Relational Interface for a Voice Controlled Wheelchair

Stefanie Tellex

May 17, 2005

Abstract

I have built a voice controlled simulated wheelchair. The system understands and obeys natural language motion commands such as "Take a right." It tries to build a relationship with the user by disclosing its internal state when it realizes it is failing at a task. I describe the system, and an evaluation

1 Introduction

Traditional joystick interfaces enable analog control of wheelchairs. Alternate interfaces exist for digital control, but existing interfaces do not obey commands in a context-sensitive way. [Fehr et al., 2000] did a survey of clinicians, asking them to give information about their patients and the usefulness of new powerchair technology. They found that 9 to 10 % of patients who receive powerchair training "find it extremely difficult or impossible to use the wheelchair for activities of daily living." A voice controlled wheelchair that understands high level commands can increase the mobility options for such people. Such a chair will be used constantly by the user, so there exists an opportunity for it to benefit by establishing a relationship with its user. If the human feels that they are in an alliance with the chair, they are more likely to help it through mistakes and not grow frustrated or angry with it.

An example interaction with the system follows:

User Go forward.

Robot begins moving forward, following the corridor, avoiding obstacles as necessary.

User Take a right.

Robot stops for about 2 seconds to compute its next goal, and moves towards the first opening in the wall on its right.

User Go left.

Robot stops for about 2 seconds to compute its next goal, and moves towards the first opening in the wall on its left.

User Stop.

Robot makes a confused beep because speech recognition failed.

User Stop.

Robot continues moving forward because speech recognition failed.

User Stop.

Robot makes a sad beep because of continued failure to understand, and continues moving forward.

User Whoa.

Robot makes a happy beep because it finally understood, and stops moving.

2 Related Work

Several groups have built voice controlled wheelchairs. The field's focus to date seems to have been on getting the control algorithms right rather than on the high level interface. I could not find any existing chairs that attempt to build a

NavChair	RoboChair	Wheelesley	Asfaloth
Stop	Forward	Forward	Go straight
Go Forward	Left (turn 30^o)	Left	Turn left
Go Backward	Right(turn 30^o)	Right	Turn right
Rotate Right/Left	Turn left (turn continuously)	Back	Take a right
Hard Right/Left	Turn right (turn continuously)	Stop	Take a left
Soft Right/Left	Stop		Turn around
	Pass door		Stop
	Approach desk		
	Follow wall		

Table 1: Command sets for several voice controlled wheelchairs.

relationship with the user. Existing systems have a set of navigation modes for high level control. Based on user input from a discrete (eg, voice) or continuous (eg, joystick) source, the system selects an operating mode and type of motion for the chair. The command set is finite and relatively context independent. (See table 1.)

Holly Yanco [Yanco, 1998] built a robotic wheelchair named Wheelesley that understands high level commands. When it is moving forward it avoids obstacles and follows hallways. She did not build a speech interface, although this would be a straightforward extension; her system was designed to support many different control interfaces.

The NavChair system is a wheelchair designed to reduce the cognitive and physical load required of the user. ([Simpson and Levine, 1997], [Levine et al., 1999]) It has three modes: general obstacle avoidance, door passage, and automatic wall following. The user controls the chair with a fixed set of context independent commands. The chair automatically figures out what mode to be in using a Bayes net. The translation of its commands to the chair's motion does not depend on context, except for obstacle avoidance.

RoboChair [Pires and Nunes, 2002] uses fuzzy logic to blend user voice and joystick commands with obstacle avoidance goals. It has three modes: intelligent obstacle avoidance, collision detection, and contour following. Similar to the NavChair system, the chair uses a fixed set of low level commands to enable the user to control the chair's moment in detail. In addition, it has a few higher level commands such as "Pass door", "Approach desk" and "Follow wall". However this is still below the level of commands that a human might typically give another human.

3 Defining the Problem

In order to better understand the problem, I performed a preliminary user study, involving six subjects. I instrumented the Player/Gazebo architecture to record and replay sessions with the mobile robot. One subject drive and one gave verbal instructions. For most of the runs, I drove the robot, obeying people's instructions. After transcribing some of this data, I found that most of the utterances mapped to "left", "right", and "straight", where the exact goal depended on the situation. There were also other commands, such as "look left" to turn left without moving, and "Follow the wall", to have the driver search the maze. Based on this preliminary study, I chose to implement "left", "right", and "straight", where the robot uses the environment to plan a context-sensitive trajectory based on available pathways. I think it is important to create a usable system first, before attempting to solve more interesting problems like the command to "Sneak across the room."

4 Architecture

The system is unified by a high level "brain" module that receives the output from the speech recognition system, decides what routines to apply, and sends appropriate commands to the robot based on the application of routines to sensor readings.

4.1 Robot Simulation

The Player/Stage/Gazebo project is an open source robot simulator and control architecture. [pla,] The Gazebo simulator sends sensor output to the Player robot control server, and receives commands from it. The brain module connects to the Player server for all interaction with the robot. The Player server can talk to a wide variety of robot platforms, so moving from the simulator to the real world, or moving among various real-world architectures should be possible simply by changing the Player configuration file. I do not expect the system to work unchanged after such switches, but it will not require a total rewrite to retarget.

4.2 Language Understanding

I use the Sphinx speech recognizer [sph,] and Peter Gorniak's speech understanding system [Gorniak and Roy, 2005] to convert the speech signal into user commands. I created a grammar for his parser using data from the preliminary study described above. First I used Charniak's parser to parse selected transcribed utterances from the study, and then used Gorniak's utility to generate an initial grammar and lexicon. I also used the CMU Statistical Language Modeling Toolkit[cmu,] to generate a language model for Sphinx. I then modified the lexicon to parse the speech into robot commands. Gorniak's parser searches among possible utterances sent by Sphinx in order to find the most probable parse.

It would have been better to run more subjects in the study, transcribe all the speech, and generate a grammar from this larger data set. This methodology would have enabled the system to understand more utterances and have a more accurate probability model for the grammar. However, I did not want to invest that much time into building out the speech understanding component without prototyping first.

4.3 Semantic Representation

The parser converts the user's utterance into a frame like representation of the motion command. The frame consists of the following fields:

path Specifies the constraints on the path given in the utterance. Consider "go" vs "turn".

goal Specifies the goal specified in the utterance. Consider "right" vs "left" vs "back".

speed Specifies the speed. Consider "fast" vs "slow". (Not yet implemented.)

For example, if the user says "Go right", the following frame is created:

path	Go
goal	Right

The robot control uses this data structure to determine what controller should be active, and thus what routines to run.

I believe many verbs can be broken down into these components. Even many verbs with no overt spatial component can be broken down into a goal to move to, and a domain specific action to perform at the goal. (eg, talk, kick, open, close, put, lift, push, pickup)

4.4 Robot Control

The robot control system consists a set of controllers that each perform different tasks. Each controller corresponds to a different behavior of the robot. The high level control algorithm selects the appropriate controller and its arguments based on input from the language understanding system.

Stop Causes the robot to stop.

Gradient Causes the robot to plan a path towards a goal, using the gradient method of [Konolige, 2000].

FindOpening Causes the robot to find an opening to the right or the left by sending out a ray until it intersects with a wall, and then tracing the wall until it ends.

FollowPath Causes the robot to follow a preplanned path.

FollowCorridor Causes the robot to follow a corridor by turning towards the longest open pathway in front of it. If there are multiple longest pathways, it chooses the one nearest its current trajectory.

Turn Causes the robot to turn in place.

At the lowest level, the robot is moved around using the Vector Field Histogram [Ulrich and Borenstein, 1998] position driver from Player. Each control algorithm sets the position of the robot in the global coordinate system and then the driver attempts to move the robot to that position, avoiding obstacles as necessary. There is no attempt to correct odometry error; it works fairly well because none of the algorithms rely on the long term accuracy of the position system.

4.5 Failure Modes

The most common failures of the system are due to errors in speech recognition and errors in robot control.

speech recognition failures Especially unvoiced fricatives seem to give problems. (eg, sssstop, ffffforward) I added "whoa" to the vocabulary as a backup to "stop" as one palliative for this problem.

robot control failures Sometimes the algorithm to "turn right/left" does not work correctly, especially on non-90^o corners and on narrower openings.

grounding failures If there is no valid place to turn when told to take a right", it behaves in correctly. If there is no wall in the turning direction, it will continue going straight. If there is no doorway, it will "hallucinate" a doorway far ahead.

parse failures The grammar and lexicon could be more complete. eg, the command "Forward!" does not work, although "Go forward." does.

4.6 Relational Behavior

The system tries to relate to the user by giving feedback when it realizes it is having trouble. It uses beeps because I thought that giving language-feedback would artificially raise user expectations from the chair. The chair makes the following kinds of beeps:

Confused It beeps in a confused way when it detects a failure to understand.

Sad It beeps in a sad way after several failures to understand.

Happy It beeps in a happy way the first time it understands after several failures.

In order to avoid annoying the user, the beep state times out after about 20 seconds: it only reacts to local successes and failures, not long-term ones. The system would be better if it detected and reacted to more types of failure, but currently it only reacts to speech recognition failures.

5 Evaluation

To evaluate the system I had four subjects come in and play with it. Two subjects used the relational version, with beeps, and two used the non-relational version without beeps. It was extremely open-ended because I do not think the system works well enough for a formal evaluation to be successful. They were allowed to ask clarifying questions and I explained the robot's behavior when it was confusing. None of them successfully used "Take a right." because of its current limitations: it takes about two seconds to respond, and it does not always work when it does respond. However, I thought that two were able to

Beep	No Beep
5.3	2.50
3.37	6.01
4.25	

Table 2: ITS results for the subjects. The * subject used the non-relational version first, before using the relational version. (His non-relational score was 2.5.)

control the system quite well using corridor following and "turn right", "turn around".

After playing with the system, subjects completed the individualized trust metric to evaluate whether the relational components caused them to trust the system more. Results were inconclusive. The difference in the means was not significant. Some subjects complained that the ITS metric was not relevant and one refused to complete all the fields. It probably would have been better to remove some entries from the metric first. The ITS score was computed by summing the scores from each subject, then dividing by the number they completed.

The system obviously works better for me than for naive users. I have trained myself to talk to the speech recognizer so that it understands me most of the time. (At least one user was able to do this as well.) I think that with some training, a user could control it in its present state. Of course, there is much room for improvement, and my next priority is to improve the control algorithms to reduce this problem.

6 Conclusion

I have built a usable end-to-end voice controlled vehicle. It can be used by by some naive users to navigate through a maze of corridors. It currently understands a limited repertoire of underlying commands. With a better control systems and a more thorough evaluation I believe it would be of interest to people in the wheelchair engineering community.

The system needs to fix some basic usability issues, and needs a more thorough evaluation. It would also be better if the relational component evolved over time: it should learn the user's schedule and habits. There has been some work in detecting affect from voices, and it would be interesting to plug this into the system and create a user model. Once the system can detect and predict the user's emotional state, it can relate to the user differently depending on their mood.

References

[pla,] http://playerstage.sourceforge.net/.

[cmu,] http://www.speech.cs.cmu.edu/slm_info.html.

[sph,] http://www.speech.cs.cmu.edu/sphinx/sphinx.html.

[Fehr et al., 2000] Fehr, L., Langbein, W. E., and Skaar, S. (2000). Adequacy of power wheelchair control interfaces for persons with sever disabilities: A clinical survey. *Journal of Rehabilitation Research and Development*, 37(3).

[Gorniak and Roy, 2005] Gorniak, P. and Roy, D. (2005). Speaking with your sidekick: Understanding situated speech in computer role playing games. In *Proceedings of Artificial Intelligence and Digital Entertainment*.

[Konolige, 2000] Konolige, K. (2000). A gradient method for realtime robot control. In *Proceedings of the 2000 IEEE/RSJ International Conference on Itelligent Robots and Systems*.

[Levine et al., 1999] Levine, S., Bell, D., Jaros, L., Simpson, R., and Koren, K. (1999). The NavChair assistive wheelchair navigation system. *IEEE Transactions on Rehabilitation Engineering*, 7(4).

[Pires and Nunes, 2002] Pires, G. and Nunes, U. (2002). A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller. *Journal of Intelligent and Robotic Systems*, 34(3):301–314.

[Simpson and Levine, 1997] Simpson, R. and Levine, S. (1997). Adaptive shared control of a smart wheelchair operated by voice control. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robots and Systems, 1997*.

[Ulrich and Borenstein, 1998] Ulrich, I. and Borenstein, J. (1998). VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation*.

[Yanco, 1998] Yanco, H. A. (1998). Wheelesley: A robotic wheelchair system: Indoor navigation and user interface. *Assistive Technology and AI*, pages 256–268.