Indexical Grounding for a Mobile Robot

by

Charles W. Kehoe

S.B. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2004

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

June 2005

Signature of Author: _____
Department of Electrical Engineering and Computer Science
May 19, 2005

Certified by: _____
Deb Kumar Roy
Associate Professor of Media Arts and Sciences
Thesis Supervisor

Accepted by: _____
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Indexical Grounding for a Mobile Robot

by Charles W. Kehoe

Submitted to the Department of Electrical Engineering and Computer Science
on May 19, 2005, in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

We have outfitted a mobile research robot with several sensors and algorithms designed to facilitate small- and large-scale navigation and natural language interaction. We begin with a parser using a large, hand-crafted English grammar and lexicon. We then add a standard gradient navigation algorithm for local obstacle avoidance, and a line segment comparison algorithm for basic, high-performance location recognition. The result is a full end-to-end system for natural-language-driven, mobile robotics research.

The theme of grounding—mapping linguistic references to the corresponding real-world entities—runs throughout our approach. After the parser simplifies linguistic symbols and structures, we must connect them to the basic concepts that they represent, and then to our system's specific sensor readings and motor commands, to make natural language interaction possible. Additionally, many of the symbols we must ground are indexicals with critical contextual dependencies. We must therefore handle the implicit context that spatial communication carries with it.

Thesis Supervisor: Deb Kumar Roy
Title: Associate Professor of Media Arts and Sciences

Table of Contents

Introduction

Over the past few decades, information technology has begun to permeate our lives at an incredible rate. Not long ago, cell phones were a rare luxury, and the Internet was still largely a research network connecting our nation's top academic institutions with government research labs. In today's world, both of these complex technologies are nearly ubiquitous. From refrigerators to automobiles, from keychains to houses, all sorts of everyday objects increasingly feature built-in digital electronics that make them easier, more powerful, safer, or more fun for their owners to use.

Despite the phenomenal progress that computer technology has made, however, most aspects of it remain surprisingly artificial. The real world is not made up of the menus, dialog boxes, scroll bars, and mouse pointers one finds on the typical computer desktop. Why do we still use this paradigm to communicate with the electronic devices that are increasingly surrounding us? Why do we insist on interacting with technology on artificial terms, rather than on human terms? In most cases, the answer is a simple matter of development costs, and a lack of interest in pursuing more natural interfaces.

Yet progress is being made towards a more familiar, human feel in the behavior of computer systems. In the new field of human-computer interaction, researchers are extending the well-established graphical user interface paradigm, and new interfaces based on gestures, virtual reality, and natural language are rapidly developing. Solutions to more difficult problems, such as rational thought, emotional awareness, and intentionality, are also taking shape. There is yet hope for a time when "learning to use the computer" will be a forgotten concept, and interacting with technology will be as easy as interacting with our friends and colleagues.

Our contribution to more natural human-computer interaction is a mobile, autonomous robot that navigates its environment with sonar, recognizes locations with a laser scanner, and responds in a natural way to simple linguistic interactions with humans. A basic model of English grammar and a lexicon of about a hundred words are sufficient to make the natural language interface workable. Our robot is programmed as an assistant, its primary purpose being to help out and obey its users. Much like a human assistant, it maintains evolving records of the world around it and its goals in that world. It remembers where it has been, and can recognize those locations later. It also adopts a simple strategy for navigating the world around it, finding paths to various goals, and avoiding obstacles along the way. Though we have no hope of making interaction with the robot completely natural, we have managed to come close in several simple situations.

In short, our robot responds in a natural way to English instructions such as "Please turn to the right" and "Can you move forwards?", or questions such as "When did you last turn around?". As is typical with natural language, these simple sentences mix complexity and ambiguity (mostly due to the context upon which they depend) with superfluous information (often resulting from cultural or personal factors). To properly interpret utterances like these, our robot must make explicit the speaker's hidden assumptions about social roles, obstacle avoidance, and frame of reference (What does "turn right" mean? Whose right? Yours? Who is "you", anyway?). At the same time, it must find and discard verbal fluff such as "can you" or "please", and differentiate questions from commands based on the way these features are encoded in English syntax. It is complications like these that make it incredibly difficult (and interesting) to build robust natural language interfaces for even the simplest command sets. However, we feel that our system has made promising steps in this direction.

Once it decides on the core meaning of each input utterance, our system must carry out its master's requests as best it can. To do this, we combine a few simple, but powerful, sensors and algorithms. A set of sonar sensors connected to a standard gradient-based navigation algorithm are used for local navigation (path finding and obstacle avoidance). These sensors are relatively reliable, but slow and low-resolution, even when data is accumulated over several scans. A laser scanner produces higher-resolution environment images, which are stored and compared against each other to accomplish basic place recognition. These images, though dense and high-speed, are subject to interference and other artifacts that make them less usable for local navigation. Together, these strategies provide a surprisingly robust navigation system.

Dubbed "Hermes", our robot was originally intended to act as an electronic guide and messenger. Though much more work needs to be done before robots like these are safe and useful in the real world, we feel that the platform we have developed shows promise for several important and exciting future applications. A tour guide, for example, need only recognize several simple places, navigate between them (potentially with some sort of audio playback involved), avoid obstacles, and respond to simple English queries. This skill set is (on a basic level) mostly present in our system. Another potential application is an interoffice mail or package delivery agent, which must be able to perform similar tasks. One can imagine, as well, adding a simple NLP interface like ours to one of the many robotic vacuum cleaners that are already on the market today. Among other things, this would make it much easier to specify when and where it should vacuum. Eventually, we hope to extend our work to automobiles, where simple voice command and automatic navigation technologies are already making inroads in the consumer space. If sufficiently safe and robust, intelligent voice-controlled navigation systems may well become as ubiquitous as the cell phone.

Background

Grounding, natural language understanding, and mobile navigation are all relatively popular and well-studied areas, but to date, there has not been widespread interest in combining ideas from these domains into a full end-to-end system. A few mobile robotics systems have been built to accept natural language instructions, focusing on such things as task learning or object handling [14, 15, 21]. Our research group has many projects that center around grounding natural language, but since they have used stationary robots, spatial language has been less of an issue [5, 8, 10, 17-20]. None of our previous projects have placed emphasis, as we do now, on path finding and linguistic flexibility. This is, in part, because so many separate issues are involved. In the paragraphs that follow, we provide an overview of these issues, and discuss how they relate to our approach.

Grounding is at the heart of some of the most difficult problems we face in artificial intelligence today. Harnad mentions the essential conceptual issues that grounding addresses, but offers no concrete solution to what he calls "The Symbol Grounding Problem" [6]. Other cognitive psychologists have attempted to flesh out the nature of the connections between linguistic symbols and real-world entities, but few have provided enough detail to construct a full computational model of the process [1, 22]. Grounding *indexicals*, or symbols whose meaning depends on their context ("you", "this", "here", "left", "now", etc.), can be particularly difficult. Unfortunately, indexicals are often at the heart of communication. In fact, Pylyshyn suggests that visual indexes may be built in to the human vision system [16]. This issue poses additional challenges for a robot that, like ours, attempts to accept relatively unconstrained language. By providing a standard frame of reference, we have been able to smooth out some of these issues. However, much remains to be done before we can participate in truly grounded, situated communication.

Natural language understanding is another critical piece of many puzzles in modern artificial intelligence research. Indeed, many people feel that natural language is, in some complex way, inherently tied to human intelligence [2, 3]. In our attempts to build intelligent systems, then, we will find many helpful clues in the structure and usage of natural language. One such clue is the correspondence between structure and reasoning; just as verbal reasoning is represented by the structure of a sentence, so spatial reasoning requires imposing structure on the physical world around us. Reasoning (and meaning, to which it is closely tied) is made possible by structure. Once that structure—verbal or spatial—has been found, many modifications, additions, and substitutions may fine-tune the meaning without really changing it. Accordingly, our natural language interface attempts to extract the basic structure from utterances it receives, in order to discover their central meaning. Extra information can then be more easily separated and interpreted or discarded as appropriate.

Intelligent mobile navigation is a critical research area today, thanks in part to accelerating development and use of mobile technology. From vacuum cleaners to Mars rovers, video games to air traffic control, a wide variety of applications are now clamoring for robust automatic navigation through complex terrains. (Our focus has been more on automatic, language-controlled navigation rather than on handling complex environments.) Much of this research, however, still relies on map-based navigation with GPS and equivalents. While this approach is accurate if done correctly, it requires the setup, and use of a wireless position triangulation system, which is an additional hassle and expense. Our approach has been to try recognizing locations, and remembering how those locations are related to each other by motor commands. This has an intuitive appeal, mainly because of its similarity to the methods that people use to navigate everyday environments.

System Overview

Our mobile robot is a Pioneer 2 from ActivMedia Robotics, equipped with 2-wheel differential drive, 12V onboard batteries, sonar, bump sensors, and a 2 degree of freedom gripper. To this standard configuration, we have added a SICK laser distance sensor and a Panasonic web server camera. All of our sensor output and robot control input (over RS232) is routed to the wireless network through the onboard laptop, allowing the robot to be physically self-contained. Local obstacle avoidance is handled on the laptop, while one or more nearby workstations handle place recognition, natural language interaction, and user feedback. Eventually, all of our computation will be moved onto the robot, adding additional laptops as needed, but for now, separate servers make it easier to add new features as they become available. All systems are Intel-based, running recent versions of RedHat Enterprise or Fedora Core Linux.

A block diagram of our system appears in figure 1 on the next page. As shown, responding to user requests involves several steps. First, the syntactic parser organizes the text into one or more syntactic parse trees. The grammar parser then walks the syntactic parse tree, simplifying it into a tree that shows which words serve which function in the sentence. Next, the grounder walks the grammar parse tree, connecting sentence structure and content to the appropriate sensor and motor primitives. Finally, these primitives are sent to the local navigation engine, which carries them out as best it can. At any time, the place memory can be asked to remember or forget one or more locations. Based on the output of the place analyzer (in turn based on laser scanner data), it then provides a confidence metric for each of its remembered locations, indicating its similarity to the current location. The complete system can easily be run on a single desktop, with the help of a wireless network and the robot's onboard laptop.

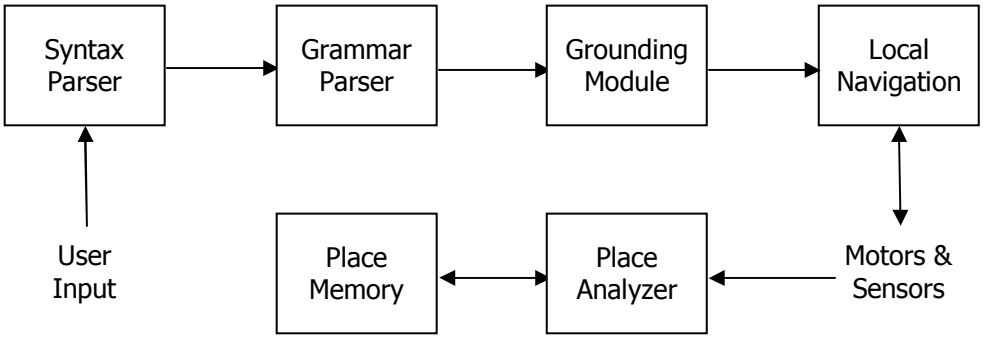


Figure 1: Overall System Architecture. User input goes through several stages of processing before finally affecting the robot itself.

Since our system, like many of the robots in our group, has several pieces that can operate independently, we use a publish-and-subscribe system based on PVM (Parallel Virtual Machine) to link the pieces together [4]. This library abstracts TCP sockets into named network "channels", to which a program can subscribe, or on which it can publish. However, PVM is designed for high-performance parallel computing, and therefore leaves many low-level details exposed. Since this is not needed for our application, we use a new library called NetP (and NetPacket, an earlier version of it) for data formatting and abstraction of the network communication details [9]. In this way, we can easily (if needed) pipe sensor from a single program, running on one computer, to several programs, running on several other computers. Similarly, several programs on different computers can send motor commands to the same program running in the same place, with minimal concern for overlapping and interleaving problems. Nor do we have to concern ourselves with port numbers (and conflicts), or data formatting and flow control. NetP makes it easy to link pieces of a system across different machines.

Our overall system, then, is designed to be as modular, extensible, and portable as possible. Its various pieces can be turned on and off as desired, with minimal effects on the others. Next, we turn to a discussion of each of the major functional groups in the system.

Natural Language Interface

Much like an assistant, our system simply takes English instructions, one at a time, and executes them. Input processing begins with two parsing steps, shown below in figure 2. First, the user types a command in plain English text. (Eventually we expect to use a speech recognizer instead.) The syntax parser then converts this string into a tree, showing the syntactic structure of the sentence according to its hand-crafted CFG of English. The tree shows how meaning is encoded in the syntax, but provides excess detail. Accordingly, the grammar parser walks through the syntax parse, extracting the subject, the verb, and their modifiers. These are then assembled into a much simpler tree, in which (almost) every node is a word, and modifiers are children of the words they modify. It is this simplified parse tree that the grounding module traverses, in order to extract the basic meaning of the sentence and ignore irrelevant extra information.

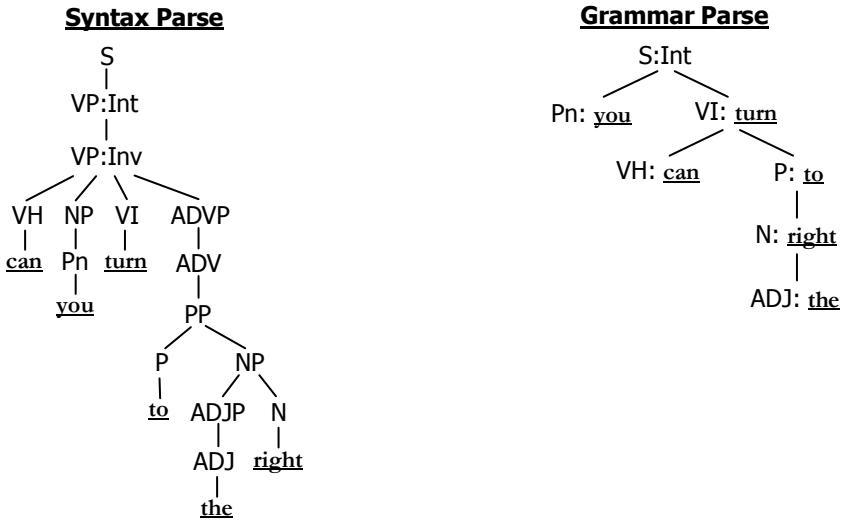


Figure 2: Input Parsing of "Can you turn right?". Due to its rather general CFG, the syntax parser produces heavily nested structures. These trees are then condensed by the grammar parser to reveal logical dependencies.

Our parser is based on a custom 600-word English lexicon with built-in grammatical information. The parser automatically generates standard plurals, participles, past forms, and so forth from the base form provided, and adds these words to the lexicon. For irregular words, alternative or missing forms can be specified as needed. Pronoun forms are listed in a dedicated section of the lexicon file, so that they can be recognized correctly. Each form is tagged with its number, person, case, gender, and/or tense as appropriate, and this information is used later on by the grammar parser and grounding module.

We perform syntax parsing with a custom bottom-up predictive chart parser, similar to an Earley parser, and based on an older parser implementation by Peter Gorniak. We add words to the chart one a time as individual constituents, looking up their part of speech in our lexicon, and producing an error if the word is not found. If the part of speech is found, we search the nearly 400 rules in our handcrafted English CFG, and predict every rule whose tail begin with the new word's part of speech. We also advance any rules expecting that part of speech at that particular point. If any rule advances to completion, we apply the same prediction and advancement procedures to it recursively, just as they were applied to the original word. In this way, we complete larger and larger constituents, until one or more full sentences are found. An example of this parsing process appears below in figure 3.

Context-Free Grammar: S → NP VP NP → ART N VP → VL ADJ

The	**ball**	**is**	**heavy**
Complete ART → The	*Complete* N → ball	*Complete* VL → is	*Complete* ADJ → heavy
Predict NP → ART N	*Complete* NP → ART N	*Predict* VP → VL ADJ	*Complete* VP → VL ADJ
	Predict S → NP VP		*Complete* S → NP VP

Figure 3: Bottom-up predictive parse of "The ball is heavy". Our custom parser allows us a great deal of flexibility in how we design the grammar, but is also designed for high performance, and can locate islands of grammaticality.

One useful feature of our custom parser is its support for type flags. These appear in the grammar as a slash followed by a letter after the main constituent type. Thus, we can differentiate subject noun phrases (NP/S) from object noun phrases (NP/O) by the type of pronouns they contain (Pn/S or Pn/O). This causes "he is a robot" to parse correctly, while "him is a robot" fails. This same feature allows us to restrict where a noun or verb phrase can be complex (i.e. contain more than one noun or verb), differentiate between participial phrases and other types of adjectives, check certain verb forms, and so forth.

As a proof of concept, our grammar and lexicon were designed to handle a wide variety of user input, even though much of the necessary grounding machinery was never implemented. Thus, in addition to the typical noun and verb phrases, our grammar handles adverbs, prepositional phrases, and participial/gerund phrases, along with relative and noun clauses. Hence inputs such as "Show me the place that we visited yesterday" and "When did I see you running across the hall" are correctly parsed, but not grounded, whereas simpler sentences like "Now turn to the right" and "Can you come back here" and handled fully. In accordance with our theme of connecting structure to meaning, it is our expectation that specifying the correct structure of these sentences in the grammar will make it much easier to ground them correctly later on.

Once the syntax parser has generated one or more possible syntactic structures for the input sentence, the grammar parser walks through the resulting tree, pruning and collapsing it into a more intelligible form. As shown in figure 2, our flexible, recursive English grammar produces a significant number of extraneous nodes that make the parse tree difficult to handle. Simplifying it into a more intuitive tree structure, showing the relation between words rather than syntactic units, is the purpose of the grammar parser. This process makes the grounding module easier to write, and helps its results to be more accurate.

The grammar parser uses several heuristics based on the relation between English syntax and meaning, and is too complex to describe in precise detail here. Predictably, it takes a recursive approach to interpreting the hierarchical syntax trees with which it is presented. The following paragraphs provide a conceptual description of the grammar parsing process.

Every grammar parse begins with a clause, which may in turn contain other clauses. Therefore, the top-level clause is broken down, if necessary, into simple clauses. A simple clause is identified as containing a verb, rather than other clauses. Each such clause is then parsed individually, and the resulting clause hierarchy is reflected in the grammar parse tree.

We perform a breadth-first search of each simple clause, to locate its top-level subject, verb, object, and adjective phrases. To make sure that these phrases are top-level, we do not allow searches for noun phrases to recurse into verb phrases, and vice-versa. This prevents, for instance, the object of a participial or gerund phrase from appearing as an object of the main verb. Thus, in the sentence "robots like running marathons", the gerund phrase "running marathons" is parsed as a noun phrase, and identified as the direct object, rather than the noun "marathons" (the object of the gerund). Similarly, verbs from a noun clause are prevented from appearing as the main verb. Hence, in the sentence "What you did is not acceptable", the noun clause "What you did" is parsed as a noun phrase, and identified as the subject; this forces the main verb to appear as "is" instead of "did" (the verb from the noun clause). The results of these searches are then used to record the top-level structure of the clause (subject, verb, plus direct and indirect object, if any). For imperative sentences, the implied subject "you" is automatically supplied.

These top-level noun, verb, and modifier phrases are then broken down, if necessary, into simple phrases. Thus, compound noun and verb phrases are split apart and parsed separately. Next, the main word (or clause) in each phrase is located; for noun phrases, this is

a single-word noun, or noun clause, and for verb phrases, it is a single-word verb that is not a verb helper (i.e. "can", "did", "have", etc.). A similar process applies for modifier phrases. The resulting phrase hierarchy, if any, is reflected in the grammar parse tree.

All members of modifier phrases are then made children of the words they modify. Thus, noun phrases containing adjective phrases are reorganized so that the adjectives involved are children of the main noun, rather than cousins (see figure 2). Modifier phrases are searched in breadth-first fashion, as before, to extract all of their single-word members, and locate verb phrases (participles) or clauses (relative) which must be handled separately.

Throughout this process, the grammar parse tree is built mostly out of words rather than syntactic units. Thus, the children of a clause are the simple subject, verb, object, and so forth. The children of the verbs and nouns are their modifiers (words or clauses). Other than clauses, compound verbs, and so forth, all of the nodes in a grammar parse tree are words.

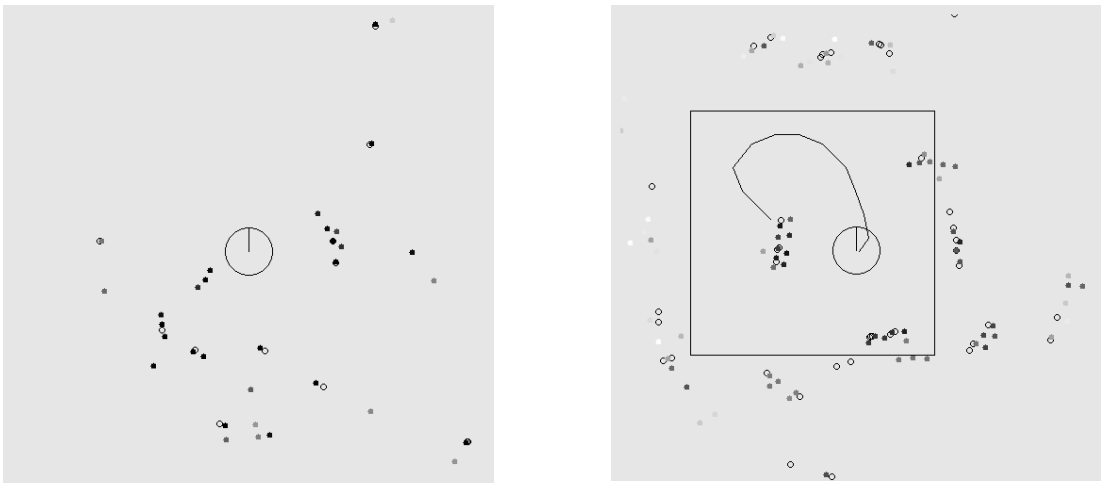
The grounding process is then based on the grammar parse. Imperative sentences (S:Imp) are treated as commands, as are interrogative sentences (S:Int) with subject "you" and verb helper "can". In these cases, the verb is extracted, and its concept tag (along with relevant modifiers) is mapped to a motor primitive. "Turn" and "rotate" both have the same concept tag; so do the words "move", "go", and "come". Thus "turn" with the modifier "right" results in a 90-degree clockwise rotation, while "move" with modifier "backward" results in setting a local navigation goal directly behind the robot. The modifier "little" will cause a 90-degree rotation to become 45 degrees instead (e.g. "Please rotate a little to the left."). Other modifiers, such as "please", "now", and "next", are ignored. "Stop" and "stay" are both mapped to movement termination. Questions are currently recognized, but not answered; likewise statements are recognized, but not handled. Thus "When did you last turn right?" or "People move backward" will not trigger an action.

The issue of context plays a central role in these grounding processes. Inherent in most of the utterances we process is an understanding that the user and the robot are in the same room, interacting with the same space. The context also involves the issue of social roles; in this case, the user and robot are set up to assume a master-servant relationship. Additionally, there is an unstated assumption that the user wants the robot to avoid collisions with nearby obstacles (this is the goal of the local navigation engine). But just as critically, terms such as "right" or "forwards" depend on the orientation of the robot, the term "you" depends on who is being addressed, and the term "last" implies a temporal context. In order to interpret utterances like these, we must provide the robot with a standard frame of reference, so that (for example) "you" is mapped to the robot, and directions are relative to the robot's current orientation. If we later add history support, we must also keep track of what is meant by "now", and how it relates to events in the past.

Building an effective natural language interface is a difficult problem, but we feel that we have made important progress here. We have explored many parser enhancements that make it easier to write effective CFGs for English, and use them to recognize both full sentences and islands of grammaticality with real-time performance. We have defined a framework for representing traditional grammar information in a parsing framework, and have used it to perform such tasks as subject-verb agreement checking, and sentence type identification. Our parser far exceeds the capabilities needed for our particular application, and shows promise for building an extendable, sophisticated conversational system with relatively high logical coherency. We have exposed much of the structure of simple discourse, laying a solid foundation for building a symbolic reasoning framework on top of it, and making yet another step towards a natural conversation system.

Local Navigation

Next, we consider the local navigation system. The sonar sensors accumulate obstacle data over time, as shown below in figure 4. We navigate around these obstacles by setting a goal, and then plotting a path to that goal. The path-finding algorithm uses a standard gradient-based approach [11].



(A) (B)

Figure 4: Obstacle avoidance with sonar. Our robot accumulates sonar readings over time (A), and uses them to plot paths around local obstacles (B). This technique allows us to avoid most collisions.

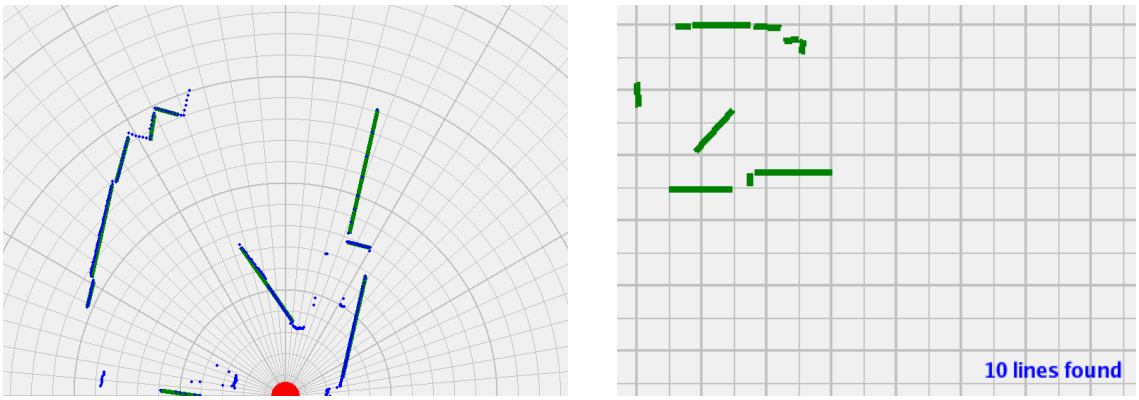
The obstacle data shown above consists of several types of data points. Current and recent sonar points are displayed and considered in navigation, but points that move or change quickly are less likely to be remembered. If a sonar point is persistent enough, it is assumed to be a wall, and is remembered more or less permanently. Stale sonar data is adjusted to coincidence more accurately with current readings. Additionally, obstacles that the sonar cannot see (bumps on the floor, chair legs, etc.) are also remembered if they cause the robot to stall (i.e. attempt to move in a certain direction, but fail to turn the wheels).

These obstacle points are then fed into a gradient navigation algorithm similar to [11]. Conceptually, gradient navigation has a relatively intuitive parallel: we simulate a hilly landscape, where Konolige's navigation cost can be thought of as the elevation. Obstacles then appear as mountains, and the goal represented by a valley. We fill in the elevation (cost) of each intermediate point, so that navigation reduces to simulating the robot as a ball rolling through the terrain, constantly moving to the adjacent point of lowest elevation, until it reaches the goal (or a local minimum). Since our obstacles (sonar data) and goal location can both potentially change at any time, we must continually recalculate the gradient field, which can be computationally quite expensive. However, for the amount of data we have, and the terrains we have been navigating, we have not found this to be a significant problem.

With this system in place, we can navigate in a certain direction by setting our goal a small distance away in that direction and constantly updating the goal so that it remains a constant distance away. As the robot rotates, we must change the goal direction to keep our overall direction of motion constant. Depending on the parameters, this approach can have its own share of problems (i.e. repeatedly steering towards a wall, then backing up, only to try again), but it does a relatively good job of local obstacle avoidance. Larger-scale navigation will eventually be handled by our place-recognition system.

Place Recognition

Our local navigation system allows us to move in a single direction at a time without colliding with obstacles. This is a useful navigation primitive, but hardly an effective global navigation strategy. As we start to consider larger places (i.e. on the scale of a building), we must start to remember individual locations, and the paths among them. This is currently the main purpose of the onboard laser distance scanner, which provides much higher-resolution images of the robot's surrounding much faster than is possible with sonar. A few samples of the scanner's (processed) output appear in figure 5 below. We convert the 361 radial distance samples to Cartesian coordinates, and then use a simple linear-least-squares algorithm to locate line segments in the resulting data. Since many indoor environments tend to have straight walls, this is an easy way to simplify the data for efficient storage and processing. In order to normalize over small rotational differences, we locate the longest segment, and rotate to make that segment horizontal. This usually puts the scene in one of four characteristic orientations, which are then easy to recognize.



(A) (B)

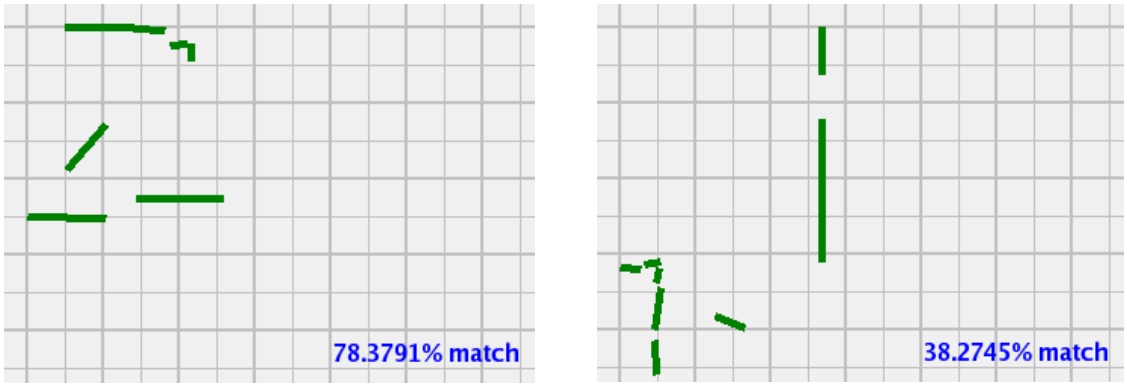
Figure 5: Laser scanner processing. The laser scanner outputs a set of radial distances from a fixed point (A). We then locate line segments in this data, and normalize the segments into a standard orientation (B).

The line-fitting process involves a number of parameters that control its strictness and accuracy. A clean, graphical interface makes it possible to adjust these parameters in real time, and observe the results. The most important parameters are the noise threshold, the maximum distance between data points on a line segment, the minimum line segment length, and the minimum number of data points that can constitute a line segment. These should be set low enough so that extraneous data is not reported, but high enough so measurement errors do not prevent valid data from being recognized. Thus, their optimal values depend mostly on the precision of the laser scanner, and the properties of the locations we used.

The noise threshold and maximum point distance were mostly adjusted to coincide with the accuracy of the scanner itself. We found, for instance, that a noise threshold (tolerance) of 20 to 30 mm was usually sufficient to recognize the same line segments our own eyes perceived in the data, without being either too messy or too picky. The maximum distance between points on a segment was used to screen out segments that occurred by coincidence, without enough evidence to back them up. We usually set this parameter around 500 mm, but often pulled it as low as 300 mm. There were many heuristic tradeoffs involved here.

The other two parameters were adjusted more in accordance with the peculiarities of our test locations. It is our hope that these locations constituted a reasonable sample of the "typical" locations our robot would encounter, but they will probably still need fine-tuning later on. Specifically, we set the minimum segment length somewhere between 100 and 200 mm, and required that a prospective line segment must be supported by at least 5 data points before it is recognized. Setting these parameters too low would cause every tiny artifact to generate a line segment, which then made recognition much less reliable. Setting them too high would prevent some important line segments from being picked up. Again, plenty of heuristic tradeoffs were involved.

Once we have decided on the important line segments in a data set, we normalize the line segments for rotation, as discussed previously. We then store the result, and compare it with later rotation-normalized sets, rotating to all four characteristic orientations as necessary. To do this, we first correct for shifting along each axis, by finding all pairs of parallel segments in different sets, and averaging their displacements, weighted by the product of the segment lengths. Segments that are more vertical are compared by their x-intercepts, and more horizontal segments by their y-intercepts. Segments are considered "parallel" if their angles match to within an adjustable tolerance, usually 5%. We then shift the segments by the resulting average x and y displacements, and repeat the process until corresponding segments are only a small distance apart; usually this takes about five iterations. (For our purposes, this "small distance" was defined as about 10 mm or less, below the noise threshold used to extract the segments in the first place.) Finally, we calculate the percentage of segment length that is overlapping between the two segment sets, or "locations", and output this percentage as our confidence score. After repeating this process for all four major orientations, we take the highest score as our final product. Sample results of the location-matching algorithm are shown below in figure 6.



(A) (B)

Figure 6: Place recognition. Given the current location depicted above in figure 3, our place recognizer shows high confidence for similar locations (A), and low confidence for less similar ones (B).

For the data shown here, our results are relatively convincing. The location-matching algorithm works best with the typical long, right-angled walls that appear in hallways and less-crowded offices; as long as some line segments can be picked out, the approach shows surprising robustness. On the other hand, heavily cluttered locations, or those without perpendicular, straight walls, are often quite problematic. To handle these, we will eventually need to pick out geometric primitives other than just line segments, or just resort to clustering the points in some intelligent way.

Another common complication with our place recognizer is due to field-of-view changes. Though we do normalize for rotation, the fact still remains that parts of the robot's former view are cut out when the scene is rotated, due to the laser's 180-degree scanning field. This results in lower matching scores, since part of the expected overlap is missing. There are a couple of potential ways to deal with this problem. One involves storing a less than a full 180-degree field of view to recognize against, so that missing wall segments from minor rotations are not penalized. Another calls for rescaling the confidence score to be non-linear, so that high scores are further boosted, and low scores drop further. Eventually, however, we will want to begin stitching together multiple 180-degree view fields, which will make place recognition much more robust than is currently possible. Without this approach, recognition depends as much on location as on orientation, and the robot loses much of its recognition ability every time it turns 90 degrees.

In short, our place recognition algorithm provides a solid foundation for true, robust memory and classification of the locations among which our robot might navigate. Though it currently has some limitations, it presents a reasonable, high-performance solution to a difficult problem. With some additional work, it should be possible to turn it into a useful and reliable navigation system.

Results and Evaluation

We have combined a few simple, but powerful, sensors and algorithms with a hand-crafted parser, grammar, and lexicon to produce a full end-to-end system for natural-language-driven, mobile robotics research. We have explored some indexical grounding issues with common words like "you" and "left", and have built a modest place recognition system that will eventually allow us to ground such indexicals as "this place" or "that hallway". Addition of a simple action history will eventually enable us to ground such phrases as "yesterday" or "an hour ago". We have reached the causal level of Kuiper's Spatial Semantic Hierarchy, and hope to move to the topological level soon [12]. We are making exciting progress.

Hermes: What shall I do next?
User: Can you come forwards?
Hermes: I'm going forward.
What shall I do next?
User: Stay right there.
Hermes: I'm stopping.

Hermes: What shall I do next?
User: Please rotate a little to the right.
Hermes: I'm turning right.
What shall I do next?
User: Now move slowly back.
Hermes: I'm going backward.

Figure 7: Sample dialogs. For a variety of sentence structures, Hermes does a decent job of selecting the core meaning, and acting on it.

Figure 7 above shows some sample dialogs with Hermes, whose responses indicate how he has interpreted each input sentence. As the astute reader will note, even the basic set of queries we support involves a significant range of problems in natural language understanding. These sorts of common problems appear in countless different applications, most of which have nothing to do with mobile, autonomous robots. Indeed, it is our hope that some of the lessons learned in this effort will be applicable to a wide range of computer systems, enabling them to interact with humans more naturally than is currently possible.

Future Work

Hermes offers an exciting look at what becomes possible when flexible natural language processing is combined with an intelligent, multi-tiered navigation strategy. Indeed, we feel that the greatest strength of our work is not what it actually achieves, but what its rich, modular framework may make possible for future projects. In particular, we offer the following list as a starting point for building on the Hermes platform:

1. *History support*. Actions should be recorded as they occur, so that (among other things) questions about past actions can then be answered.
2. *Place aggregation and naming*. Currently, we remember individual laser views, but give them no name, and do no grouping to link views that are collected in nearby locations. Fixing these issues would make place recognition more useful.
3. *Place linking*. Once we have a history, and places are aggregated, we can start to link places based on movements that take the robot from one place to another. Chaining known locations together makes large-scale navigation possible.
4. *Spatial reasoning*. With linking of named places available, we can start to explore spatial reasoning queries, such as "How far is the office from the kitchen?", or "How do I get to the lobby from here?", or "What is left of my office?".
5. *Natural interaction*. With the addition of a speech recognizer and synthesizer, as well as some other basic abilities, the above skillset is probably enough to make users feel comfortable treating Hermes as a child rather than a robot. This natural human-computer interaction is the ultimate goal of our research.

Acknowledgements

This work would not have been possible without generous contributions of advice and support from many of my friends and colleagues here at MIT. First and foremost, I would like to thank my advisor, Deb Roy, for suggesting, guiding, and supporting the project throughout the past year. I also extend my sincere gratitude to Peter Gorniak and Kai-Yuh Hsiao, whose mentorship, patience, and assistance proved invaluable during my work on this project. Without their support, and that of the rest of the Cognitive Machines Group here at the MIT Media Lab, this project would simply not have made any progress.

I am also indebted to many others for their hard work on several parts of this project, and I extend my sincerest gratitude for their contributions of time and energy. I owe most of the low-level robot setup and sonar navigation system to Dany Qumsiyeh, whose hard work throughout the summer of 2004 produced a solid framework upon which the rest of the project was built. Kai-Yuh Hsiao and Peter Gorniak contributed critical parts of the network architecture and natural language interface. Jonathan Wang's assistance was invaluable in a variety of ways, including setting up and debugging the laser scanner interface. And Stefanie Tellex was always eager to help out with any miscellaneous issues I encountered.

Without the support and inspiration provided by countless others, both here at MIT and back home, I would never have had the opportunity to begin this work in the first place. I have been fortunate enough to study with some excellent teachers over the past several years, and I will always be grateful for their consistent hard work and selfless contributions. My friends in high school and college have been similarly supportive and caring, making the past several years far more memorable and rewarding than I ever expected. But I owe my deepest debt of gratitude to my parents, whose selfless devotion and constant support have never failed to turn my dreams into reality. Their love will stay with me forever.

References

[1] Elizabeth Bates. *The Emergence of Symbols*. Academic Press, 1979.

[2] Daniel Dennett. The Role of Language in Intelligence. In *What is Intelligence?*, The Darwin College Lectures, ed. Jean Khalfa. Cambridge: Cambridge Univ. Press. 1994.

[3] Merlin Donald. *Origins of the Modern Mind: Three Stages in the Evolution of Culture and Cognition*. Cambridge, MA: Harvard Univ. Press, 1991.

[4] A. Geist et al. *PVM—Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing*. Cambridge, MA: MIT Press, 2004.

[5] Peter Gorniak and Deb Roy. Grounded Semantic Composition for Visual Scenes. *Journal of Artificial Intelligence Research*, 21:429–470, 2004.

[6] Stevan Harnad. *The symbol grounding problem*. Physica D, 42:335–346, 1990.

[7] Gerd Herzog and Peter Wazinski. VIsual TRAnslator: Linking Perceptions and Natural Language Descriptions. *Artificial Intelligence Review*, 8:175–187, 1994.

[8] Kai-yuh Hsiao, Nikolaos Mavridis, and Deb Roy. Coupling perception and simulation: Steps towards conversational robotics. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, 2003.

[9] Kai-Yuh Hsiao, Peter Gorniak and Deb Roy. NetP: A Network API for Building Heterogeneous Modular Intelligent Systems. To appear in *Proceedings of the AAAI-05 Workshop on Modular Construction of Human-Like Intelligence*, 2005.

[10] Joshua Juster and Deb Roy. Elvis: Situated Speech and Gesture Understanding for a Robotic Chandelier. In *Proceedings of the International Conference on Multimodal Interfaces*, 2004.

[11] Kurt Konolige. A Gradient Method for Realtime Robot Control. In *Proceedings of the IEEE/RSJ International Conference on Intelligent 7Robots and Systems*, 2000.

[12] Benjamin Kuipers. *The Spatial Semantic Hierarchy*. Artificial Intelligence, 119:191–233, 2000.

[13] Barbara Landau and Ray Jackendoff. "What" and "Where" in Spatial Language and Cognition. *Behavioral and Brain Sciences*, 16: 217-265, 1993.

[14] Stanislao Lauria et al. Training Personal Robots Using Natural Language Instruction, *IEEE Intelligent Systems*, pages 38–45, Sept./Oct. 2001.

[15] L. Seabra Lopes and A. Teixeira. Human-Robot Interaction through Spoken Language Dialogue. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

[16] Zenon W. Pylyshyn. Situating Vision in the World. *Trends in Cognitive Sciences*, 4(5):197-207, 2000.

[17] Deb Roy. *Learning Words from Sights and Sounds: A Computational Model*. PhD thesis, Massachusetts Institute of Technology, 1999.

[18] Deb Roy. Learning Visually-Grounded Words and Syntax for a Scene Description Task. *Computer Speech and Language*, 16(3), 2002.

[19] Deb Roy, Peter Gorniak, Niloy Mukherjee, and Josh Juster. A Trainable Spoken Language Understanding System for Visual Object Selection. In *International Conference of Spoken Language Processing*, Denver, 2002.

[20] Deb Roy and Niloy Mukherjee. Visual Context Driven Semantic Priming of Speech Recognition and Understanding. *Computer Speech and Language*, In press.

[21] Marjorie Skubic et al. Spatial Language for Human-Robot Dialogs. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 34(2):154-167, 2004.

[22] Leonard Talmy. How Language Structures Space. In *Spatial Orientation: Theory, Research, and Application*, ed. H. Pick & L. Acredelo. Plenum Press, 1983.

[23] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Autonomous Robots*, 5(3-4):253-271, 1998.