Communications

A Natural Language Interface to a Robot Assembly System

MALLORY SELFRIDGE AND WALTER VANNOY

Abstract—**Natural language interaction with robots is an important goal since it promises access to robots by non-experts. Specifically, natural language interaction with robots would allow robots to be used easily and to be taught new skills. To produce such interaction requires, however, a choice among a variety of technologies in several areas. A working natural language interface to a robotic assembly system is described that allows a user to converse with the system about both vision and manipulation and to teach the system new vision knowledge and new assembly plans. The success of this approach suggest that its technologies are an appropriate choice for future real-world natural language interfaces to robot assembly systems.**

I. Introduction

Natural language interaction with robots is an important goal since it promises access to robots by non-experts. For example, a robot assembly system employing natural language could be controlled by interactions with an assembly expert rather than a roboticist, thus saving time and money. However, building a natural language interface to a robot assembly system is a difficult task. In particular, choices must be made among a number of different technologies within the areas of natural language processing, robot control, vision processing, learning, and overall system architecture. Since little prior research has been done on natural language interfaces to robot systems [17], the choice of technologies, and the demonstration that a given choice is appropriate, pose a significant research problem.

A natural language interface to a robot assembly system should meet a number of different goals. It must provide flexible communication between the robot system and the user to enable the user to control the system with little knowledge of the system's details. To accomplish this, it must be flexible enough to handle variant syntax, missing input words, handle mixed-initiative conversation and ellipsis. Provided these abilities, the users should be able to direct the vision processing and the manipulation of the system by natural language interaction. If the system does not know how to perform the desired action, it should ask the user for an explanation. If the system cannot answer a user's question, it should request additional information. If a natural language interface to a robot assembly system performs in a promising manner with respect to these goals, then this suggests that its choice of technologies is promising.

In order to achieve such goals in a realistic manner, five different areas must be specified: natural language processing, robot control, vision processing, learning, and overall system architecture. In each area, there are a number of different techniques available. First, the natural language component could be based on key words [16], semantic grammars [6], ATN processing [18], or conceptual analysis [2], [14]. Second, robot control could be achieved through robot programming [10], collision avoidance [7], or high-level planning [3]. Third, vision processing could be based on pattern recognition, binary vision, or intelligent vision techniques [1]. Fourth, the learning component could make use of a number of different techniques in machine learning [9]. Fifth, the components themselves could communicate via direct function calls, an overall executive, or some other architecture. The question is, which choice of technologies are appropriate for achieving the desired goals?

This paper describes a natural language interface to a robot system that achieves these goals and thus is relevant to the question of how such an interface should be designed. The interface described in this paper provides access to image processing, robotic manipulation, and learning. It provides the ability to question and direct these major subsystems without requiring the user to learn the system's details. That is, it allows the user to control low-level image processing and high-level image processing, ask high-level questions about the relationships between the objects in the image, and describe the visual appearance of unknown objects. It also allows the user to direct the assembly process and to teach the system how to assemble new subassemblies which can then be used in the overall assembly.

This interface is based on the use of conceptual analysis technology to perform natural language processing, the use of high-level planning to perform robot control, the use of intelligent vision processing, the use of learning-by-being-told, and a rule-based stack-oriented architecture for the overall system. Its performance suggests that this choice of technology is promising. Although this interface is clearly not appropriate for real-world application at this time, it allows the identification of research issues which must be addressed if real-world application is to be achieved.

Section II presents an example of the system's performance. Section III describes the system in detail, including its subsystems and their interaction, and Section IV discusses future research directions and conclusions.

II. An Example

The robot assembly system, described more fully in the next section, consists of two small manipulators that share the same workspace and two TV cameras directed at that workspace. The output of the cameras goes to digitizers and a custom-built frame buffer mounted in an Apple II+ microcomputer. The manipulators are controlled by the same microcomputer. The natural language interface, the learning mechanisms, the high-level robot programs and the high-level vision programs are implemented on a VAX 11/780, which communicates with the microcomputer via serial line. The user communicates with the system via a terminal to the VAX and observes image processing output via a monitor on the microcomputer. The overall arrangement of the system is shown in Fig. 1.

The example presented in this section is edited from a complete run of a partial assembly of a DC motor. This partial assembly involves three components of the motor: the case, the armature, and the cap, and it consists of placing the armature inside the case and then placing the cap on the case. This partial assembly is appropriate for the current research because it is the simplest example of assembly in a realistic domain, which involves the need for plan–step ordering. Issues of detailed assembly, such as the need to insure that the armature shaft is properly aligned with the bearings in the case, have been ignored in the present work, primarily because they require a level of accuracy which the current manipulators cannot achieve, and also because they require a level of sophistication in motion control [7] and in tactile sensing and control [8], which is beyond the scope of this research.

In Listing 1, the system knows what a motor armature and case look like, while a motor cap is an unknown object. The user interacts with the system to identify the unknown component and then instructs the system on how to perform the assembly. When the system does not know how to perform a requested action, it informs the user, who explains that action in turn. When the user's input is incomplete in some way, the system generates the most plausible completion and verifies it with the user. The user's inputs are shown in lowercase, the system's response in uppercase, and parenthetical notes have been added to describe system operation. The initial position of both manipulators is with the finger tips in a known position and the wrist pitch $-90°$, pointing straight down.

Manuscript received November 5, 1985; revised February 26, 1986.

The authors are with the Department of Electrical Engineering and Computer Science, The University of Connecticut, Storrs, CT 06268, USA.

IEEE Log Number 8608955.

0882-4967/86/0900-0167$01.00

Listing 1. User session to identify unknown object (motor cap).

)Display the current images.
OK. (System displays the images on the monitor.)
)Threshold the current images at twenty.
OK. (Thresholds the image at 20.)
)Try thirty.
DO YOU MEAN: "Threshold the current images at thirty?"
)Yes.
OK. (System thresholds original images at 30).
)Apply boundary following for regions larger than thirty pixels.
DO YOU MEAN: "Apply boundary following for regions larger than thirty pixels in the current images?"
)Yes.
OK. (System applies boundary following for regions larger than thirty pixels.)
)What is in the current images?
THERE IS A MOTOR CASE IN THE CURRENT IMAGES.
(Draws a box around the recognized regions in each image.)
THERE IS A MOTOR ARMATURE IN THE CURRENT IMAGE.
(Draws a box around the recognized regions in each image).
THERE IS ALSO ONE UNIDENTIFIABLE OBJECT IN THE CURRENT IMAGE.
)Is there a motor cap in the current image?
WHAT DOES A MOTOR CAP LOOK LIKE?
)A motor cap is a half inch square piece of plastic.
THERE IS A MOTOR CAP IN THE CURRENT IMAGE.
(Draws a box around the recognized regions in each image.)
)Make a motor.
I DO NOT KNOW HOW TO MAKE A MOTOR.
PLEASE EXPLAIN ...
)Move the motor case to the assembly position.[1]
I DO NOT KNOW HOW TO MOVE THE MOTOR CASE TO THE ASSEMBLY POSITION. PLEASE EXPLAIN ...
)Use the left robot arm.
(System invokes command for use of left arm.)
)Grasp the motor case.
(System grasps the motor case, using knowledge of the location of the motor case derived from the vision system.)
)Raise the robot hand two inches.
(System raises the robot hand two inches.)
)Pitch the robot wrist to zero degrees.
(System bends the robot wrist to position the motor case to that the armature can be inserted.)
)Move the robot arm to the assembly position.
(System moves robot arm to assembly position, and updates workspace knowledge about the new positions of the arm and the motor case.)
)End.
I KNOW NOW HOW TO MOVE THE MOTOR CASE TO THE ASSEMBLY POSITION.
FIRST I USE THE LEFT ROBOT ARM
THEN I GRASP THE MOTOR CASE
THEN I RAISE THE ROBOT HAND TWO INCHES
THEN I PITCH THE ROBOT WRIST TO ZERO DEGREES
THEN I MOVE THE ROBOT ARM TO THE ASSEMBLY POSITION.
WHAT IS THE NEXT STEP IN "MAKE A MOTOR?"
)Insert the motor armature into the motor case.
I DO NOT KNOW HOW TO INSERT THE MOTOR ARMATURE INTO THE MOTOR CASE. PLEASE EXPLAIN ...
)Move the motor case to the assembly position.
(System does nothing since it knows the motor case is already in the assembly position.)
)Use the right robot arm.
(System invokes command for use of right arm.)
)Grasp the motor armature.
(System grasps the motor armature.)
)Raise the robot hand two inches.
(System raises the robot hand two inches.)
)Pitch the robot wrist to zero degrees.
(System bends the robot arm upward 90 degrees to position the armature for insertion into the motor case.)
)Move the robot hand two inches above the motor case.
(System moves the right hand two inches above the motor case, using updated workspace knowledge.)
)Lower the robot hand one inch.
(System partially lowers the armature into the motor case.)
)Open the robot hand.
(System opens robot hand, thus dropping the motor armature into the motor case.)
)End.
I KNOW NOW HOW TO INSERT THE MOTOR ARMATURE INTO THE MOTOR CASE:
FIRST I MOVE THE MOTOR CASE TO THE ASSEMBLY POSITION
THEN I USE THE RIGHT ROBOT ARM
THEN I GRASP THE MOTOR ARMATURE
THEN I RAISE THE ROBOT HAND TWO INCHES
THEN PITCH THE ROBOT WRIST TO ZERO DEGREES
THEN I MOVE THE ROBOT HAND TWO INCHES ABOVE THE MOTOR CASE
THEN I LOWER THE ROBOT HAND ONE INCH
THEN I OPEN THE ROBOT HAND.
WHAT IS THE NEXT STEP IN "MAKE A MOTOR?"
)Place the motor cap on the motor case.
I DO NOT KNOW HOW TO PLACE THE MOTOR CAP ON THE MOTOR CASE.
PLEASE EXPLAIN ...
)Move the motor case to the assembly position.
(System does nothing since it knows the motor case is already in the assembly position.)
)Use the robot arm.
(System invokes command for use of right arm.)
)Grasp the motor cap.
(Robot grasps the motor cap using workspace knowledge about the location of the motor cap derived from the vision system.)
)Move the robot arm two inches above the motor case.
(System moves the right hand two inches above the motor case, using updated workspace knowledge.)
)Lower the robot hand two inches.
(System lowers the robot hand two inches, placing the motor cap on the case.)
)Open the robot hand.
(System opens robot hand.)
)End.
I NOW KNOW HOW TO PLACE THE MOTOR CAP ON THE MOTOR CASE.
FIRST I MOVE THE MOTOR CASE TO THE ASSEMBLY POSITION
THEN I USE THE RIGHT ROBOT ARM
THEN I GRASP THE MOTOR CAP
THEN I MOVE THE ROBOT ARM TWO INCHES ABOVE THE MOTOR CASE
THEN I LOWER THE ROBOT HAND TWO INCHES
THEN I OPEN THE ROBOT HAND.
WHAT IS THE NEXT STEP IN "MAKE A MOTOR?"
)End.
I NOW KNOW HOW TO MAKE A MOTOR:
FIRST I MOVE THE MOTOR CASE TO THE ASSEMBLY POSITION
THEN I PLACE THE MOTOR ARMATURE IN THE MOTOR CASE
THEN I PLACE THE MOTOR CAP ON THE MOTOR CASE.
(At this point the user replaces the components in the workspace.)
)Make a motor.
(System carries out the assembly as it was just taught.)

[1] The assembly position is a predefined position in the workspace that can be reached by both arms; such that one arm can hold a component so the other arm can perform some manipulation on or about that component.

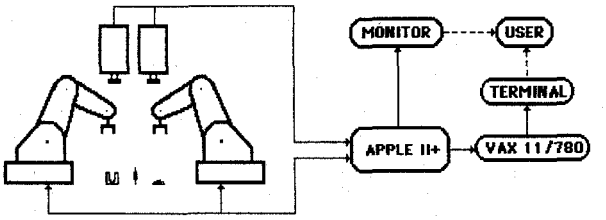

Fig. 1. Overall arrangement of the system.

This example showed a complete run of the system learning to perform a partial assembly of a small electric motor. In the first part of the example the user guided low-level image processing, operating in parallel for both cameras. Incompletely understood utterances were understood and verified by the user. High-level image processing was then performed automatically, recognizing two of the three motor components. It learns what the motor cap looks like via a mixed-initiative natural language interaction and then recognizes the cap in the image. The real-world coordinates of the recognized components are then calculated and stored for use by the manipulation system. In the second part of the example the user directed the system to assemble the three-motor components. When the system did not know how to perform an action, it asked for and received more detailed directions for that action.

This example displayed a variety of natural language abilities. Using a vocabulary of about 50 words, the system understood commands directing low-level vision processing, high-level questions requiring high-level vision processing, and commands to perform actions with the manipulators. Furthermore, it engaged in mixed-initiative conversation, it used conversational history to understand a command, and it displayed an ability to deal with ellipsis and conversational topic shifts. Finally, it learned new visual knowledge and several new planning rules from natural language input by the user. Thus, natural language conversational ability enabled a user to direct a complex assembly task with only minimal technical knowledge.

III. System Architecture

The system has three components. First, the vision and manipulation component consists of a high-level planning system, a low-level robot control system, planning knowledge, a high-level vision system, a low-level vision system, visual knowledge, and knowledge about the locations and orientations of the manipulators and objects in the workspace. Second, the natural language interface consists of a natural language analyzer, a natural language generator, and language knowledge of syntax and semantics. Finally, the learning system enables plan learning and learning visual knowledge. The system is shown in Fig. 2, where boxes denote system modules, ovals denote data structures, bold lines denote control flow, fine lines denote data flow, and hollow lines denote knowledge acquisition.

This section describes the three components separately and then examines the flow of control occurring between the components during the example in the previous section.

A. Vision and Manipulation Component

This component consists of a high-level vision system, a low-level vision system, a high-level planning system, a low-level robot control system, and workspace knowledge about the locations and orientations of the manipulators and objects in the workspace. Accordingly, the high-level and low-level vision systems use visual knowledge and the planning system and low-level robot control systems use planning knowledge. These systems will be described in this order.

Low-level image processing takes place on the Apple II+ microcomputer. Two calibrated Microworks DS-65 digitizers accept video from two RCA 2011/16 cameras. The image from each camera is stored in one 120 × 120 × 8 frame of a custom-built four-frame frame buffer. Frame buffer ROM routines and BASIC programs used to build a set of descriptors that describe the two-dimensional regions found in the image of each camera in terms of pixels and pixel coordinates. These descriptors include information on the area, circularity, centroid, extent, boundary and polygonal approximation of each region in each image. These descriptors are transmitted to the VAX via serial line.

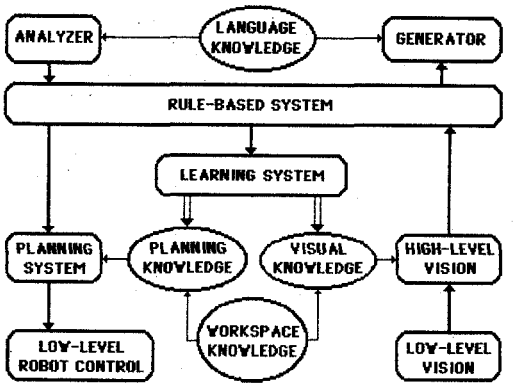

Fig. 2. System architecture.

High-level image processing requires the low-level region descriptors received from the Apple. These descriptors correspond to the two-dimensional regions found in each image. Characteristics of each region are used in a simple recognition scheme to identify each region with a known object. The regions from each image which correspond to the same object are used to provide a stereo pair from which, along with camera calibration data, the three-dimensional location of that object is computed. Finally, the identity and location of each workspace object is added to the system's workspace knowledge, where it is accessible to the manipulation system. This approach can be characterized as "intelligent vision processing" since, although it is quite rudimentary, it attempts to combine high-level model-based knowledge with intermediate and low-level knowledge through a reasoning process [1], rather than relying on, say, standard pattern recognition or binary vision techniques.

The low-level manipulation system consists of two stepper-motor five-degree-of-freedom Microbot Minimover-5 manipulators controlled by the Apple II+. A program on the Apple accept commands from the VAX in terms of arm coordinates to be achieved. It computes the forward solution to obtain the joint angles needed to place the manipulator in this configuration, and uses information about the manipulator's current position to compute the desired straight-line finger-tip trajectory to the desired position. In order to drive the tip along this desired trajectory, the program interpolates a sequence of intermediate points along the desired trajectory. The program then uses joint-interpolation motor control to drive the tip along an approximation to the desired straight-line trajectory and thus to the desired position.

The high-level manipulation system is written in LSP and runs on the VAX 11/780. Its task is to accept a high-level goal and produce the sequence of manipulator configurations necessary to achieve that goal. Its architecture is that of a stack-oriented goal-based planning system [3]. It uses a set of if–then rules that expresses the knowledge that in order to achieve a certain goal, the system must either achieve a certain sequence of subgoals or must execute a certain primitive action, which sends a command to the Apple. Goals are expressed using a set of state predicates and generated symbols. Primitive actions correspond to opening and closing the fingers, raising, lowering, pitching, or rolling the hand, moving the arm to some location, and switching between manipulators. Planning begins when a goal is placed on the stack. If according to workspace knowledge the goal is not achieved, then if–then rules are retrieved which match that goal, and which push additional subgoals or primitive actions onto the stack. If ever a primitive action is at the top of the stack, it is executed and the stack is popped. In addition, workspace knowledge is updated with the effect of the action. Thus any goal can be achieved by continuously decomposing that goal to obtain a series of primitive actions to control the manipulators.

The vision and manipulation component has certain limitations. It does not employ three-dimensional vision processing. It does not use tactile sensors or sensor-based movement strategies [8]. Nonetheless, it is adequate for the research presented here, and the current system is compatible with the removal of such limitations in the future. Although the current system does not employ them, previous versions of the vision and manipulation components performed the additional tasks of inferring the three dimensional volume for each component, and using that in a C-Space-based collision avoidance algorithm [7]. These were omitted from the current system in the interests of execution time.

B. Natural Language Interface

The natural language component of the system is also written in LISP on a VAX 11/780. The representation for the meaning and syntax will first be described, followed by language analysis and generation. Interaction between the analyzer, generator, and the vision and manipulation component and the learning component is mediated by a rule-based system, which will be described at the end of this subsection.

Word meanings and domain knowledge are represented using conceptual dependency (CD) [13]. CD is a system in which knowledge is represented by CD "concepts." Such concepts consist of a predicate representing the concept followed by labeled "slots," which carry particular information associated with that instance of the concept or which specify potential links to other concepts. However, it is important to note that neither the natural language component nor the system as a whole is limited to CD's; rather, within broad limits, any conceptual representation can be used. Syntactic knowledge is represented using syntactic features associated with every empty slot in a particular word meaning. These features describe where in the input a filler for that slot can be found. They are formed from the positional predicates "precedes" and "follows," which relate the position of a candidate slot filler to either the word whose meaning contains the slot being filled or a filler of another slot in that same meaning of the lexical function word.

The system's language analysis program is a version of the CA analyzer [2], [15]. It operates by reading the words of a user's input one by one, left to right. When it reads a word, it places the CD concept representing its meaning on the concept list or CLIST, an immediate processing memory. Then, the semantic and syntactic requirements associated with the slots in the concepts from previous words are checked to see if the new concept can be used to fill any of them. Likewise, the requirements associated with the new concept are checked to see if they can be filled by any of the concepts already on the CLIST. In this way, the meaning of the whole user input is built up. If, however, the user omitted certain words, the analyzer uses semantic constraints imposed by the knowledge representation to search the conversational history and its general domain knowledge for appropriate fillers. If the input contained words in an unusual order, the analyzer relaxes the syntatic constraints to produce the most plausible interpretation of the input. In both cases, the analyzer's completed interpretation is checked with the user before the system proceeds.

The system's language generator is a version of the CGEN program [4]. Language generation begins when a knowledge structure is placed on the CLIST and the generator is called. When the generator finds a concept on the front of the CLIST, it searches its dictionary to find the word whose CD meaning matches the concept being generated. When it finds one which matches exactly, it places this work on the front of the CLIST. When the generator finds a word on the front of the CLIST, it "says" the word by printing it. If, however, the dictionary contains only a word whose meaning matches only the root CD concept of the overall concept being expressed, then the generator places that word on the CLIST, and the CD concepts that filled slots in the root concept are inserted into the CLIST in the positions specified by syntax stored under the word matching the root concept. The generator continues by examining the front of the CLIST, either printing the words it finds or looking up words for the concepts, until the CLIST is empty.

Responses to user inputs are controlled by a rule-based top–down, stack-oriented program, called the conversational controller or CCON. The stack is initialized with the understood meaning of that input and the program is called. It compares the "if" clause of each if–then rule with the meaning on the top of the stack. When a match is found, the stack is popped and the "then" clause of the matched rule is evaluated or is said to have "fired." Evaluation results in either other concepts being pushed on the stack or calls made to the vision/manipulation system. This execution of image processing and robotic manipulation is the link between these functions and natural language.

The system currently has a vocabulary of about 50 different words and phrases of a number of different types. It knows object words, action words, object descriptors, action descriptors, function words, numbers, relation words, and determiners. Although currently limited, this vocabulary has allowed testing of the system's ability to handle different kinds of words, constructions, and input types. It should be noted that the system currently knows words and phrases for components the appearance of which it does not know. This is not unreasonable, since humans sometimes know that a certain object exists without being able to visually identify that object. In fact, providing such words and phrases was a research tactic designed to avoid the problem of learning new words within the research reported here. A real-world system would know a number of domain-independent words and concepts and a number of domain-dependent words and concepts. It is not clear what percentage of each would be present in a real-world system; additional experience is needed in order to assess the effort needed to build a real-world vocabulary.

C. Learning System

The system can exhibit learning in two different areas, learning new visual knowledge and learning new assembly plans. Although rudimentary by comparison with a system that could be useful in a realistic manufacturing environment, these abilities demonstrate the utility of learning in a natural language conversation. The underlying strategy behind both learning abilities is that learning takes place when the user asks the system to perform an action, and the system realizes that it lacks certain data to perform that action. The system generates a request to the user for that data. If the user responds with the needed information, the system can then proceed to carry out the original action. Each type of learning is implemented in a set of if-then rues used by the CCON program.

Learning of visual knowledge occurs in the above example in connection with the motor cap, and employs techniques which are relatively general. When the user asks whether the current image contains a particular component, the system attempts to retrieve a description of the appearance of that component in order to search the workspace knowledge for an object of that appearance. It does this by pushing a goal onto CCON's stack to obtain the appearance of the component, which triggers a rule which asks the user for the appearance of the component. The meaning of the user's response is placed on the stack. If it contains information that answers the question, it triggers a rule which adds that information to the workspace knowledge. If, as in the example, the user's answer is not a description of the appearance of the component, other rules are tested. In the example, the user in fact provides a description of the component's physical characteristics in terms of shape, dimensions, and material. This triggers a rule which infers an object's appearance from its physical characteristics; In this example, this rule builds a specification of the appearance of the component by accessing knowledge that plastic objects appear light and knowledge about shape and dimension. The system then uses this knowledge of the appearance of the component to locate it, and it proceeds to answer the user's original question. In addition, of course, it now has learned new visual knowledge about the physical characteristics and visual appearance of the component.

The learning of new assembly plans is initiated when the user directs the system to achieve a goal for which it knows no plans. In a manner similar to learning visual knowledge, the system asks the user for an explanation of the plan to achieve that goal. The user supplies commands to carry out the components of the plan, and the system them. After successfully achieving the original goal, the system builds a new planning rule expressing the knowledge that, to achieve the original goal, use the plan provided by the user and adds that rule

to the planning knowledge. The next time the user requests that the system achieve that goal, it will have a plan available. In addition, it can occur, as in the example, that the system does not know how to carry out some part of the user's explanation. In this case, the learning process is reinvoked. Reinvokation can continue until the user explains how to achieve a goal in terms of primitive manipulator motions that the system does know how to perform, using a special stack to keep track of which subgoals are part of plans to achieve which higher-level goals. Note that no new planning rule contained any variables; each was specific to the goal intended to trigger it. The issue of generalizing planning rules is complex [9] and beyond the scope of this research.

D. Flow of Control

The cycle of the system begins when the user is initially prompted for natural language input. The input is analyzed and its meaning is placed on the stack. CCON retrieves a rule whose "if" clause matches the user's meaning on the top of the stack, and executes its "then" clause. If the "then" clause refers to low-level image operations, such as in "Threshold the current image at forty," the appropriate low-level vision program is invoked on the Apple. If the user's input refers to a high-level vision operation, as in "What is in the current image," the "then" clause specifies a series of steps which update the workspace knowledge with information about the objects in the workspace. If the user's input directs the system to achieve a goal, that goal is pushed on the stack and the system retrieves planning rules to achieve it.

Mixed-initiative interactions occur since either the user or the system can ask questions and make statements. Indeed, either can answer a question with a question, since either may need to know certain information in order to answer the original question. The system uses CCON's stack to keep track of which goal it is currently trying to achieve, and thus which conversational topic is being pursued at the moment. Learning occurs when input by the user's is recognized as a statement and is added to the workspace knowledge where it is accessible to the system.

Thus, the system's operation centers around a central stack and if-then rules which are tested against the top of the stack. The user is prompted for input when no further rules are triggered by the top of the stack. His response is placed on the stack, and the rules are tested again. This cycle continues throughout the session with the system.

IV. Conclusion

This paper has described a natural language interface to a robot assembly system which includes the ability to learn new visual knowledge and new assembly plans. Although in absolute terms performance is modest, it appears to us that each component of the system represents a technology which is relatively generic. Thus, the vision and manipulation component operates to interact with objects in the workspace in a more-or-less general way, the natural language system uses techniques which have been widely applied and extended in a number of different areas [5], [12], [15] and the learning mechanisms appear generalizable to a wide range of vision and planning knowledge. It thus appears that the techniques which have been employed in this system, conceptual analysis technology to perform natural language processing, high-level planning to perform robot control, intelligent vision processing, learning-by-being-told and an overall rule-based stack oriented architecture, represent an appropriate starting point for research toward a real-world natural language interface to a robot assembly system.

The techniques used here could not be applied directly, however. Each component would have to be improved to where it represented the state-of-the-art in its particular area. While this system offers a basis for a general robotic assembly system, it is not a complete general robotic assembly system. Specifically, the vision component should build a full three-dimensional geometric representation of the objects in the workspace [1] and the manipulator system should use high-performance problem solving techniques [3], path planning and obstacle avoidance [7], and tactile sensing and control [8]. The natural language system requires a greatly expanded vocabulary, in order to respond to a wide variety of user inputs, and more sophisticated processing [5], [12]. The learning mechanisms must be expanded to include the learning and comparison of multiple plans to achieve a goal, and generalization of planning rules [9]. Further research will address the question of incorporating such advanced techniques into the system and consider the feasibility its use in a real-world environment.

References

[1] D. Ballard and C. Brown, *Computer Vision.* Englewood Cliffs, NJ: Prentice-Hall, 1982.

[2] L. Birnbaum and M. Selfridge, "Conceptual analysis of natural language," in *Inside Computer Understanding,* R. C. Schank and C. K. Riesbeck, Eds. Hillsdale, NJ: Lawrence Erlbaum.

[3] E. Charniak and D. McDermott, *Introduction to Artificial Intelligence.* Reading, MA: Addison-Wesley, 1985.

[4] R. E. Cullingford, M. W. Krueger, M. Selfridge, and M. Bienkowski, "Automated explanations as a component of a computer-aided design system," *IEEE Trans. Syst., Man, Cybern.,* (Special Issue on Human Factors and User Assistance), 1981.

[5] M. Dyer, *In-Depth Understanding.* Cambridge, MA: M.I.T. 1983.

[6] G. G. Hendrix, "LIFER: A Natural Language Interface Facility," *SIGART Newsletter,* vol. 61.

[7] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-11, no. 6, pp. 681–698, 1981.

[8] M. T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-11, no. 6, pp. 418–432, 1981.

[9] R. Michalski, J. G. Carbonell, and T. M. Mitchell, *Machine Learning: An Artificial Intelligence Approach.* Palo Alto, CA: Tioga, 1983.

[10] R. Paul, *Robot Manipulators: Mathematics, Programming, and Control.* Cambridge, MA: M.I.T., 1981.

[11] R. C. Schank, "Identification of conceptualizations underlying natural language," in *Computer Models of Thought and Language,* R. C. Schank and K. M. Colby, Eds. San Francisco: W. H. Freeman, 1973.

[12] R. C. Schank, *Dynamic Memory: A Theory of Learning in Computers and People.* Cambridge: Cambridge University.

[13] R. C. Schank and R. P. Abelson, *Scripts, Plans, Goals, and Understanding.* Hillsdale, NJ: Lawrence Erlbaum, 1977.

[14] R. C. Schank and C. K. Riesbeck, "Comprehension by computer: expectation-based analysis of sentences in context," tech. rep. 78, Dept. Comp. Sci., Yale University, New Haven, CT.

[15] M. Selfridge, "Integrated processing produces robust understanding," *Computational Linguistics,* to be published.

[16] J. Weizenbaum, "ELIZA—A computer program for the study of natural language communication between man and machine," *Communications of the ACM,* vol. 9, no. 1, pp. 36–44, 1966.

[17] T. Winograd, *Understanding Natural Language.* New York: Academic, 1982.

[18] W. A. Woods, "Transition Network Grammars for Natural Language Analysis," *Commun. ACM,* vol. 13, pp. 591–606, 1977.

Robot Accuracy Analysis Based on Kinematics

W. K. VEITSCHEGGER AND CHI-HAUR WU

Abstract—**The positioning accuracy problem of robot manipulators has long been one of the principal concerns of robot design and control. In a previous work, a linear model that described the robot positioning accuracy due to kinematic errors was developed. However, the previous**

Manuscript received July 25, 1985; revised February 20, 1986. This work was supported by the National Science Foundation under NSF Grant DMC-8503916 and the Engineering Foundation under Grant RI-A-84-10.

The authors are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60201, USA.

IEEE Log Number 8608956.