Command, Goal Disambiguation, Introspection, and Instruction in Gesture-Free Spoken Dialogue with a Robotic Office Assistant

Vladimir A. Kulyukin
Computer Science Assistive Technology Laboratory (CSATL)
Utah State University
U.S.A.

1. Introduction

Robotic solutions have now been implemented in wheelchair navigation (Yanco, 2000), hospital delivery (Spiliotopoulos, 2001), robot-assisted navigation for the blind (Kulyukin et al., 2006), care for the elderly (Montemerlo et al., 2002), and life support partners (Krikke, 2005). An important problem facing many assistive technology (AT) researchers and practitioners is what is the best way for humans to interact with assistive robots. As more robotic devices find their way into health care and rehabilitation, it is imperative to find ways for humans to effectively interact with those devices. Many researchers have argued that natural language dialogue (NLD) is a promising human-robot interaction (HRI) mode (Torrance, 1994; Perzanowski et al., 1998; Sidner et al., 2006).

The essence of the NLD claim, stated simply, is as follows: NLD is a promising HRI mode in robots capable of collaborating with humans in dynamic and complex environments. The claim is often challenged on methodological grounds: if there is a library of robust routines that a robot can execute, why bother with NLD at all? Why not create a graphical user interface (GUI) to that library that allows human operators to invoke various routines by pointing and clicking? Indeed, many current approaches to HRI are GUI-based (Fong & Thorpe, 2001). The robot's abilities are expressed in a GUI through graphical components. To cause the robot to take an action, an operator sends a message to the robot through a GUI event. The robot sends feedback to the GUI and the interaction cycle repeats.

The NLD proponents respond to this challenge with three arguments. First, it is argued that, to humans, language is the most readily available means of communication. If the robot can do NLD, the human can interact with the robot naturally, without any of the possibly steep learning curves required for many GUI-based approaches. Second, it is argued that, from the practical point of view, GUI-based approaches are appropriate in environments where the operator has access to a monitor, a keyboard, or some other hardware device, e.g., an engineer monitoring a mining robot or an astronaut monitoring a space shuttle arm. In some environments, however, access to hardware devices is either not available or impractical. For example, an injured person in an

Source: Mobile Robots Towards New Applications, ISBN 3-86611-314-5, Edited by Aleksandar Lazinica, pp. 784, ARS/plV, Germany, December 2006

urban disaster area is unlikely to communicate with a rescue robot through a GUI. Thus, NLD in robots interacting with humans has practical benefits. Finally, an argument is made that building robots capable of NLD yields insights into human cognition (Horswill, 1995). Unlike the first two arguments made by the NLD camp, this argument does not address either the practicality or the naturalness of NLD, the most important objective being a cognitively plausible integration of language, perception, and action. Although one may question the appropriateness or feasibility of the third argument in a specific context, the argument, in and of itself, appears to be sound insomuch as the objective is to test a computational realization of a given taxonomy or some postulated cognitive machinery (Kulyukin, 2006). This chapter should be read in the light of the third argument.

In this chapter, we will present a robotic office assistant that realizes an HRI approach based on gesture-free spoken dialogue in which a human operator engages to make a robotic assistant perform various tasks. The term *gesture-free* is construed as devoid of gestures: the operator, due to a disability or a task constraint, cannot make any bodily gestures when communicating with the robot. We discuss how our approach achieves four types of interaction: command, goal disambiguation, introspection, and instruction-based learning.

The two contributions of our investigation are a shallow taxonomy of goal ambiguities and an implementation of a knowledge representation framework that ties language and vision both top-down and bottom-up. Our investigation is guided by the view that, given the current state of the art in autonomous robotics, natural language processing, and cognitive science, it is not feasible to build robots that can adapt or evolve their cognitive machinery to match the level of cognition of humans with whom they collaborate. While this remains the ultimate long-term objective of AI robotics, robust solutions are needed for the interim, especially for people with disabilities. In our opinion, these interim solutions can benefit from knowledge representation and processing frameworks that enable the robot and the operator to gradually achieve mutual understanding with respect to a specific task. Viewed in this light, human-robot dialogue becomes a method for achieving common cognitive machinery in a specific context. Once the operator knows the robot's goals and what language inputs can reference each goal, the operator can adjust language inputs to suit the robot's recognition patterns. Consequently, the human-robot dialogue is based on language pattern recognition instead of language understanding, the latter being a much harder problem.

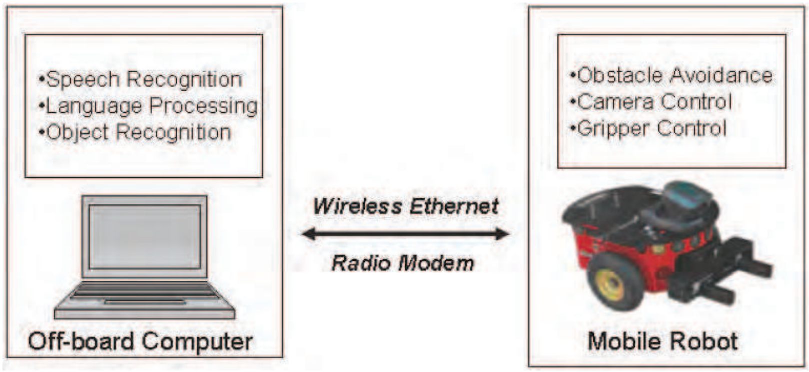


Fig. 1. Realization of the 3T skills on the hardware.

The scenario discussed in the chapter is the operator using gesture-free spoken dialogue to make the robot pick up various pieces of trash, such as soda cans, coffee cups, and crumpled pieces of paper, and to carry them to designated areas on the floor. The environment is a standard office space where the operator and the robot are not the only actors. The operator wears a headset that consists of one headphone and one microphone.
The chapter is organized as follows. Section 2 presents the system's architecture. Section 3 is dedicated to knowledge representation. We discuss the declarative, procedural, and visual types of knowledge and how these types interact with each other. Section 4 discusses how command, goal disambiguation, introspection, and instruction can be achieved in gesture-free spoken dialogue. Section 5 presents some experimental evidence. Section 6 reviews related work. Section 7 states several limitations of our approach and discusses future work. Section 8 presents our conclusion.

2. System Architecture

The robot is a Pioneer 2DX robot assembled from the robotic toolkit from ActivMedia Robotics, Inc. (www.activmedia.com). The robot has three wheels, an x86 200 MHZ onboard Windows PC with 32MB of RAM, a gripper with two degrees of motion freedom, and a pan-tilt-zoom camera. The camera has a horizontal angle of view of 48.8 degrees and a vertical angle of view of 37.6 degrees. Images are captured with the Winnov image capture card (www.winnov.com) as 120 by 160 color bitmaps. The other hardware components include a CCTV-900 wireless video transmitter, an InfoWave radio modem, and a PSC-124000 automatic battery charger. Captured images are sent through the video transmitter to an offboard Windows PC where the image processing is done. The processing results are sent back to the robot through the radio modem.
The robot has a 3-tier (3T) architecture (Kortenkamp et al., 1996). The architecture consists of three tiers: deliberation, execution, and control. The deliberation tier does knowledge processing; the execution tier manages behaviors; the control tier interacts with the world through continuous control feedback loops called *skills*. The deliberation tier passes to the execution tier task descriptions in the form of sequences of goals to satisfy, and receives from the execution tier status messages on how the execution proceeds. The execution tier sends commands to the control tier and receives sensory data messages from the control tier. The execution tier is implemented with the Reactive Action Package (RAP) system (Firby, 1989). The system consists of a behavior programming language and an interpreter for executing behaviors, called *RAPs*, written in that language. A RAP becomes a *task* when its index clause is matched with a goal description passed to the RAP system from the deliberation tier. Each task is added to the task agenda. To the RAP interpreter, a task consists of a RAP, the RAP's variable bindings, the execution progress pointer in the RAP's code, and the RAP's execution history. The interpreter continuously loops through the task agenda picking up tasks to work on. The interpreter executes the task according to its current state. It checks if the success condition of the task's RAP is satisfied. If the success condition is satisfied, the task is taken off the agenda. If the success condition is not satisfied, the interpreter looks at the step pointed to by the execution progress pointer. If the pointer points to a skill, the control tier is asked to execute it. If the pointer points to a RAP, the interpreter finds a RAP, checks the success condition and then, if the success condition is not satisfied, selects a method to execute. The RAP system runs on-board the robot. If the chosen task points to a primitive command, the control tier executes an appropriate skill. The RAPs are discussed in further detail in the section on procedural knowledge representation below.

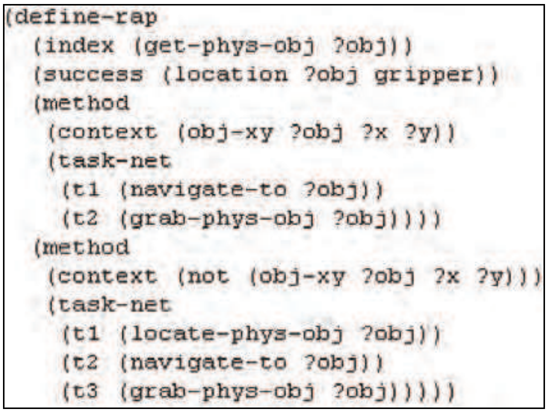
The control tier hosts the robot's skill system. The skill system consists of speech recognition skills, object recognition skills, obstacle avoidance skills, and skills that control the camera and the gripper. The speech recognition and object recognition skills run on an off-board computer. The rest of the skills run on-board the robot. The skills running on-board the robot interface to Saphira from ActivMedia Robotics, Inc., a software library for controlling the Pioneer robot's hardware, e.g., reading sonars, setting rotational and translational velocities, tilting and panning the camera, closing and opening the gripper, etc. Skills are enabled when the execution tier asks the control tier to execute them. An enabled skill runs until it is either explicitly disabled by the execution tier or it runs to completion. The control tier is not aware of such notions as success or failure. The success or failure of a skill is determined in the execution tier, because both are relative to the RAP that enabled the skill. Figure 1 shows the realization of the control tier skills on the system's hardware.

3. Knowledge Representation

Our knowledge representation framework is based on the idea commonly expressed in the cognitive science and AI literature that language and vision share common memory structures (Cassell et al., 1999). In our implementation, this idea is realized by having language and vision share one memory structure - a semantic network that resides in the deliberation tier. The semantic network provides a unified point of access to vision and action through language and shapes the robot's interaction with its environment. It also provides a way to address the symbol grounding problem by having the robot interpret the symbols through its experiences in the world.

3.1 Procedural Knowledge

The robot executes two types of actions: *external* and *internal*. External actions cause the robot to sense and manipulate external objects or move around. Internal actions cause the robot to manipulate its memory. Objects and actions are referenced by language. Referenced actions become goals pursued by the robot. Procedural knowledge is represented through RAPS and skills. A RAP is a set of methods for achieving a specific goal under different circumstances.



```
(define-rap
  (index (get-phys-obj ?obj))
  (success (location ?obj gripper))
  (method
   (context (obj-xy ?obj ?x ?y))
   (task-net
    (t1 (navigate-to ?obj))
    (t2 (grab-phys-obj ?obj))))
  (method
   (context (not (obj-xy ?obj ?x ?y)))
   (task-net
    (t1 (locate-phys-obj ?obj))
    (t2 (navigate-to ?obj))
    (t3 (grab-phys-obj ?obj)))))
```

Fig. 2. A RAP for getting a physical object.

As an example, consider the RAP for getting a physical object, i.e., a coffee cup, a soda can or a crumpled piece of paper, given in Figure 2. The index clause of this RAP specifies the goal the

RAP is supposed to achieve. The symbols that begin with question marks denote variables. The success clause specifies the condition that must be true in the RAP memory for the RAP to succeed. In this case, the RAP succeeds when the object is in the robot's gripper. The RAP has two methods for getting an object. The steps needed to execute a method are referred to as *task nets*. The applicability of each method is specified in its context clause. For example, the RAP's first method assumes that the robot knows where the object is, i.e., the *x* and *y* coordinates of the object relative to the robot. On the other hand, the second method makes no such assumption and instructs the robot to locate the object first. For example, *(get-phys-obj pepsican)* matches the index clause of the RAP in Figure 2, unifying the variable *?obj* with the symbol *pepsican*.
Vision is considered a function that extracts restricted sets of symbolic assertions from images. This approach has its intellectual origins in *purposive vision* (Firby et al., 1995): instead of constructing the entire symbolic description of an image, the robot extracts only the information necessary for the task at hand. For example, *detect-obj-skill* is enabled when a RAP method needs to detect an object. The skill has the following predicate templates associated with it: *(detected-obj ?obj)*, *(dist-to ?obj ?mt)*, *(obj-xy ?obj ?x ?y)*, and *(sim-score ?obj ?sc)*. These templates, when filled with variable bindings, state that an object *?obj* has been detected *?mt* meters away from the robot, and the bottom left coordinates of the image region that had the best matching score of *?sc* are *?x* and *?y*. The output of a skill is a set of symbolic assertions, i.e., predicate templates instantiated with variable bindings and asserted in the RAP interpreter's memory. In this way, information obtained from vision percolates bottom-up to the deliberation tier and becomes accessible through language.

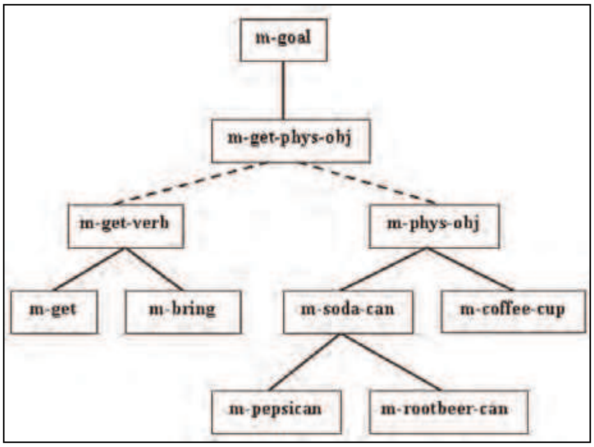


Fig. 3. A part of the robot's DMAP-Net.

3.2 Declarative Symbolic Knowledge

The declarative symbolic knowledge is a semantic network of goals, objects, and actions. Figure 3 shows a small part of the robot's semantic network. Nodes in the network are activated by a spreading activation algorithm based on direct memory access parsing (DMAP) (Martin, 1993). Since the semantic network uses DMAP, it is referred to as DMAP-Net. The activation algorithm is detailed in the section on knowledge access below. Each node in the network is a memory organization package (MOP) (Riesbeck & Schank, 1989) and, therefore, starts with the *m*-prefix. The solid lines correspond to the *abstraction* links; the dotted lines denote the *packaging* links. For example, *m-get-phys-obj*, which corresponds to the action of getting a physical object, has two *packaging* links: the first one links to *m-get-verb* and the second one links to *m-phys-obj*. A packaging link is loosely interpreted as a *part-of* link. The two links assert that *m-get-phys-obj* has two typed slots: *m-get-verb* and *m-phys-obj*. Every MOP is either an *abstraction* or a *specification*. Since every action, when activated, becomes a goal pursued by the robot, every action MOP is a specification of *m-goal*, and, conversely, *m-goal* is an abstraction of every action MOP. Since the robot acts on a small closed class of objects, such as soda cans and crumpled pieces of paper, every goal's MOP has two packaging links: action and object. The action link denotes an executable behavior; the object link denotes an object to which that behavior is applied.

3.3 Visual Knowledge

In addition to symbolic names, objects are represented by model libraries created from images taken from different distances and with different camera tilts. Each object has two types of models: Hamming distance models (HDMs) and color histogram models (CHMs). The two types of models are used by the object recognition algorithms briefly described below. The distance is measured in meters from the robot's gripper to the object. The models are created for the following distances: 0.5, 1.0, 1.5, and 2 meters. This is because the robot's camera cannot detect anything further than 2 meters: the vision system is incapable of matching objects beyond that distance because of image resolution. For example, under the current resolution of 120 by 160 pixels, the size of a soda can that is more than 2 meters away from the robot is about 3-4 pixels. The vision system cannot perform meaningful matches on objects that consist of only a few pixels.

To create an HDM, an object's image is taken from a given distance and with a given camera tilt. The subregion of the image containing the object's pixels is manually cropped. The subregion is processed with the gradient edge detection mask (Parker, 1993) and turned into a bit array by thresholding intensities to 0 or 1. Figure 4 shows the process of HDM creation. The top left image containing a soda can is manually cropped from a larger image. The top right is the image after the application of the gradient edge detection mask. The bottom 2D bit array is obtained from the top right image by thresholding intensities to 0 and 1. An HDM consists of a 2D bit array and two integer row constraints specifying the region of an image where a model is applicable. For example, the row constraints of the two meter pepsican model are 40 and 45. At run time, if the robot is to find out if there is a pepsican two meters away from it, the robot will apply the model only in the subregion of the image delineated by rows 40 and 45. Each object has five HDMs: two models for the distance 0.5, one with tilt 0 and one with tilt -10, and three models with tilt 0 for the remaining distances.

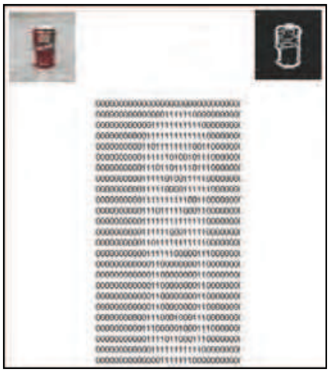


Fig. 4. HDM creation.

The CHMs are created for the same distances and tilts as the HDMs. To create a CHM, an object's image is taken from a given distance and with a given camera tilt. The region of the image containing the object is manually cropped. Three color histograms are created from the cropped region. The first histogram records the relative frequencies of red intensities. The other two do the same for blue and green. Each histogram has sixteen intensity intervals, i.e., the first is from 0 to 15, the second is from 16 to 31, etc. Thus, each CHM consists of three color histograms. The same row constraints as for the HDMs hold.

3.4 Object Model Matching

Two object recognition techniques are used by the robot: template matching (Young et al., 1994; Boykov & Huttenlocher, 1999) and color histogram matching (Swain & Ballard, 1991; Chang & Krumm, 1999). Object recognition happens in the control tier, and is part of the robot's skill system. HDM and CHM matching run in parallel on the off-board Windows PC connected to the robot via wireless Ethernet and a radio modem. A match is found when the regions detected by the HDM and the CHM for a given object and for a given distance have a significant overlap. The significance is detected by thresholding the matches in the overlap area.
Our template matching metric is based on a generalization of the Classic Hamming Distance (CHD) (Roman, 1992). CHD is an edit distance with the operations of insertion and deletion. Given two bit vectors of the same dimension, CHD is the minimum number of bit changes required to change one vector into the other. However, since it measures only the *exact* extent to which the corresponding bits in two bit vectors agree, CHD does not recognize the notion of approximate similarity. The *Generalized Hamming Distance* (GHD) is also an edit distance, but the set of operations is extended with a shift operation. By extending the operation set with the shift operation, we account for the notion of approximate matches when corresponding bits in two bit vectors are considered aligned even when they are not in the exact same positions. The mathematical properties of GHD are described in Bookstein et al. (2002). A dynamic programming algorithm for computing GHD is detailed in Kulyukin (2004).
Object recognition with HDMs is done as follows. Let $M_H (O, D, T, L, U)$ denote an HDM for object O at distance D and with tilt T and two row constraints L and U for the lower and upper rows, respectively. Let H and W be the model's height and width. Let I be an image. I is first processed with the gradient edge detection mask. To recognize O in I, the HDM is matched with the region of I whose bottom row is L and whose top row is $U + H$. The matching is done by moving the model from left to right C columns at a time and from top to bottom R rows at a time. In our experiments, $R=C=1$. The similarity coefficient between the model and an image region is the sum of the GHDs between the corresponding bit strings of the model and the image region normalized by the model's size. The sum is normalized by the product of the model's height and width. Similarity results are 4-tuples $<x, y, m, s>$, where x and y are the coordinates of the bottom left corner of the image region that matches the model m with the similarity score s. Similarity results are sorted in non-decreasing order by similarity scores and the top N matches are taken. In our experiments, $N=1$.
Object recognition with CHMs is similar. Let $M_C (O, D, T, L, U)$ be the $H \times W$ CHM of object O at distance D and with tilt T and two row constraints L and U. Let I be an image. To recognize O in I with $M_C (O, D, T, L, U)$, the $H \times W$ mask is matched with the appropriate region of the image R. The similarity between M_C and R is the weighted sum of the similarities between their corresponding color histograms. Let $h(H_1, H_2)$ is the similarity between two color histograms H_1 and H_2 computed as the sum of the absolute differences of their relative frequencies. Then $0 <= h(H_1, H_2) <= Q$, where Q is

a suitably chosen constant. Illumination invariant color spaces were not used because of the constant lighting assumption.

We do not claim that these metrics are superior to other template or color histogram matching metrics. Since our objective is the integration of perception and action, our focus is on the integration mechanisms that bring these cognitive processes together. It is our hope that, once these mechanisms are established, other object recognition techniques will be integrated into the proposed framework.

3.5 Declarative Knowledge Access

Knowledge in the DMAP-Net is accessed by processing *token sequences* obtained from speech recognition (Kulyukin & Steele, 2002). For example, when the operator says "Get the soda can," the speech recognition engine converts the input into a sequence of four tokens: *(get the soda can)*. A token is a symbol that must be directly seen in or activated by the input.

Token sequences index nodes in the DMAP-Net. For example, *m-pepsican* is indexed by two token sequences: *(a pepsican)* and *(the pepsican)*. If an input contains either sequence, *m-pepsican* is activated. Token sequences may include direct references to packaging links. For example, the only token sequence associated with *m-get-phys-obj* is *(get-verb-slot phys-obj-slot)*, which means that for *m-get-phys-obj* to be activated, *m-get-verb* must be activated and then *m-phys-obj* must be activated.

Token sequences are tracked at run time with *expectations* (Fitzgerald & Firby, 2000). An expectation is a 3-tuple consisting of a MOP, a token sequence, and the next token in the token sequence that must be seen in the input. For example, *X* = <*m-pepsican, (a pepsican), a*> is a valid expectation. The first element of an expectation is referred to as its target; the second element is a token sequence associated with the target; the third element is the *key*. When an input token is equal to an expectation's key, the expectation is *advanced* on that token. For example, if the next token in the input is *a*, *X* is advanced on it and becomes <*m-pepsican, (pepsican), pepsican*>. If the next input token is *pepsican*, it will cause *X* to become <*m-pepsican, (), null*>. When the token sequence becomes empty, the target is activated.

3.6 Connecting Procedural and Declarative Knowledge through Callbacks

Since the robot acts in the world, the robot's declarative knowledge must be connected to the robot's procedural knowledge represented with RAPs. The two types of knowledge are connected through *callbacks*. A callback is an arbitrary procedure that runs when the MOP with which it is associated is *activated*. The MOP activation algorithm is as follows:

1. Map a speech input into a token sequence.
2. For each token *T* in the token sequence, advance all expectations that have *T* as their key.
3. For each expectation whose token sequence is empty, activate the target MOP.
4. For each activated MOP and for each callback associated with the MOP, run the callback.

This algorithm is lazy and failure-driven in that the robot never initiates an interaction with the operator unless asked to achieve a goal. This distinguishes our approach from several dialogue-based, mixed-initiative approaches where autonomous agents act *proactively* (Rich et al., 2001; Sidner et al., 2006). We conjecture that when the robot and the operator share common goals, the complex problems of generic language understanding, dialogue state tracking and intent recognition required in proactive approaches can be avoided. The algorithm has the robot pursue a goal when the goal's MOP is activated by the input. Such a

MOP is made an instance of *m-active-goal*. When the goal is achieved, the goal becomes an instance of *m-achieved-goal*, and is delinked from *m-active-goal*. When the MOP associated with the goal is partially referenced by the input, i.e., no token sequences associated with the goal's MOP are advanced to the end, it becomes an instance of *m-ambiguous-goal*.

In sum, to be operational, the system must have the following pieces of a priori knowledge: 1) the semantic network of goals and objects; 2) the library of object models; 3) the context-free command and control grammars for speech recognition; 4) the library of RAP behaviors; and 5) the library of robotic skills. The DMAP-Net is constructed manually and manipulated and modified by the robot at run time. The library of object models is constructed semi-automatically. The designer must specify the region of an image containing an object. From then on everything is done automatically by the system. The context-free command and control grammars are constructed manually. However, construction of such grammars from semantic networks may be automated (Kulyukin & Steele, 2002). The initial RAP behaviors and skills are also programmed manually.

This knowledge representation ties language and vision both top-down and bottom-up. Language and vision are tied top-down, because the semantic network links each object representation with its models used by object recognition skills. Thus, the robot's memory directly connects symbolic and visual types of knowledge. The bottom-up connection is manifested by the fact that each object recognition skill in the control tier has associated with it a set of predicate templates that are instantiated with specific values extracted from the image and asserted into the robot's memory.

4. Human-Robot Interaction

The robot and the operator can gradually achieve mutual understanding with respect to a specific activity only if they share a common set of goals. It is the common cognitive machinery that makes human-robot dialogue feasible. Since the operator knows the robot's goals, he or she can adjust language inputs to suit the robot's level of understanding. Since the robot knows the goals to be referenced by the operator, the robot's dialogue is based on language pattern recognition instead of language understanding, the latter being a much harder problem.

4.1 Mapping Phonemes to Symbols

Speech is recognized with Microsoft Speech API (SAPI) 5.1, which is freely available from www.microsoft.com/speech. SAPI is a software layer that applications use to communicate with SAPI-compliant speech recognition (SR) and text-to-speech (TTS) engines. SAPI includes an Application Programming Interface (API) and a Device Driver Interface (DDI). SAPI 5.1 includes the Microsoft English SR Engine Version 5, an off-the-shelf state-of-the-art Hidden Markov Model speech recognition engine. The engine includes 60,000 English words, which we found adequate for our purposes. Users can train the engine to recognize additional vocabularies for specific domains. SAPI couples the Hidden Markov Model speech recognition with a system for constraining speech inputs with context-free command and control grammars. The grammars constrain speech recognition sufficiently to eliminate user training and provide speaker-independent speech recognition. Grammars are defined with XML Data Type Definitions (DTDs).

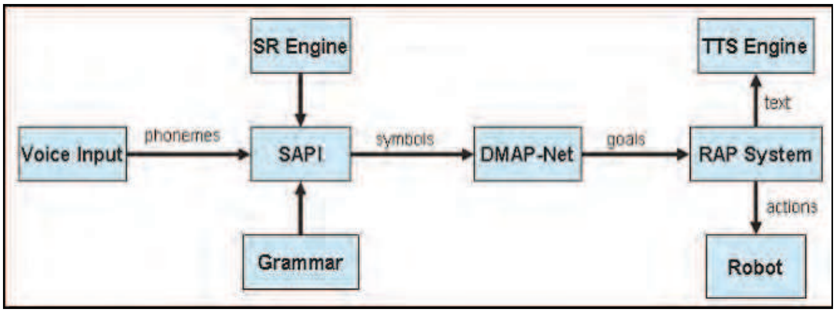


Fig. 5. Language recognition process.

The system receives a stream of phonemes, from the operator. SAPI converts the stream of phonemes into a sequence of tokens. For example, if the operator says "turn left twenty," SAPI converts it into the token sequence *(turn left twenty)*. The symbolic sequence is sent to the semantic network. The spreading activation algorithm uses the symbolic sequence to activate MOPs. In this case, the MOP *m-turn-command* is referenced. When this MOP is referenced, the callback associated with it runs and installs the following goal *(achieve (turn-left 20))* on the RAP system's agenda. The RAP system finds an appropriate RAP and executes it on the robot. The robot issues speech feedback to the user through a text-to-speech engine (TTS). Figure 5 shows the language recognition process.

4.2 Four Types of Interaction

The algorithm outlined in Section 3.6 allows for four types of interaction that the operator and the robot engage in through spoken dialogue: command, goal disambiguation, introspection, and instruction-based learning. The selection of the interaction types is not accidental. If an autonomous robot is to effectively communicate with a human operator in natural language, the robot must be able to understand and execute commands, disambiguate goals, answer questions about what it knows, and, when appropriate, modify its behavior on the basis of received language instructions. In our opinion, these types of interaction are fundamental to the design and implementation of autonomous systems capable of interacting with humans through spoken dialogue. We now proceed to cover each interaction type.

4.2.1 Command

Command is the type of interaction that occurs when the operator's input references a goal in the robot's memory unambiguously. For example, when the operator asks the robot to pick up a pepsican in front of the robot, the robot can engage in the pickup behavior right away. Command is a basic minimum requirement for human-robot dialogue-based interaction in that the robot must at least be able to execute direct commands from the operator.

Let us go through an example. Consider the left image in Figure 6. Suppose the operator's command to the robot at this point is "Get the pepsican." One of the token sequences associated with *m-get-phys-obj* is *(get-verb-slot phys-obj-slot)*. To activate this MOP, an input must first activate its verb slot and then its physical object slot. The *m-get*'s token sequence is *(get)*, thus the token *get*, the first token extracted from the operator's command, activates *m-get*, which, through spreading activation, activates *m-get-verb* . The token sequence of the

MOP's expectation is advanced one token to the right and becomes *(phys-obj-slot)* . The tokens obtained from "the pepsican" activate *mpepsican* and, since *m-phys-obj* is an abstraction of *m-pepsican*, *phys-obj-slot* is activated. When *m-pepsican* is activated, the object models for pepsicans are brought into the robot's memory through a callback associated with *m-pepsican* unless the models are already in memory due to the previous activities.

After *m-phys-obj* is activated, the token sequence of the MOP's expectation becomes empty, thus activating the MOP. When the MOP is activated, an instance MOP, call it *m-get-phys-obj-12*, is created under it with values *m-get* for the verb slot and *m-pepsican* for the object slot. The new MOP is placed under *m-active-goal* to reflect the fact that it is now one of the active goals pursued by the robot. As soon as *m-get-phys-obj* is activated, its callback installs *(achieve m-get-phys-obj)* on the RAP interpreter's agenda. The RAP for achieving this goal checks for instances of the MOP *m-get-phys-obj* that are also instances of *m-active-goal*. In this case, the RAP finds *m-get-phys-obj-12*, extracts the *m-pepsican* value from the MOP's object slot and installs *(get-phys-obj m-pepsican)* on the RAP interpreter's agenda. During execution, the RAP enables *detect-obj-skill*. The skill captures the image shown on the left in Figure 6, detects the pepsican (the right image in Figure 6 shows the detection region whitened), and puts the following assertions in the RAP memory: *(detected-obj pepsican)*, *(dist-to 1.5)*, *(obj-xy 81 43)*, and *(sim-score .73)*. These assertions state that the skill detected a pepsican 1.5 meters away from the robot, and the bottom left coordinates of the image region that had the best matching score of .73 are *x=81*, *y=43*. The *get-phys-obj* RAP invokes the homing skill. The skill causes the robot to navigate to the object. When the robot is at the object, the RAP enables the pickup skill so that the robot can pick up the can with the gripper. Finally, the RAP removes *m-get-phys-obj-12* from under *m-active-goal* and places it under *m-achieved-goal*.

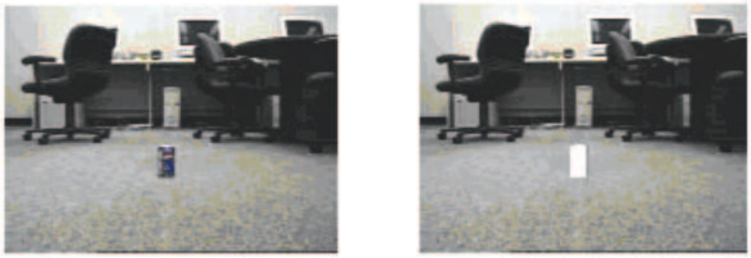


Fig. 6. Command Execution.

4.2.2 Goal Disambiguation

The system is designed to handle three types of goal ambiguity: *sensory*, *mnemonic*, and *linguistic*. Sensory ambiguity occurs when a goal referenced by the input is ambiguous with respect to sensory data. Mnemonic ambiguity takes place when the input references a goal ambiguous with respect to the robot's memory organization. Linguistic ambiguity arises when the linguistic input itself must be disambiguated to reference a goal.

A goal is ambiguous with respect to sensory data when it refers to an object from a detected homogeneous set or when it refers to an abstraction common to all objects in a detected heterogeneous set. To understand sensory ambiguity, consider a situation when the robot has detected two pepsicans and the operator asks the robot to bring a pepsican. The homogeneous set of detected objects consists of two pepsicans. The input refers to a pepsican, i.e., an object from the detected homogeneous set. All things being equal, the robot always chooses to act on objects closest to it. Consequently, the robot disambiguates the input to pick up the closest pepsican, i.e., the closest object from the detected homogeneous set. This disambiguation strategy does not require any human-robot dialogue.

A goal is mnemonically ambiguous when it references an abstraction with multiple specifications. To understand mnemonic ambiguity, consider a situation when the robot receives the input “Get a soda can” after it has detected one pepsican and two root beer cans. In this case, the set of detected objects is heterogeneous. The operator's input references an abstraction common to all objects in the set, because each object is an instance of *m-soda-can*. Since there is no way for the robot to guess the operator's intent with respect to the type of soda can, the robot initiates a dialogue to have the operator commit to a specific object type in the detected set. In this case, the robot asks the operator what type of soda can it needs to pick up.

The difference between sensory ambiguity and mnemonic ambiguity is that the former is *post-visual*, i.e., it occurs with respect to objects already detected, i.e., after the sensing action has been taken, whereas the latter is *pre-visual*, i.e., it occurs before a sensing action is taken. To appreciate the difference, consider a situation when the operator asks the robot to find a soda can. Assume that the robot does not have any sensory data in its memory with respect to soda cans. The input references an abstraction, i.e., *m-soda-can*, that, according to Figure 3, has two specifications: *m-pepsican* and *m-rootbeer-can*. Thus, the robot must disambiguate the input's reference to one of the specifications, i.e., choose between finding a pepsican or a root beer can. This decision is pre-visual.

Mnemonic ambiguity is handled autonomously or through a dialogue. The decision to act autonomously or to engage in a dialogue is based on the number of specifications under the referenced abstraction: the larger the number of specifications, the more expensive the computation, and the greater the need to use the dialogue to reduce the potential costs. Essentially, in the case of mnemonic ambiguity, dialogue is a cost-reduction heuristic specified by the designer. If the number of specifications under the referenced abstraction is small, the robot can do a visual search with respect to every specification and stop as soon as one specification is found.

In our implementation, the robot engages in a dialogue if the number of specifications exceeds two. When asked to find a physical object, the robot determines that the number of specifications under *m-phys-obj* exceeds two and asks the operator to be more specific. On the other hand, if asked to find a soda can, the robot determines that the number of specifications under *m-soda-can* is two and starts executing a RAP that searches for two specification types in parallel. As soon as one visual search is successful, the other is terminated. If both searches are successful, an arbitrary choice is made.

One type of linguistic ambiguity the robot disambiguates is *anaphora*. Anaphora is a pronoun reference to a previously mentioned object (Jurafsky & Martin, 2000). For example, the operator can ask the robot if the robot sees a pepsican. After the robot detects a pepsican and answers affirmatively, the operator says “Go get it.” The pronoun *it* in the operator's input is a case of anaphora and must be resolved. The anaphora resolution is based on the recency of reference: the robot searches its memory for the action MOP that was most recently created and resolves the pronoun against the MOP's object. In this case, the robot finds the most recent specification under *m-detect-object* and resolves the pronoun to refer to the action's object, i.e., a pepsican.

Another type of linguistic ambiguity handled by the system is, for lack of a better term, called *delayed reference*. The type is detected when the input does not reference any goal MOP but references an object MOP. In this case, the input is used to disambiguate an active ambiguous goal. The following heuristic is used: an input disambiguates an active ambiguous goal if it references the goal's object slot. For example, if the input references *m-color* and an active ambiguous goal is *m-get-soda-can*, whose object slot does not have the color attribute instantiated, the referenced *m-color* is used to instantiate that attribute. If the input does not refer to any goal and fails to disambiguate any existing goals, the system informs the operator that the input is not understood. To summarize, the goal disambiguation algorithm is as follows:

- Given the input, determine the ambiguity type.
- If the ambiguity type is sensory, then
  - o Determine if the detected object set is homogeneous or heterogeneous.
  - o If the set is homogeneous, choose the closest object in the set.
  - o If the set is heterogeneous, disambiguate a reference through dialogue.
- If the ambiguity type is mnemonic, then
  - o Determine the number of specifications under the referenced abstraction.
  - o If the number of specifications does not exceed two, take an image and launch parallel search.
  - o If the number of specifications exceeds two, disambiguate a reference through dialogue.
- If the ambiguity type is linguistic, then
  - o If the input has an impersonal pronoun, resolve the anaphora by the recency rule.
  - o If the input references an object MOP, use the MOP to fill an object slot of an active ambiguous MOP.
- If the ambiguity type cannot be determined, inform the operator that the input is not understood.

4.2.3 Introspection and Instruction-Based Learning

Introspection allows the operator to ask the robot about it state of memory. Introspection is critical for ensuring the commonality of cognitive machinery between the operator and the robot. Our view is that, given the current state of the art in autonomous robotics and natural language processing, it is not feasible to build robots that can adapt or evolve their cognitive machinery to match the level of cognition of humans with whom they collaborate. While this remains the ultimate long-term objective of cognitive robotics, robust solutions are needed for the interim. Introspection may be one such solution, because it is based on the flexibility of human cognitive machinery. It is much easier for the operator to adjust his or her cognitive machinery to the robot's level of cognition than vice versa. Introspection is also critical for exploratory learning by the operator. For, if an application does not provide introspection, learning is confined to input-output mappings.

One important assumption for introspection to be successful is that the operator must know the robot's domain and high-level goals. Introspective questions allow the operator to examine the robot's memory and arrive at the level of understanding available to the robot. For example, the operator can ask the robot about physical objects known to the robot. When the robot lists the objects known to it, the operator can ask the robot about what it can do with a specific type of object, e.g., a soda can, etc. The important point here is that introspection allows the operator to acquire sufficient knowledge about the robot's goals and skills through natural language dialogue. Thus, in principle, the operator does not have to be a skilled robotic programmer or have a close familiarity with a complex GUI.

```
(define-rap
  (index (dump-trash ?dist ?speed))
  (method
   (task-net
    (sequence
     (t1 (robot-deploy-gripper-prim))
     (t2 (robot-move-prim 0 ?speed
                          ?dist))
     (t3 (robot-back-up-prim ?speed
                             ?dist))
     (t4 (robot-store-gripper-prim))
     ))))
```

```
(define-rap
  (index (dump-trash ?dist ?speed))
  (method
   (task-net
    (sequence
     (t1 (robot-deploy-gripper-prim))
     (t2 (robot-move-prim 0 ?speed
                          ?dist))
    (parallel
     (t3 (robot-back-up-prim ?speed
                             ?dist)
        (t4 (robot-store-gripper-prim)))
    ))))
```

Fig. 7. Introspection and Learning.

We distinguish three types of introspective questions: questions about past or current state (What are you doing? Did/Do you see a pepsican?), questions about objects (What objects do you know of?), and questions about behaviors (How can you pick up a can?). Questions about states and objects are answered through direct examination of the semantic network or the RAP memory. For example, when asked what it is doing, the robot generates natural language descriptions of the active goals currently in the semantic network. Questions about behaviors are answered through the examination of appropriate RAPs. Natural language generation is done on the basis of descriptions supplied by the RAP programmer. We extended the RAP system specification to require that every primitive RAP be given a short description of what it does. Given a RAP, the language generation engine recursively looks up descriptions of each primitive RAP and generates the RAP's description from them. For example, consider the left RAP in Figure 7. When asked the question “How do you dump trash?”, the robot generates the following description: “Deploy the gripper. Move forward a given distance at a given speed. Back up a given distance at a given speed. Store the gripper.”
Closely related to introspection is instruction-based learning (Di Eugenio, 1992). In our system, this type of learning occurs when the operator instructs the robot to modify a RAP on the basis of the generated description. The modification instructions are confined to a small subset of standard plan alternation techniques (Riesbeck & Schank, 1989): the removal of a step, insertion of a step, and combination of several steps. For example, given the above description from the robot, the operator can issue the following instruction to the robot: “Back up and store the gripper in parallel.” The instruction causes the robot to permanently modify the last two tasks in the *dump-trash* RAP to run in parallel. The modified rap is given on the right in Figure 7. This is an example of a very limited ability that our system gives to the operator when the operator wants to modify the robot’s behavior through natural language.

5. Experiments

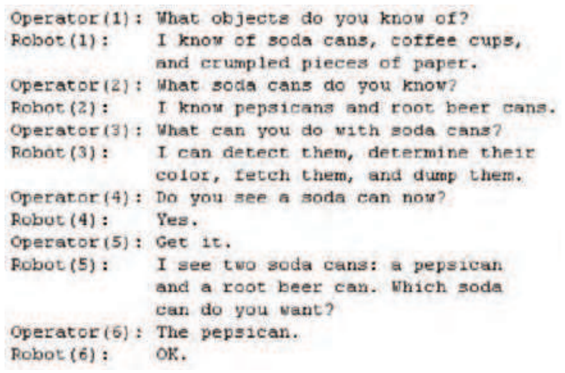
Our experiments focused on command and introspection. We selected a set of tasks, defined the success and failure criteria for each task type, and then measured the overall percentage of successes and failures. There were four types of tasks: determination of an object's color, determination of proximity between two objects, detection of an object, and object retrieval. Each task required the operator to use speech to obtain information from the robot or to have the robot fetch an object. The color task would start with questions like “Do you see a red soda can?” or “Is the color of the coffee cup white?” Each color task contained one object type. The proximity task would be initiated with questions like “Is a pepsican near a crumpled piece of paper?” Each proximity task involved two object types. The proximity was calculated by the robot as a thresholded distance between the detected object types mentioned in the input. The object detection task would be initiated with questions like "Do you see a soda can?" or "Do you not see a root beer can?" and mentioned one object type per question. The object retrieval task would start with commands to fetch objects, e.g., "Bring me a pepsican," and involved one object type per task.

The experiments were done with two operators, each giving the robot 50 commands in each of the task categories. Hence, the total number of inputs in each category is 100. In each experiment, the operator and the robot interacted in the local mode. The operator would issue a voice input to the robot and the robot would respond. Both operators, CS graduate students, were told about the robot's domain, i.e., but did not know anything about the robot's knowledge of objects and actions, and were instructed to use introspection to obtain that knowledge. Introspection was heavily used by each operator only in the beginning. Speech recognition errors were not counted. When the speech recognition system failed to recognize the operator's input, the operator was instructed to repeat it more slowly. Both operators were told that they could say the phrase "What can I say now?" at any point in the dialogue. The results of our experiments are summarized in Table 1. The columns give the task types.

Color	Proximity	Detection	Retrieval
78	81	84	85
100	100	100	92

Table 1. Test Results.

The color, proximity, and detection tasks were evaluated on a binary basis: a task was a success when the robot gave a correct yes/no response to the question that initiated the task. The responses to retrieval tasks were considered successes only when the robot would actually fetch the object and bring it to the operator.



```
Operator(1): What objects do you know of?
Robot(1):    I know of soda cans, coffee cups,
             and crumpled pieces of paper.
Operator(2): What soda cans do you know?
Robot(2):    I know pepsicans and root beer cans.
Operator(3): What can you do with soda cans?
Robot(3):    I can detect them, determine their
             color, fetch them, and dump them.
Operator(4): Do you see a soda can now?
Robot(4):    Yes.
Operator(5): Get it.
Robot(5):    I see two soda cans: a pepsican
             and a root beer can. Which soda
             can do you want?
Operator(6): The pepsican.
Robot(6):    OK.
```

Fig. 8. Initial dialogue.

The first row of the table gives the total number of successes in each category. The second row contains the number of accurate responses, given that the vision routines extracted the correct sets of predicates from the image. In effect, the second row numbers give an estimate of the robot's understanding, given the ideal vision routines. As can be seen the first row numbers, the total percentage of accurate responses, was approximately 80 percent. The primary cause of object recognition failures was radio signal interference in our laboratory.

One captured image distorted by signal interference is shown in Figure X. The image contains a white coffee cup in the center. The white rectangle in the bottom right corner shows the area of the image where the vision routines detected an object. In other words, when the radio signal broadcasting the captured image was distorted, the robot recognized non-existing objects, which caused it to navigate to places that did not contain anything or respond affirmatively to questions about the presence or absence of objects.
The percentage of successes, given the accuracy of the vision routines, was about 98 percent. Several object retrieval tasks resulted in failure due to hardware malfunction, even when the vision routines extracted accurate predicates. For example, one of the skills used by the *fetch-object* RAP causes the robot to move a certain distance forward and stop. The skill finishes when the robot's speed is zero. Several times the wheel velocity readers kept returning non-zero readings when the robot was not moving. The robot would then block, waiting for the moving skill to finish. Each time the block had to be terminated with a programmatic intervention. The object recognition skills were adequate to the extent that they supported the types of behaviors in which we were interested.
Several informal observations were made during the experiments. It was observed that, as the operators' knowledge of the robot's memory increased, the dialogues became shorter. This appears to support the conjecture that introspection is essential in the beginning, because it allows human operators to quickly adjust their cognitive machinery to match the cognitive machinery of the robotic collaborator. Figure 8 shows a transcript of an initial dialogue between the robot and an operator.

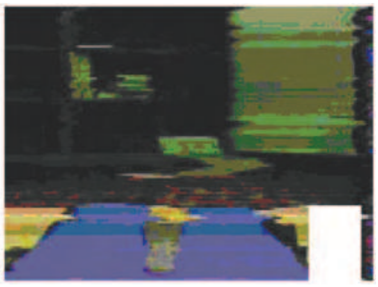


Fig. 9. Signal interference.

6. Related Work

Our work builds on and complements the work of many researchers in the area of human-robot interaction. Chapman's work on Sonja (Chapman, 1991) investigated instruction use in reactive agents. Torrance (1994) investigated natural language communication with a mobile robot equipped with odometry sensors. Horswill (1995) integrated an instruction processing system with a real-time vision system. Iwano et al. (1996) analyzed the relationship between semantics of utterances and head movements in Japanese natural language dialogues and argued that visual information such as head movement is useful for managing dialogues and reducing vague semantic references. Matsui et al. (1999) integrated a natural spoken dialogue system for Japanese with their Jijo-2 mobile robot for office services. Fitzgerald and Firby (2000) proposed a reactive task architecture for a dialogue system interfacing to a space shuttle camera.
Many researchers have investigated human-robot interaction via gestures, brainwaves, and teleoperation. Kortenkamp et al. (1996) developed a vision-based gesture recognition system

that resides on-board a mobile robot. The system recognized several types of gestures that consisted of head and arm motions. Perzanowski et al. (1998) proposed a system for integrating natural language and natural gesture for command and control of a semi-autonomous mobile robot. Their system was able to interpret such commands as “Go over there” and “Come over here.” Rybski and Voyles (1999) developed a system that allows a mobile robot to learn simple object pickup tasks through human gesture recognition. Waldherr et al. (2000) built a robotic system that uses a camera to track a person and recognize gestures involving arm motion. Amai et al. (2001) investigated a brainwave-controlled mobile robot system. Their system allowed a human operator to use brain waves to specify several simple movement commands for a small mobile robot. Fong et al. (2001) developed a system in which humans and robots collaborate on tasks through teleoperation-based collaborative control. Kruijff et al. (2006) investigated clarification dialogues to resolve ambiguities in human-augmented mapping. They proposed an approach that uses dialogue to bridge the difference in perception between a metric map constructed by the robot and a topological perspective adopted by a human operator. Brooks and Breazeal (2006) investigated a mechanism that allows the robot to determine the object referent from deictic reference including pointing gestures and speech.

Spoken natural language dialogue has been integrated into several commercial robot applications. For example, the newer Japanese humanoids, such as Sony's SDR and Fujitsu's HOAP, have some forms of dialogue-based interaction with human users. According to Sony, the SDR-4X biped robot is able to recognize continuous speech from many vocabularies by using data processing synchronization with externally connected PCs via wireless LANs (www.sony.com). Fujitsu's HOAP-2 humanoid robot, a successor to HOAP-1, can be controlled from an external PC, and is claimed to be able to interact with any speech-based software that the PC can run.

Our approach is complementary to approaches based on gestures, brain waves, and GUIs, because it is based on gesture-free spoken language, which is phenomenologically different from brain waves, gestures, and GUIs. From the perspective of human cognition, language remains a means of choice for introspection and instruction-based learning. Although one could theorize how these two interaction types can be achieved with brain waves, gestures, and GUIs, the practical realization of these types of interaction is more natural in language-based systems, if these systems are developed for users with visual and motor disabilities.

Dialogue-management systems similar to the one proposed by Fong et al. (2001) depend in critical ways on GUIs and hardware devices, e.g., Personal Digital Assistants, hosting those GUIs. Since these approaches do not use natural language at all, the types of human-robot interaction are typically confined to yes-no questions and numerical template fill-ins.

Our approach rests, in part, on the claim that language and vision, at least in its late stages, share memory structures. In effect, it is conjectured that late vision is part of cognition and, as a consequence, closely tied with language. Chapman (1991) and Fitzgerald and Firby (2000) make similar claims about shared memory between language and vision but do not go beyond simulation in implementing them. Horswill's robot Ludwig (Horswill, 1995) also grounds language interpretation in vision. However, Ludwig has no notion of memory and, as a system, does not seem to offer much insight into memory structures used by language and vision.

Our approach proposes computational machinery for introspection, i.e., a way for the robot to inspect its internal state and, if necessary, describe it to the operator. For example, the robot can describe what it can do with respect to different types of objects and answer questions not only about what it sees now, but also about what it saw in the past. Of the above approaches, the systems in Torrance (1994), Perzanowski et al. (1998), and Fitzgerald and Firby (2000) could, in our opinion, be extended to handle introspection, because these approaches advocate elaborate memory organizations. It is difficult to see how the reactive approaches advocated by Chapman (1991) and Horswill (1995) can handle introspection, since they lack internal states. GUI-based dialogue-management approaches (Fong et al., 2001) can, in principle, also handle introspection. In this case, the effectiveness of introspection is a function of GUI usability.

7. Limitations and Future Work

Our investigation has several limitations, each of which, in and of itself, suggests a future research venue. First, the experimental results should be taken with care due to the small number of operators. Furthermore, since both operators are CS graduate students, their performance is unlikely to generalize over people with disabilities.

Second, our system cannot handle deictic references expressed with pronouns like “here,” “there,” “that,” etc. These references are meaningless unless tied to concrete situations in which they are used. As Perzanowski et al. (1998) point out, phrases like “Go over there” are meaningless if taken outside of the context in which they are spoken. Typically, resolving deictic references is done through gesture recognition (Brooks & Breazeal, 2006). In other words, it is assumed that when the operator says “Go over there,” the operator also makes a pointing gesture that allows the robot to resolve the deictic reference to a direction, a location, or an object (Perzanowski et al., 1998). An interesting research question then arises: how far can one go in resolving deictic references without gesture recognition? In our experiments with a robotic guide for the visually impaired (Kulyukin et al., 2006), we noticed that visually impaired individuals almost never use deictic references. Instead, they ask the robot what it can see and then instruct the robot to go to a particular object by naming that object. This experience may generalize to many situations in which the operator cannot be at or near the object to which the robot must navigate. Additionally, there is reason to believe that speech and gesture may have a common underlying representation (Cassell et al., 1999). Thus, the proposed knowledge representation mechanisms may be useful in systems that combine gesture recognition and spoken language.

Third, our system cannot handle universal quantification and negation. In other words, while it is possible to ask the system if it sees a popcan, i.e., do existential quantification, it is not possible to ask questions like “Are all cans that you see red?” Nor is it currently possible for the operator to instruct the system not to carry out an action through phrases like “Don't do it.” The only way for the operator to prevent the robot from performing an action is by using the command “Stop.”

Fourth, our system can interact with only one user at a time. But, as Fong et al. (2001) show, there are many situations in which multiple users may want to interact with a single robot. In this case, prioritizing natural language inputs and making sense of seemingly contradictory commands become major challenges.

Finally, our knowledge representation framework is redundant in that the RAP descriptions are stored both as part of RAPs and as MOPs in the DMAP-Net. This creates a knowledge maintenance problem, because every new RAP must be described in two places. This redundancy can be eliminated by making the DMAP-Net the only place in the robot’s memory that has the declarative descriptions of RAPs. In our future work, we plan to generate the MOPs that describe RAPs automatically from the RAP definitions.

8. Conclusion

We presented an approach to human-robot interaction through gesture-free spoken dialogue. The two contributions of our investigation are a taxonomy of goal ambiguities (sensory, mnemonic, and linguistic) and the implementation of a knowledge representation framework that ties language and vision both top-down and bottom-up. Sensory ambiguity occurs when a goal referenced by the input is ambiguous with respect to sensory data. Mnemonic ambiguity takes place when the input references a goal ambiguous with respect to the robot's memory organization. Linguistic ambiguity arises when the linguistic input itself must be disambiguated to reference a goal. Language and vision are tied top-down, because the semantic network links each object representation with its models used by object recognition skills.

Our investigation is based on the view that, given the current state of the art in autonomous robotics and natural language processing, it is not feasible to build robots that can adapt or evolve their cognitive machinery to match the level of cognition of humans with whom they collaborate. While this remains the ultimate long-term objective of cognitive robotics, robust solutions are needed for the interim. In our opinion, these interim solutions are knowledge representation and processing frameworks that enable the robot and the operator to gradually achieve mutual understanding with respect to a specific activity only if they share a common set of goals. It is the common cognitive machinery that makes human-robot dialogue feasible. Since the operator knows the robot's goals, he or she can adjust language inputs to suit the robot's level of understanding. Since the robot knows the goals to be referenced by the operator, the robot's dialogue is based on language pattern recognition instead of language understanding, the latter being a much harder problem.

9. Acknowledgments

The research reported in this chapter is part of the Assisted Navigation in Dynamic and Complex Environments project currently under way at the Computer Science Assistive Technology Laboratory (CSATL) of Utah State University. The author would like to acknowledge that this research has been supported, in part, through NSF CAREER grant (IIS-0346880) and two Community University Research Initiative (CURI) grants (CURI-04 and CURI-05) from the State of Utah. The author would like to thank the participants of the 2005 AAAI Spring Workshop on Dialogical Robots at Stanford University for their comments on his presentation “Should Rehabilitation Robots be Dialogical?”

10. References

Amai, W.; Fahrenholtz, J. & Leger, C. (2001). Hands-Free Operation of a Small Mobile Robot. *Autonomous Robots*, 11(2):69-76.

Bonasso, R. P.; Firby, R. J.; Gat, E.; Kortenkamp, D. & Slack, M. 1997. A Proven Three- tiered Architecture for Programming Autonomous Robots. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(1):171-215.

Bookstein, A.; Kulyukin, V. & Raita, T. (2002). Generalized Hamming Distance. *Information Retrieval*, 5:353-375.

Boykov, Y. & Huttenlocher, D. (1999). A New Bayesian Framework for Object Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society.

Brooks, G. & Breazeal, C. (2006). Working with Robots and Objects: Revisiting Deictic References for Achieving Spatial Common Ground. *Proceedings of the ACM Conference on Human-Robot Interaction (HRI-2006)*. Salt Lake City, USA.

Cassell, J.; McNeill, D. & McCullough, E. (1999). Speech-Gesture Mismatches: Evidence for One Underlying Representation of Linguistic and Non-linguistic Information. *Pragmatics and Cognition*, 7(1):1-33.

Chang, P. & Krumm, J. (1999). Object Recognition with Color Co-occurrence Histograms. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society.

Chapman, D. (1991). *Vision, Instruction, and Action*. MIT Press: New York.

Cheng, G. and Zelinsky, A. (2001). Supervised Autonomy: A Framework for Human- Robot Systems Development. *Autonomous Robots*, 10(3):251-266.

Di Eugenio, B. (1992). Understanding Natural Language Instructions: The Case of Purpose Clauses. *Proceedngs of 30th Annual Meeting of the Association for Computational Linguistics (ACL-92)*, pp. 120-127, Newark, USA.

Firby, R. J.; Prokopowicz, P. & Swain, M. (1995). Collecting Trash: A Test of Purposive Vision. In Proceedings of the Workshop on Vision for Robotics, International Joint Conference on Artificial Intelligence, Montreal, Canada, AAAI Press.

Firby, R. J. (1989). *Adaptive Execution in Complex Dynamic Worlds*. Unpublished Ph.D. Dissertation, Computer Science Department, Yale University.

Fitzgerald, W. & Firby, R. J. (2000). Dialogue Systems Require a Reactive Task Architecture. *Proceedings of the AAAI Spring Symposium on Adaptive User Interfaces*, Stanford University, USA, AAAI Press.

Fong, T. & Thorpe, C. (2001). Vehicle Teleoperation Interfaces. *Autonomous Robots*, 11(2):9-18.

Fong, T.; Thorpe, C. & Baur, C. (2001). Collaboration, Dialogue, and Human-Robot Interaction. *Proceedings of the 10th International Symposium of Robotics Research*, Lorne, Victoria, Australia.

Hainsworth, D. W. (2001). Teleoperation User Interfaces for Mining Robotics. *Autonomous Robots*, 11(1):19-28.

Horswill, I. (1995). Integrating Vision and Natural Language without Central Models. *Proceedings of the AAAI Fall Symposium on Emboddied Language and Action*, MIT, USA, AAAI Press.

Iwano Y.; Kageyama S.; Morikawa E., Nakazato, S. & Shirai, K. (1996). Analysis of Head Movements and its Role in Spoken Dialogue. *Proceedings of the International Conference on Spoken Language Processing (ICSLP-96)*, Vol. 4, pp. 2167-2170, Philadelphia, PA, 1996.

Kortenkamp, D.; Huber, E. & Bonasso, P. (1996). Recognizing and Interpreting Gestures on a Mobile Robot. *Proceedings of the AAAI/IAAI Conference*, Vol. 2, pp. 915-921, Portland, USA.

Kortenkamp, D. & Schultz, A. (1999). Integrating Robotics Research. *Autonomous Robots*, 6(3):243-245.

Kulyukin, V. & Steele, A. (2002). Input Recognition in Voice Control Interfaces to Three-Tiered Autonomous Agents. *Proceedings of the International Lisp Conference, Association of Lisp Users*, San Francisco, CA.

Kulyukin, V. (2004). Human-Robot Interaction through Gesture-Free Spoken Dialogue. *Autonomous Robots*, 16(3): 239-257.

Kulyukin, V.; Gharpure, C.; Nicholson, J. & Osborne, G. (2006). Robot-Assisted Wayfinding for the Visually Impaired in Structured Indoor Environments. *Autonomous Robots*, 21(1), pp. 29-41.

Kulyukin. (2006). On Natural Language Dialogue with Assistive Robots. *Proceedings of the ACM Conference on Human-Robot Interaction (HRI 2006)*, Salt Lake City, USA.

Krikke, J. (2005). Robotics research exploits opportunities for growth. *Pervasive Computing*, pp. 1-10, July-September.

Kruijff, G.J.; Zender, H.; Jensfelt, P. & Christensen, H. (2006). Clarification Dialogues in Human-Augmented Mapping. *Proceedings of the ACM Conference on Human-Robot Interaction (HRI 2006)*, Salt Lake City, USA.

Jurafsky, D & Martin, J. (2000). *Speech and Language Processing*. Prentice-Hall, Inc. Upper Saddle River, New Jersey.

Lane, J C.; Carignan, C. R. & Akin, D. L. (2001). Advanced Operator Interface Design for Complex Space Telerobots. *Autonomous Robots*, 11(1):69-76.

Martin, C. (1993). *Direct Memory Access Parsing*. Technical Report CS93-07, Computer Science Department, The University of Chicago.

Matsui, T.; Asah, H.; Fry, J.; Motomura, Y.; Asano, F.; Kurita, T.; Hara, I. & Otsu, N. (1999). Integrated Natural Spoken Dialogue System of Jijo-2 Mobile Robot for Office Services. *Proceedings of the AAAI Conference*, Orlando, FL, pp. 621-627.

Montemerlo, M.; Pineau, J.; Roy, N.; Thrun, S. & Verma, V. (2002). Experiences with a mobile robotic guide for the elderly. *Proceedings of the Annual Conference of the American Association for Artificial Intelligence (AAAI)*, pp. 587-592. AAAI Press.

Parker, J. R. (1993). *Practical Computer Vision Using C*. John Wiley and Sons: New York.

Perzanowski, D.; Schultz, A. & Adams, W. (1998). Integrating Natural Language and Gesture in a Robotics Domain. *Proceedings of the IEEE International Symposium on Intelligent Control: ISIC/CIRA/ISAS Joint Conference*. Gaithersburg, MD: National Institute of Standards and Technology, pp. 247-252.

Roman, S. (1992). *Coding and Information Theory*. Springer-Verlag: New York.

Rich, C., Sidner, C., and Lesh, N. (2001). COLLAGEN: Applying Collaborative Discourse Theory to Human-Computer Interaction. *AI Magazine*, 22(4):15-25.

Riesbeck, C. K. and Schank, R. C. (1989). Inside Case-Based Reasoning. Lawrence Erlbaum Associates: Hillsdale.

Rybski, P. and Voyles, R. (1999). Interactive task training of a mobile robot through human gesture recognition. *Proceedings of the IEEE International Conference on Robotics and Automation*, Detroit, MI, pp. 664-669.

Sidner, C.; Lee, C.; Morency, L. & Forlines, C. (2006). The Effect of Head-Nod Recognition in Human-Robot Conversation. Proceedings of the ACM Conference on Human-Robot Interaction (HRI-2006). Salt Lake City, USA.

Spiliotopoulos. D. (2001). Human-robot interaction based on spoken natural language Dialogue. *Proceedings of the European Workshop on Service and Humanoid Robots*, Santorini, Greece.

Swain, M. J. and Ballard, D. H. (1991). Color Indexing. *International Journal of Computer Vision*, 7:11-32.

Torrance, M. (1994). *Natural Communication with Robots*. Unpublished Masters Thesis, MIT.

Yanco, H. (2000). *Shared User-Computer Control of a Robotic Wheelchair System*. Ph.D. Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA.

Young, S.; Scott, P.D.; & Nasrabadi, N. (1994). Multi-layer Hopfield Neural Network for Object Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society.

Waldherr, S.; Romero, R. & Thrun, S. (2000). A Gesture Based Interface for Human- Robot Interaction. *International Journal of Computer Vision*, 9(3):151-173.