Spatial Routines for a Simulated Speech-Controlled Vehicle

Stefanie Tellex
MIT Media Lab
20 Ames St. Rm. 486
Cambridge, MA 02141
stefie10@media.mit.edu

Deb Roy
MIT Media Lab
20 Ames St. Rm. 483
Cambridge, MA 02141
dkroy@media.mit.edu

ABSTRACT

We have defined a lexicon of words in terms of *spatial routines*, and used that lexicon to build a speech controlled vehicle in a simulator. A spatial routine is a script composed from a set of primitive operations on occupancy grids, analogous to Ullman's visual routines. The vehicle understands the meaning of context-dependent natural language commands such as "Go across the room." When the system receives a command, it combines definitions from the lexicon according to the parse structure of the command, creating a script that selects a goal for the vehicle. Spatial routines may provide the basis for interpreting spatial language in a broad range of physically situated language understanding systems.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing; I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

algorithms, human factors

Keywords

visual routines, spatial routines, wheelchair, spatial language, situated language processing, language grounding

1. INTRODUCTION

A robot capable of understanding and producing spatial language could be controlled by a novice user to perform complex tasks using succinct, intuitive commands. Moreover, there is evidence that spatial reasoning underlies many parts of human cognition [3]; a system that performs spatial reasoning may allow us to bootstrap our understanding of many other facets of intelligence. By trying to build spatially competent machines, we create a useful system in itself and also provide insight into possible mechanisms for modeling human cognition.

HRI '06 Salt Lake City, Utah

Spatial language is situated in space and time: the meaning of many spatial utterances depends on the shared environment of the speaker and the listener. For example, if one person asks another to "sneak into the room," the same verbal request could have very different meanings depending on social and physical context: if there are people or windows in the room, these will affect how the listener decides to act. We seek to build a machine that grasps the situation-dependent meaning of spatial language in human-like ways.

Landau and Jackendoff [8] divide spatial processing into object recognition, and specifying paths and locations. We focus on specifying paths and locations by developing a speech controlled wheelchair that understands high level natural language commands such as "take a left." In this problem domain, linguistic commands map to concrete behavior. This domain makes the problem tractable, while still preserving the context dependence that makes the open domain problem hard; to act correctly, a system must take into account the environment in which a command is issued. For example, if the vehicle is the center of a room, the command "Go right" specifies a different trajectory than the same command issued in a corridor near an intersection. See figure 1 for an illustration.

Inspired by Ullman's [15] visual routines, we created a set of primitive operations over occupancy grids. We defined words in terms of these primitives, creating spatial routines. The system combines these definitions based on the parse structure of the utterance received from the speech recognition engine, and uses the resulting script to control the behavior of the vehicle. Our system, called Asfaloth, can be used to navigate around a simulated world. A sample interaction with the system follows:

> User says "Go forward."
>
> Robot begins moving forward, following the corridor, avoiding obstacles as necessary.
>
> User says "Go right."
>
> Robot moves towards the first opening in the wall on its right.
>
> User says "Go across the room."
>
> Robot computes the convex hull of the obstacles it sees (the walls of the room), then goes through the center of mass to the edge of the region.

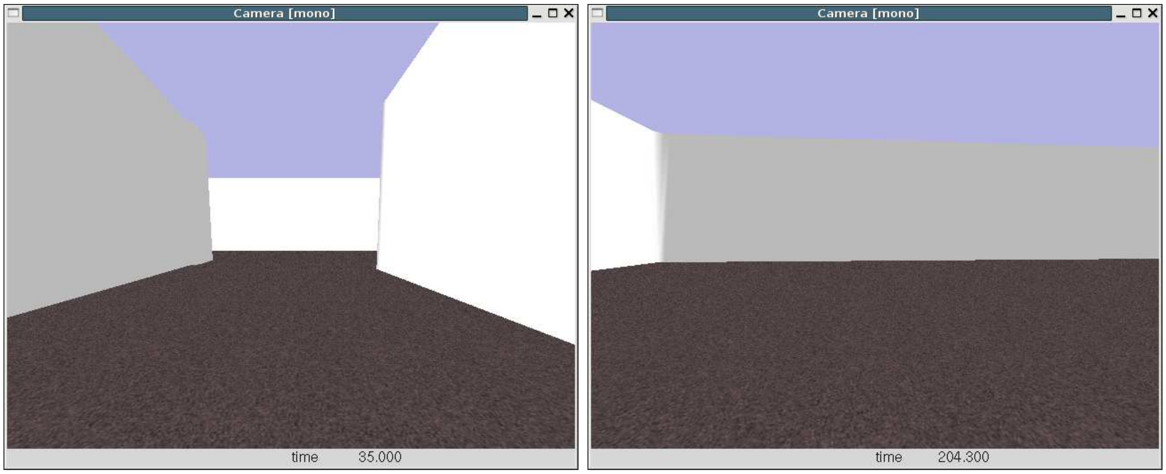

Figure 1: The command "Go right" means different things in different situations. In the situation to the left, it means to go to the intersection and turn right; in the second situation, it means turn roughly 90^o and go forward.

2. RELATED WORK

First, we review the work done by Ullman and others developing visual routines as cognitive models for image processing. Second, there is a large set of work on mobile robots and robotic wheelchairs, of which we review selected works most relevant to our current system.

2.1 Visual Routines

Ullman [15] suggests a set of four primitive operations that process images to perform visual tasks. The primitives operate on an image bitmap and points within the map. Some operations run on all elements of the grid, and others run at specified locations. For example, the *indexing* primitive identifies odd-man out locations in the base representation; humans do something analogous when they find the odd-colored object in a group. The *coloring* operation spreads activation in the base representation, stopping at boundaries. This can be used to identify whether a point is inside a bounded region.

Rao [11] presents the RG system which uses a visual routine-based architecture to process black and white images. This builds on Ullman's ideas and adds a concrete implementation. His base data types are extremely similar to the ones used in our system:

- Location Map – Bitmap selecting a location.
- Property Map – Matrix of numeric values.
- Summary Value – Numeric value for a region (e.g., perimeter, area).

He organizes his operations based on the function type:

- Location Map ⟶ Property Map
  - Region-bounded – For example area, perimeter, connected components.
  - Diameter-bounded – Properties within some radius of a location (for example orientation, size, density).
  - Relative spatial properties – One pixel might point to another one (for example, to the region nearest that pixel).
- Property Map ⟶ Location Map
  - Select locations – Based on property value, using a cutoff or some other function.
  - Coloring – Select locations connected to seed locations.
- Property Map × Location Map ⟶ Summary Values
  - Area of a region.
  - Min, Max.
  - Direction towards a point.

Rao sees routines as a language for the focus of attention. Inherent biases in visual attention guide processing, and create patterns of routine calls. These patterns become a way of identifying and predicting visual events. Once it recognizes the beginning of a pattern, the system can predict which calls might come next, and use that to bias further visual processing. In addition, a system can assign labels to patterns of interesting sequences of primitives. These labels could map to natural language, and provide an interface to understand and talk about the visual field.

Rao refers to the elementary operations as primitives, and groups of primitives chained together as the visual routine; in this paper we follow the same convention, referring to primitives and "spatial routines."

2.2 Mobile Robots

[t!]

There is a large body of work building wheelchairs with alternative interfaces to the conventional joystick. Joystick

RobChair [10]	NavChair [13]	Wheelesley [17]	Current System
Stop	Stop	Stop	Stop
Go forward	**Forward**	**Forward**	**Go straight**
Go backward	Left/right (turn 30^o)	Back	Face left/right (turn 90^o)
Rotate right/left	Turn left/right (turn continuously)	Left/right	**Go left/right**
Hard right/left (turn 20^o)	**Pass door**		Turn around
Soft right/left (turn 10^o)	**Approach desk**		**Go across the room**
	Follow wall		**Go to the object**
			Go to the left/right of the object

Table 1: Command sets for several speech controlled wheelchairs. Boldface indicates the command takes into account the robot's current situation.

SPOTT [18]					
Verb	Destination		Direction		Speed
	Preposition	Target	Orientation	Path	
Go	across	door	forward/backward	along	slowly
	against	hallway	north/south/east/west	around	quickly
	along	room	left/right	via	normal
	alongside	wall		to	
	to the right of	chair		toward	
				from	
				away from	

Table 2: Command set for Zelek's SPOTT architecture.

interfaces enable analog control of wheelchairs. In contrast, a natural language interface is discrete: a human directing another human specifies the location with a short sequence of words, relying on the shared environment and the listener's common sense to disambiguate the instruction. Fehr et al. [4] 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 speech controlled wheelchair that understands high level commands could increase the mobility options for such people because it would require less motor control to operate.

In order to better understand the kinds of commands that make sense in a linguistic interface we performed a preliminary study involving six able-bodied subjects. Subjects were instructed to give verbal instructions to a human driver to navigate through a simulated maze. For the first task, the subject was instructed to search the maze for hidden statues. For the second task, the subject was given a map of the maze and told to drive to a target location. Many of the instructions that subjects used 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 work, we chose to implement "left," "right," and "straight," where the robot uses the environment to plan a context-sensitive trajectory based on available pathways.

Most previous work on robotic wheelchairs has focused on low level control rather than high level (e.g., linguistic) interfaces. Existing systems have a set of navigation modes that determine which control algorithms are active. Based on user input from a discrete (e.g., voice) or continuous (e.g., joystick) source, the system selects an operating mode and type of motion for the chair. When a natural language interface is used, the command set is usually context independent (see table 1). All systems support some form of obstacle avoidance.

The NavChair system ([13], [9]) is a wheelchair designed to reduce the cognitive and physical load required of the user. It has three modes: general obstacle avoidance, door passage, and automatic wall following. The user controls the chair with a set of speech commands. The chair uses a Bayes net to decide what mode to activate based on the chair's current location; if it is near a wall or a door, it is more likely to be in wall following or door passage mode respectively. Its right/left commands, however, do not depend on context except for obstacle avoidance.

RobChair [10] uses fuzzy logic to blend user speech and joystick commands with obstacle avoidance goals. It has three control modes: intelligent obstacle avoidance, collision detection, and contour following. It uses a simple arbitrator to select between modes. When near an obstacle, it goes into collision detection mode and only allows backwards movement. Otherwise, the system selects the control mode based on the user command.

Yanco [17] built a robotic wheelchair named Wheelesley which 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 as her system was designed to support many different control interfaces.

Skubic et al. [14] developed Coyote, a mobile robot that can understand and generate spatial language about its environment. It can sense objects, assign number-labels to them, and then a human can assign names to the numbered objects. It can obey commands like “Go behind the pillar” (once a human has identified an object as a pillar) and describe where objects are relative to itself (e.g., “The object # 1 is behind me but extends to the left relative to me. The object is very close.”) It keeps track of object labels even after moving. Of the systems we review, Coyote’s interface is closest in spirit to our own, and its language understanding ability is greater than our own.

Gribbal et al. [6] describe a plan for a robotic wheelchair with an interface dervied from Kuiper’s Spatial Semantic Hierarchy. At the highest level, the user can select a location on a map and the robot will navigate to it. At the middle level, users select commands analgous to the ones we describe: “turn right/left”,”go straight” and “stop.” Finally at the lowest control level, the user can select the control law the chair is using, causing it to for example rotate clockwise or follow a corridor.x

Zelek [18] identified a subset of spatial natural language that he uses to control a mobile robot. Commands are formalized as specifying a verb, destination, direction, and speed. Spatial prepositions form part of the destination and trajectory specifications for a command. He claims the lexicon is a minimal spanning semantic set for two dimensional navigational tasks. The system understands a large vocabulary - too large to list compactly in the table.

3. ARCHITECTURE

Our system is unified by a high level module that receives the output from the speech recognition system and simulated sensor data, creates a script using the lexicon and the parse structure of the command, and then sends appropriate commands to the simulated robot.

3.1 Robot Simulation

We implemented the system in simulation in order to quickly test possible design choices while initially sidestepping issues with sensor reliability and hardware failure. Our ultimate goal is to build an actual hardware system. We used the Gazebo simulator with the Player robot control server. The Player/Stage/Gazebo [1] project is an open source robot simulator and control architecture. The Gazebo simulator sends sensor output to the Player robot control server, and receives commands from it. The Player server can talk to a wide variety of robot platforms, as well as the simulator. We are using Gazebo to simulate a Pioneer 2 AT robot equipped with a SICK LMS LIDAR unit, and the Pioneer’s 16 onboard sonar sensors. For debugging purposes we also used a camera mounted on the virtual robot, although we only used range data in the system because of its simplicity and increased robustness in the real world. Eventually we plan to migrate this system to the physical robot. To run on a given platform, the system requires at least 180 degrees of range sensor data, as well as an odometry/motor control interface.

3.2 Language Understanding

We use the Sphinx speech recognizer [2] and Gorniak and Roy’s speech understanding system [5] to convert the speech signal into user commands. We handcrafted a grammar and a lexicon for the system based on the types of utterances we wished to cover. Gorniak and Roy’s parser searches among possible utterances generated by Sphinx to find the most probable parse. For testing purposes, we also implemented a textual interface where commands are typed to the robot. This appears as noise-free speech recognition output to the system.

3.3 Semantic Representation

The parser creates a series of nested procedure calls based on the grammatical structure of the utterance. Each procedure call generated by the parser maps to a word in the lexicon. The content of the procedure is the definition of that word in terms of its associated spatial routine.

For example, if the user says “Go right,” the parser creates the following representation: `go(right())`. In the dictionary, the “right” procedure is executed, and its results are passed to the “go” procedure. The result of this is a script that when executed yields a goal and a path to the goal. The robot then follows the path to reach the goal.

3.4 Routines

The fundamental data structure that routines operate on is an occupancy grid. Following Rao [11], the data structures are as follows:

Numeric Grid Real numbers range $[0, 1]$.

Freeman Grid Integers in the range $[0, 7]$ specifying one of the eight neighboring points. This is used to specify paths.

GridMask Boolean grid; used to select regions in the grid.

Region Contiguous region; represented as a grid mask where all unmasked points are connected.

Point A point in the grid.

Direction A unit vector.

The routines below are defined as functions on the above data types.

Gradient Takes a goal point and finds paths to that goal point using Dijkstra’s algoritm. This algorithm is described by Konolige. [7].

TraceLine Calls a function on each point in a ray specified by a start point and a direction point. Writes the output of the function to an output grid.

ColorRegion Calls a function on each point in a region. It takes as a parameter a function that returns whether a point is in a region or not, so it can be used to create region data structures. Writes the output of the function to an output grid.

UnmaskCircle Returns a region unmasked around a specified point and a specified radius.

Salience Returns a grid marking each point with a baseline salience based on whether it is in front or behind the robot and how far it is from the robot. Close points near the front are considered more salient.

ConvexHull Returns the convex hull of a mask.

Area Returns a grid where each point is labeled with the area of the region it is part of.

Max/Min Returns the point where a grid takes on its maximum/minimum value.

CenterOfMass Returns the center of mass of a region.

BlockedScore Scores a point based on how many directions obstacles are visible for a given radius.

Distance Computes the Euclidean distance between two points.

Divide Divides one occupancy grid by another, element wise.

The lexicon in the current implementation is grounded in spatial routines as follows:

go Takes a numeric grid, a region, or a point. Converts the first two to a point by taking the max or center of mass respectively. Finds a path to that goal using Gradient. Words like "sneak" and "hide" could be defined similarly, using a different cost function for the gradient algorithm.

face Takes a numeric grid. Normalizes the points by dividing by the distance from the goal point, then takes the minimum adjacent point as the goal. This is used for "face right."

left/right Returns a numeric grid. Reachable points far to the left/right are scored high. When passed a point as an optional argument, it returns a grid assigning an attention vector sum [12] score to points around its argument, as in "the left of the object."

stop Returns a zero length path ending at the current position.

around Returns a numeric grid where points immediately behind the robot are scored high. This is used for "turn around."

room Returns a region that is the convex hull of the obstacles currently visible to the robot's range sensors. This relies on the finite range and resolution of the range sensors.

object Returns the smallest most salient contiguous region visible to the robot. This algorithm is not very robust. Skubic et al. [14] describe an algorithm for smoothing occupancy grids that could make it more reliable.

across Takes a region. Finds its center of mass. Traces a line from the robot's current location through the center of mass to the edge of the region. Returns the end of the line segment.

We developed these lexical definitions manually, based on our intuitive sense of what the robot should do when given the corresponding command. In the future, we plan to investigate automatic learning algorithms that acquire such definitions from examples of robot movements paired with linguistic descriptions.

	Correct	Inc.	NA	% Correct (exc. NA)
Go left	16	4	0	80% (80%)
Go right	14	2	4	70% (87.5%)
Go across the room	7	5	8	35% (58.3%)

Table 3: We report the number of correct trials, incorrect trials, and NA trials where in our judgement there was no acceptable behavior.

3.5 Robot Control

The system analyzes the path and goal received from the language processing module in order to chose one of the following three controllers:

FollowSpline Causes the robot to follow a path generated by executing a script, and stop at the end.

MoveToAngle Rotates the robot in place. Executed when the goal point is very close; causes the robot to turn in place to face the goal.

Stop Stops the robot. Executed when the goal point is the robot's current location.

For following corridors, it uses a control algorithm that guides the robot towards the longest open pathway in front of it. If there are multiple equally long pathways, it chooses the one nearest its current trajectory.

At the lowest level, the robot's movements are controlled using the Vector Field Histogram [16] 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. The controller works fairly well because none of the algorithms rely on the long term accuracy of robot's global position.

4. INITIAL RESULTS

To evaluate the system we created a maze (shown in figures 4-7), and tested the robot's ability to interpret verbal commands in various locations. For each command we picked twenty locations from a uniform distribution inside the maze, excluding those with walls within approximately two of the simulated robot's radii. The robot's orientation was picked from a uniform distribution. At each location, the command was issued textually, and we judged whether the goal point chosen was semantically appropriate. In order to focus on the performance of the spatial routines component of the system, we only judged the planned path, rather than the path the robot actually followed. These can be different especially when the path is close to obstacles. When the goal point was incorrect, we judged whether a human would have understood the command at that location. We report the percentage correct over all trials, and the percentage correct excluding the trials where we thought a human would not have understood the command.

The purpose of this evaluation is to show that it is possible to use these routines to build a system that is functional rather than to show that we have identified the best possible algorithms, or to characterize a human's understanding

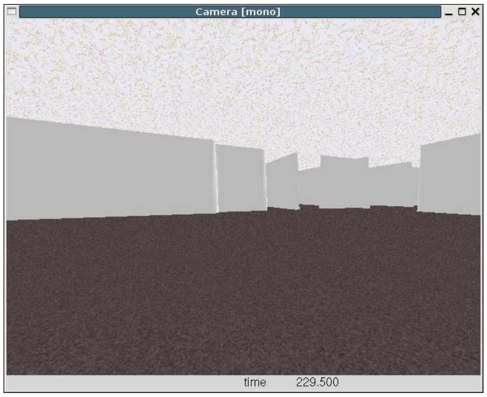

Figure 2: The robot's camera when receiving the command "Go across the room."

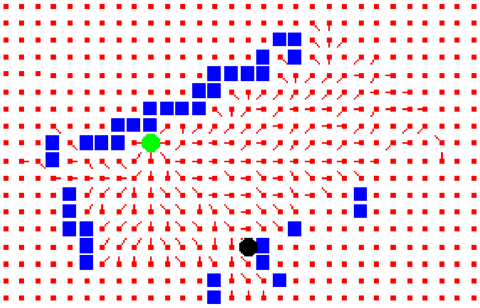

Figure 3: The occupancy grid and planned path for the command "Go across the room." Green marks the goal, black is the current location of the robot, and blue is detected obstacles.

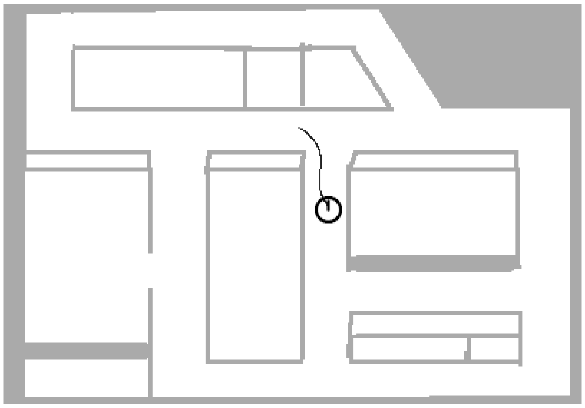

Figure 4: The robot's behavior when told "Go left." near a "T" intersection.

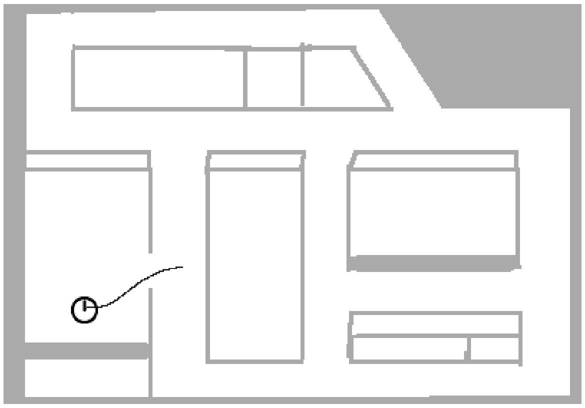

Figure 5: The robot's behavior when told "Go right." in a room near a doorway.

of these commands. We are unaware of prior implementations of these commands, so we do not compare our system's performance to either a baseline or other implementations. We also present several images showing the robot's behavior when given a command at particular locations. The robot is drawn in its starting orientation, and the line shows the path that it actually took after receiving the command.

The commands "go left" and "go right" worked correctly on 75% of all trials. This performance level is encouraging since these two are likely to be among the most frequently used commands. The system tends to fail in one of two ways. First, if the robot faces a "T" intersection at an angle, as shown in figure 7, and is told to turn away from the intersection, sometimes it selects a goal point going down the corridor away from the intersection. This situation is somewhat ambiguous, but we think it is more intuitive if the robot follows the corridor and moves forward. Second, the camera in the simulator can see farther than the LIDAR range. As a result, sometimes when a human sees a doorway and says "go left," the robot cannot sense the door that the human is referring to, and picks an inferior goal point.

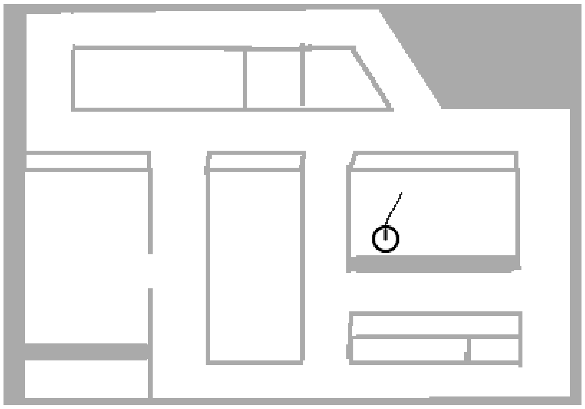

Figure 6: The robot's behavior when told "Go across the room." at a particular location.

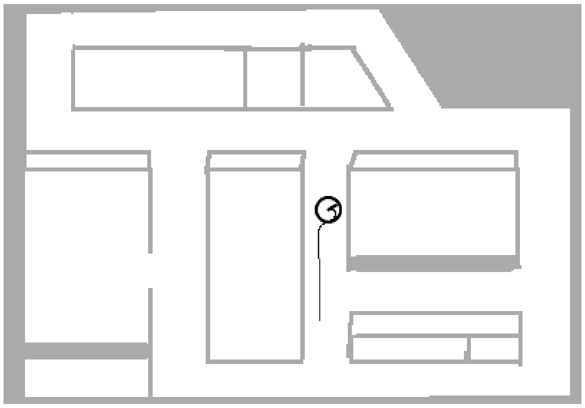

Figure 7: The robot's behavior when told "Go right." Here the side sensors see down the corridor so it turns to the right rather than going to the intersection and turning right.

The command "go across the room" worked less well, only on 35% of trials. Figure 6 shows an example of the algorithm working. Because of the maze we used, "across" was more likely to be tested in corridors, where there is no good behavior. When those trials are excluded, the performance improves markedly. Remaining failures are probably due to trials where the sensors detected a corridor leading off from the room, which skewed the convex hull computation. Some are also from being in rooms where the robot could not see the other end; in these cases the far walls were excluded from the convex hull computation.

We only evaluate three commands here although our system understands several others. First, we excluded commands that were not context dependent such as "stop" or "turn around." Second, we excluded commands involving "object" because the system does not reliably identify the object being referred to using only range sensor data. In situations where the object is unambiguous, it reliably picks appropriate goal points for commands like "Go to the object" and "Go to the right of the object." Skubic et al [14] describe several solutions to this problem: they filter and blur the range sensor data. In addition, their robot can verbally report the objects it sees and accept labels for them. An ability like this combined with geometric object selection (such as "the object on the right") would improve performance dramatically.

When used in text mode, the most common failures of the system stem from its inability to detect situations where commands do not make sense. Ideally in this case the system should indicate its confusion. When used in speech recognition mode, the most common errors stem from speech recognition failures. The robot's control systems also occasionally fail, especially when the goal chosen is behind the robot, or when it is close to a wall.

5. CONTRIBUTIONS

We have identified a set of routines that can be used to define words for an implementation of a speech controlled vehicle. This architecture could be applicable a range of applications in human-robot interaction, such as controlling a robotic vacuum cleaner, in which context-dependent interpretation of spatial language is needed. Moreover it might be possible to use the routines to identify types of movement, yielding a system that could describe its own actions, and even watch a camera and describe the spatial content and trajectories of objects. Such functionality could be used for video retrieval via natural language.

The system in its current state is a proof of concept for an initial set of spatial routines, and there are many possible extensions and improvements. Most pressingly, the current implementation can only act based on the current snapshot of sensor readings. This stateless design leads to errors in behavior. Second, it should handle uncertainty better. When given a command that does not make sense, the system currently takes its best guess. Instead, when there is no good goal point, it should signal failure to the user. Finally, the simulator makes many things easy. Implementing the system on a robotic wheelchair or another mobile robot will of course expose additional problems that we may have overlooked.

Looking farther into the future, implementing some form of Rao's attentional biases and learning algorithms would enable the system to possibly discover routines and spatial relations on its own.

6. ACKNOWLEDGMENTS

We would like to thank Piotr Mitros, Gregory Marton, and Lin Wu at MIT for their comments on previous drafts of this paper. The images showing the robot's trajectory were formatted by Piotr Mitros.

7. REFERENCES

[1] http://playerstage.sourceforge.net/.

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

[3] L. Boroditsky. Metaphoric structuring: Understanding time through spatial metaphors. *Cognition*, 75:1–28, 2000.

[4] L. Fehr, W. E. Langbein, and S. Skaar. Adequacy of power wheelchair control interfaces for persons with

severe disabilities: A clinical survey. *Journal of Rehabilitation Research and Development*, 37(3), 2000.

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

[6] W. S. Gribble, R. L. Browning, M. Hewett, E. Remolina, and B. J. Kuipers. *Integrating Vision and Spatial Reasoning for Assistive Navigation*, volume 1458, page 179. 1998.

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

[8] B. Landau and R. Jackendoff. "What" and "where" in spatial language and spatial cognition. *Behavioral and Brain Sciences*, 16:217–265.

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

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

[11] S. Rao. *Visual Routines and Attention*. PhD thesis, Massachusetts Institute of Technology, February 1998.

[12] T. Regier and L. Carlson. Grounding spatial language in perception: An empirical and computational investigation. *Journal of Experimental Psychology: General*, 130(2):273–279, 2001.

[13] R. Simpson and S. Levine. 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*, 1997.

[14] M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, M. Bugajska, and D. Brock. Spatial language for human-robot dialogs. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 34(2), May 2004.

[15] S. Ullman. Visual routines. Technical Report AIM-723, Massachusetts Institute of Technology, 1983.

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

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

[18] J. S. Zelek. Human-robot interaction with a minimal spanning natural language template for autonomous and tele-operated control. In *International Conference on Intelligent Robots and Systems (IROS '97)*, 1997.