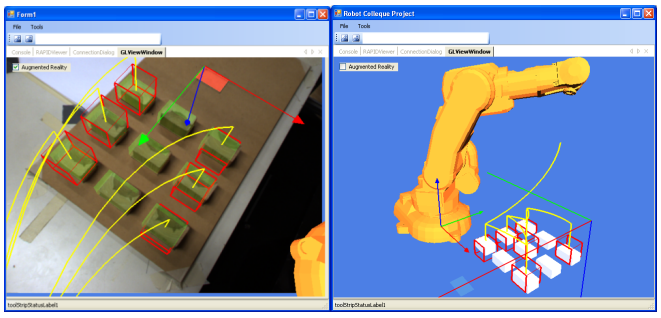

Fig. 3. Screenshots from Augmented and Virtual reality operating modes of the system.

B. Reasoning System

Reasoning system functions as the control unit of the robot. While interpreting the speech commands given by the user, it plans and executes necessary motor tasks to realize the commands. For this project a Prolog engine has been selected to implement the resoning system and the knowledge-base [19].

The reasoning system is tightly connected with the simulation environment. With the low-level movement and manipulator commands offered by the simulation environment the reasoning system is responsible for sequencing these functions to support higher level functionality which works at the level of objects instead of coordinates.

Upon calling the robot manipulation functions including basic movement and gripper functions, these calls are first tested for reachability along the path. If a point along the path is not reachable, or the orientation requested is not achievable or if the robot exceeds its joint limitations in any part of the path then the test fails. If the reachability test fails then the command is rejected from the queue, otherwise it is added to the job queue and the path of the tool center point is shown in the AR environment. When the “execute” function is called all the functions in the execution queue are translated in to RAPID [1] code and sent to the controller as a RAPID module where it is executed automatically.

As objects are inserted into the simulation environment,information about the objects are asserted into the knowledge-base. Only structural relations about the objects are kept in the knowledge-base, where as, position and orientation are not stored. Within this framework the objects around the robot are put into five sub categories: (i) manufacturing parts; (ii) movable tools; (iii) non-movable machines used by robots and/or humans; (iv) working benches; (v) miscellaneous objects [2]. Upon recognition, each object in the scene is assigned to one of the relevant categories or sub-categories derived from these main categories, while inheriting the default properties of this category. Additional properties can also be defined that is only relevant to this object or base properties can be overridden. Assigning the objects to a functional group is very useful in several ways. It limits the operations that are allowable on the objects, therefore eliminating the risks of the robot to perform irrelevant tasks, such as; picking up a workbench or trying to palletizing a CNC machine. It increases the overall robustness and security of the system.

On a higher level of abstraction, commands issued by the user, such as picking up an object is realized through a set of low-level movement functions. Given the position, the orientation of an object and the gripping pattern to be used in order to pickup that object, the necessary sequence of low-level movement functions for approaching, grasping and retracting is generated.

C. Multimodal language

The main characteristic of proposed system approach for language analysis is its incremental disposition. This means that the system will process different modality inputs as they are being received and builds up the syntactic and semantic representation of those inputs in an incremental fashion. It also means that the process will continue in higher levels of the reasoning system such as action planner and dialog manager. These parts will start to build up a plan and a dialog response (if needed) to incompletely perceived inputs. Our current system involves two different modalities however its design allows for integration of additional without the need for a change in the main system. This can be achieved by developing two external components for each modality. One of the components will be responsible for gathering input information and sending it to the central parser and the other one is responsible for parsing the inputs of the specific modality upon request from the central parser. These modality-specific parsers act on a unified multimodal input. Since users tend to employ multimodal commands mostly for spatial control [15], a speech/mouse system as the first step for a multimodal interface has been developed. In this

setup the user shares the view of the robots camera and can select objects or locations by clicking on the view while giving verbal commands. In a complex setup which may contain several objects, such commands make a more robust system compared to a system which only acts verbally [16]. Speech recognition is performed by Microsoft Speech API 5.0 (SAPI) in command and control mode. In this mode SAPI relies on external grammar definitions in XML format. Since the grammar needs to address all the modalities, a multimodal grammar definition language has been created named "3MG". Grammars written in this format can directly be used by the parsers or can be converted to modality-specific grammars when required. The latter has been the case only for SAPI, since it is an external component and its requirements should be fulfilled. For each modality there is a modality-specific parser, which collaborates with the main parser to build up the final syntactic outcome of the inputs. Grammar language and parsers are discussed in the coming sections, followed by a brief description of the approach for modality fusion.

1) Grammar definition language: The grammar definition language is a modified version of Johnstons [11], [10] unification based grammar. Our modifications give us the freedom of having as many modalities as needed and also help us to implement an easier interface for communicating with Prolog language interpreter which is used by semantic analysis and action planning systems. It also supports definition of optional and wildcard phrases. Optional phrases are special inputs which may be perceived but are not an integral part of a sentence. Wildcards have the same definition but they differ from optional phrases in the way that we have no hint on what they may be and therefore accept any input in their place. These are two features of SAPI which may help us in implementation of a more robust system. Figure 4 shows a sample of the grammar language which is named 3MG. Braces define optional phrases and colons define different modality inputs.

```
command(ObjetAnchor, Retval) --> <pickup(ObjetAnchor, Retval)>
{call(Retval)}.
command(ObjetAnchor, Retval) --> <putin(ObjetAnchor, Retval)>
{call(Retval)}.
command(ObjetAnchor, Retval) --> <execute>.

pickup(TragObj, Retval) --> pickup:singleObjectSelect
{TargObj=mouseReturnValue, Retval=pickup(TargObj)}.
pickup(TragObj, Retval) --> pickup <object(TargObj,Props)>
{Retval = (Props,pickup(TargObj))}.

object(TargObj, Retval) -->a wooden block {Retval = property
(TargObj,name,woodenblock)}.

putin(TargLoc, Retval) --> put it [here]:singleObjectSelect
{TargLoc=mouseReturnValue, Retval = (holding(CurObj), putin
(CurObj, TargLoc))}.
putin(TargLoc, Retval) --> put it in <object(TargLoc,Props)>
{Retval = (holding(CurObj), Props, putin(TargLoc, CurObj))}.

execute --> execute {execute}.
```

Fig. 4. A sample of the 3MG language.

2) Multimodal Parser: The multimodal parser is capable of receiving inputs from unlimited number of input modalities and extracts the syntactic meaning of them by using dedicated parsers for each modality. The fusion takes place at the same time with the help of multimodal grammar graph. Figure 5 shows a graphical representation of the subsystems involved.

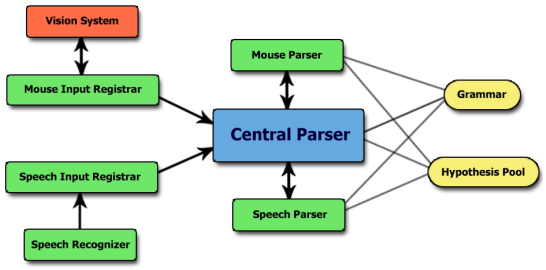

Fig. 5. Multimodal parser and its subsystems in our speech/mouse setup.

The central parser constructs a grammar graph based on the 3MG grammar definition file. The graph nodes contain required data for different modalities as well as additional information regarding optional and wildcard phrases. This design allows for in-time fusion of inputs from different modalities.

Central parser also holds a pool of hypotheses. All the hypotheses in the pool have multimodal representations and therefore can be used by modality-specific parsers if needed. The central parser checks hypotheses pool after each incremental parse session and removes invalid hypotheses from the pool. It also looks for any fully parsed sentence and terminates the parse session upon finding one.

Dedicated modality parsers are modality-specific parsers which have access to the hypotheses pool. When a new parse request is received from the central parser, they try to fit the new input in the current active hypotheses and update them accordingly.

Input registrars are other modality-specific objects which are responsible for gathering input data, packing and sending it to the central parser.

Both input registrars and modality-specific parsers are designed in a way that implementing new modalities can be done with ease and without requirement of any change in the core components of the system.

3) Modality Fusion: The central parser is also responsible for combining different modality inputs into a single final statement. The multimodal grammar graph has the key role in this process. As mentioned before, the grammar supports definition of optional phrases in it. This means that all the modality-specific parsers have the ability to skip over optional phrases if they are not perceived. The central parser takes advantage of this feature for modality fusion.

When a new input arrives in central parser, it checks the hypothesis pool to get a list of next possible phrases. If the next phrase is of the received input type the parse will continue in the modality-specific parser. But, in cases where the next phrase is of other modality than the one perceived, then the central parser makes that node as an optional node

and therefore allows the modality-specific parser to continue its parser with the newly received data.

This approach allows for later integration of skipped phrases without any strict limitations on time-span and order of inputs for different modalities. Limitations only apply when the inputs belong to previous utterance or have no effect on the final outcome.

Another benefit from this design is its flexibility. Users will not have to comply with the exact definition of the grammar and the system accepts inputs if they have pauses in their speech. For example in order to give a command for moving an object to a new place, any of the following is acceptable:

- speech:“move”, click on object, click on location
- speech:“move”, click on location, click on object
- speech:“move it”, click on object, click on location

III. Experimental Results

To demonstrate the usefulness of the proposed system, a test scenario has been created. The subject have been asked to pick and place wooden blocks according to the tasks given to them (Fig. 6).. The palette has been designed so that it has 10 drop-locations. Nine of them are organized as a 3x3 matrix, whereas the tenth drop-location is places further away from the others towards right. In each location the robot can place a wooden block. The wooden blocks themselves have a drop-location over their top so that the blocks can be stacked on top of each other. The experiments are carried out with four subjects. Two of them are expert robot programmers, and two are engineers not having previous experience in robot programming. Preceding each experiment the subjects received oral instructions on what they will perform; similar to how it would be in an industrial environment. The goal was also to keep the instructions to a minimum, and let the subject discover the system by testing it. Oral instructions were, thus, limited to 5 minutes for each subject.

A. Experiment 1

As an introductory warm up task, the subjects were asked to stack the wooden blocks to make a stack. In this experiment the subject would choose a drop-location, and move wooden blocks placed in other drop-locations to the target drop-location. Note that the target drop-location does not have to be empty prior to the experiment, i.e. it may be occupied by one or more wooden blocks as in Figure 7. The purpose of this task is to make the users familiar with the commands and the operation logic of the system. Once the users have gained experience, they easily moved the wooden blocks around, and build stacks with them. After roughly 5 minutes of testing the subjects felt that they can interact with the robot easily. In this initial experiment there were no differences between the performance of experienced robot programmers, and other subjects. Thus, all four subjects find the interaction the robot straightforward using the instructions.

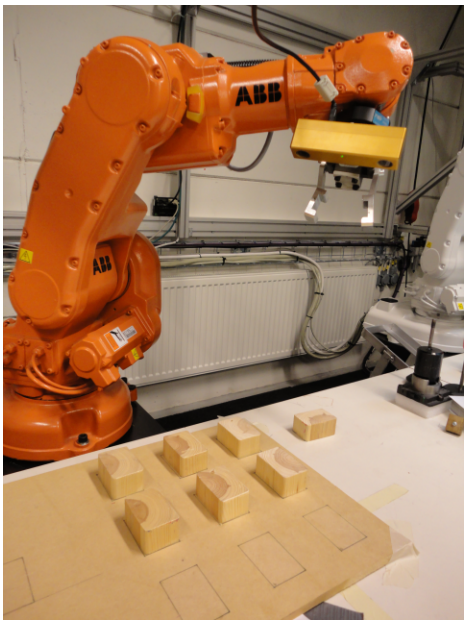

Fig. 6. Experimental setup consisting of.

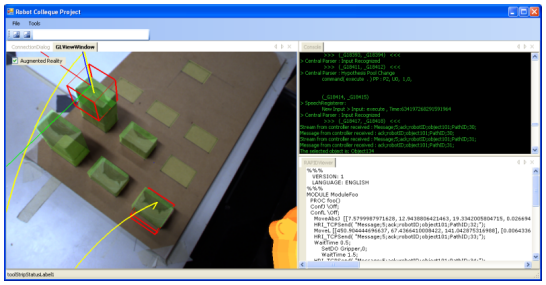

Fig. 7. Setup for experiment 2, where the users were asked to build stacks of wooden blocks.

B. Experiment 2

In this experiment the subjects perform a more complex task. They were asked to sort the numbered wooden blocks in sequential order starting from WoodenBlock 1 top-left to WoodenBlock 9 bottom-right as shown in Figure 8. Prior the experiment wooden blocks are placed randomly on top or each other, or on empty drop-locations. This task is more challenging than Experiment 1, since it requires a temporary storage area. In addition, the problem solving skills of the subjects plays an important role, since different strategies could be utilized for solving the problem. The simple and naive way would be to swap the blocks using the empty space on the wooden board one by one. However, this requires many code execution operations. A more efficient way to solve the problem would be to stack the wooden blocks in an ascending order on the empty space on the board while freeing up the target locations and then move them into the desired drop-locations. Using such a method it is possible to give multiple pick ad place commands in one execution batch. Which method to be followed depends on the experience and intuitive of the operator.

This simple task demonstrated how subjects just

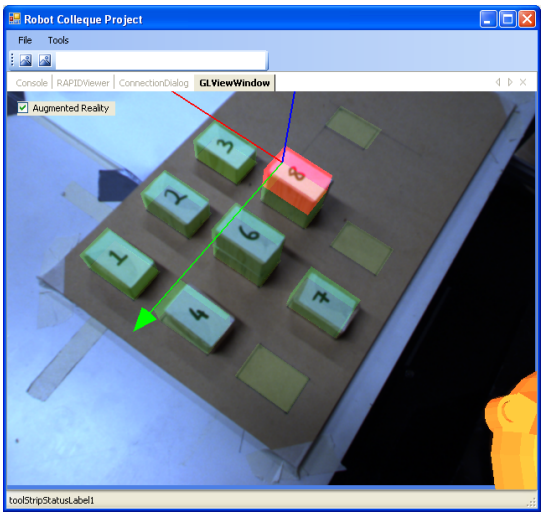

Fig. 8. Setup for experiment 2, where the users were asked to sort the numbered wooden blocks in an ascending order.

immediately after the start of the experiment constructed a mental model of the problem and solved it. Obviously, based on background the solutions differed considerably. One thing was common to all subjects though. They all could solve the problem as they planned initially, and without feeling that they are limited by the system.

IV. Conclusion

The proposed system demonstrates an alternative method for interaction between industrial robots and humans. Using natural means of communication is definitely an interesting alternative to well-established robot programming methods. These methods require considerable larger amount of time, and perhaps more importantly, a programming expert. In the experiment it is demonstrated that efficient collaboration between the robot and its human peer can help to clarify vague situations easily. Multimodal language seems to allow the system to understand the user's intentions in a faster and more robust way. There is no doubt that additional tests have to be carried out to fully test the system. It is, however, evident that the users with robot programming background do not feel that they are restricted due to the interaction method. Users with no experience in robot programming experience that they are as fast at robot programming as their expert counterparts.

This language also makes it possible for the user to assist the robot. This means that some of the computational load is shifted to the user. One example is when the user pointing out regions in robot's operational space through the clicking modality. There is, thus, no need for a lengthy description of the location. In addition the clicking modality also help to reduce the search space of the object recognition module (not demonstrated here). Thus, with multimodal communication both the robot and the operator can help each other in the way they are best at. This opens new possibilities for human-robot interaction in industrial environments, and hence is an evidence for the usefulness of this paradigm shift. In the near future work the multimodal language and the AR environment will be integrated even further. The system will also make use of the camera in object recognition as well as spatial reasoning.

References

[1] ABB Flexible Automation, 72168, Vasteras, SWEDEN. *RAPID Reference Manual 4.0*.

[2] B. Akan, B. Curuklu, and L. Asplund. Interacting with industrial robots through a multi-modal language and sensor systems. Seoul, Korea, Oct. 2008. International Symposium on Robotics.

[3] B. Akan, B. Curuklu, G. Spampinato, and L. Asplund. Object Selection using a Spatial Language for Flexible Assembly. Mallorca, Spain, Sept. 2009. Emerging Technologies on Factory Automation, ETFA09.

[4] R. A. Bolt. Put-that-there: Voice and Gesture at the Graphics Interface. *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, July 1980.

[5] T. Brick and M. Scheutz. Incremental natural language processing for HRI. *Proceeding of the ACM/IEEE international conference on Human-robot interaction - HRI '07*, page 263, 2007.

[6] J. Chong, S. Ong, a.Y.C. Nee, and K. Youcef-Youmi. Robot programming using augmented reality: An interactive method for planning collision-free paths. *Robotics and Computer-Integrated Manufacturing*, 25(3):689–701, June 2009.

[7] H. Fang, S. K. Ong, and A. Y.-C. Nee. *Robot Programming Using Augmented Reality*. IEEE, Sept. 2009.

[8] K.-y. Hsiao, S. Vosoughi, S. Tellex, R. Kubat, and D. Roy. Object schemas for responsive robotic language use. *Proceedings of the 3rd international conference on Human robot interaction - HRI '08*, page 233, 2008.

[9] C. A. Jara, F. A. Candelas, P. Gil, M. Fernández, and F. Torres. An augmented reality interface for training robotics through the web. *Communication*, pages 189–194, 2005.

[10] M. Johnston. Unification-based multimodal parsing. In *In COLING/ACL*, pages 624–630. Press, 1998.

[11] M. Johnston and S. Bangalore. Finite-state multimodal parsing and understanding. *Proceedings of the 18th conference on Computational linguistics* -, pages 369–375, 2000.

[12] R. Marin, P. Sanz, P. Nebot, and R. Wirz. A Multimodal Interface to Control a Robot Arm via the Web: A Case Study on Remote Programming. *IEEE Transactions on Industrial Electronics*, 52(6):1506–1520, Dec. 2005.

[13] R. Marin, P. Sanz, and J. Sanchez. A very high level interface to teleoperate a robot via Web including augmented reality. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, (May):2725–2730, 2002.

[14] A. Olwal, J. Gustafsson, and C. Lindfors. Spatial augmented reality on industrial CNC-machines. *Proceedings of SPIE*, 6804:680409–680409–9, 2008.

[15] S. Oviatt, A. DeAngeli, and K. Kuhn. Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction. *Proceedings of Conference on Human Factors in Computing Systems*, Mar. 1997.

[16] S. L. Oviatt. Ten myths of multimodal interaction. *In CACM*, volume42(11):pages74–81, 1999.

[17] T. Pettersen, J. Pretlove, C. Skourup, T. Engedal, and T. Lokstad. Augmented reality for programming industrial robots. *The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.*, pages 319–320, 2006.

[18] J. N. Pires. Robotics for small and medium enterprises : control and programming challenges. *Industrial Robot*, 2006.

[19] J. Wielemaker. An overview of the {SWI-Prolog} Programming Environment. In F. Mesnard and A. Serebenik, editors, *Proceedings of the 13th International Workshop on Logic Programming Environments*, pages 1–16, Heverlee, Belgium, Dec. 2003. Katholieke Universiteit Leuven.