# Miscommunication Detection and Recovery for Spoken Dialogue Systems in Physically Situated Contexts

Matthew Marge

CMU-LTI-15-015

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

**Thesis Committee:**
Alexander I. Rudnicky (Chair), Carnegie Mellon University
Alan W Black, Carnegie Mellon University
Justine Cassell, Carnegie Mellon University
Matthias Scheutz, Tufts University

*Submitted in partial fulfillment of the requirements*
*for the degree of Doctor of Philosophy*
*in Language and Information Technologies*

**Keywords:** physically situated dialogue, spoken dialogue systems, human-robot communication, human-robot interaction, recovery strategies, instance-based learning

*To my wife, Boyoung*

# Abstract

Intelligent agents that are situated in a physical space, like robots and some virtual assistants, enable people to accomplish tasks. Their potential utility in many applications has also brought about a desire to communicate with them effectively using spoken language. Similarly, advances in speech recognition and synthesis have enabled spoken dialogue systems to run on these platforms. These agents face an array of problems when they try to apply instructions from people to their surroundings, even when there are little to no speech recognition errors. These *situated grounding problems* can be referential ambiguities or impossible-to-execute instructions. Navigation tasks are a prime example of where such problems can occur. Existing approaches to dialogue system error detection and recovery traditionally use speech and language input as evidence. In this thesis, we incorporate additional information from the agent's planner and surroundings to detect and recover from situated grounding problems.

The goal of this thesis was to produce a framework for handling situated grounding problems in task-oriented spoken dialogue. First, we created a representation for grounding that incorporates an agent's surroundings to judge when it is grounded (i.e., on the same page) with a user. Second, we developed a method for detecting situated grounding problems that combines evidence from the user's language with the agent's plan and map information. Third, we implemented recovery strategies that allow an agent to repair situated grounding problems. Fourth, we used an instance-based learning approach to improve how an agent selects recovery strategies over time. When an agent encounters a new grounding problem, it looks back on its interaction history to consider how it resolved similar situations. Detecting and recovering from situated grounding problems allow spoken dialogue systems for agents to be more resilient to miscommunication and operate in a wide range of scenarios with people.

# Acknowledgments

When I began my time at Carnegie Mellon, I arrived with a passion for artificial intelligence and wanted to discover ways to improve how we can interact with and benefit from intelligent agents. I would like to thank my advisor, Alex Rudnicky, for helping me hone that passion into this body of work. He taught me to think about the big picture, while at the same time pondering deeply about the inner workings of system processes. I am grateful for the academic freedom he gave me to pursue this line of research.

My thesis committee members, Alan Black, Justine Cassell, and Matthias Scheutz, have been exceptionally supportive from the moment I began exploring this thesis topic. I am thankful for my many conversations with Alan as we discussed this thesis and other academic pursuits. I also appreciate valuable advice from Justine and Matthias that helped me find the balance between ambition and empirically *grounded* work.

My work has been supported by the National Science Foundation Graduate Research Fellowship.

I have benefited from many fruitful interactions at Carnegie Mellon. I am thankful to Satanjeev Banerjee, Thomas Harris, and Carolyn Rosé for their advice and many discussions during the early years. Thanks to Maxim Makatchev, Luís Marujo, Prasanna Kumar Muthukumar, and Vittorio Perera for their companionship during the highs and lows of the doctoral program. I would like to thank Aasish Pappu, Ming Sun, and Long Qin for being wonderful officemates and colleagues whom with I could brainstorm my research ideas.

This work also benefited from regular meetings with the Dialogs on Dialogs reading group; in particular I would like to thank David Cohen, Heriberto Cuayáhuitl, Rohit Kumar, José David Lopes, Yoichi Matsuyama, Elijah Mayfield, Gabriel Parent, Alexandros Papangelis, Antoine Raux, and Zhou Yu for their insightful conversations. I would also like to thank the regular participants of the weekly Sphinx Lunch for their feedback on my research. I'm also thankful to many other friends I gained throughout my years at Carnegie Mellon: Nia Bradley, Jonathan Clark, Sourish Chaudhuri, Brian Coltin, Kevin Dela Rosa, Michael Denkowski, Brian Langner, Meghana Kshirsagar, Subhodeep Moitra, Kenton Murray, Gabriel Parent, Danny Rashid, David Reitter, Hugo Rodrigues, Nathan Schneider, and Rushin Shah. Thank you for keeping me upbeat and positive.

My colleagues at the Army Research Lab have been a valuable resource for feedback and ideas. Thanks to Clare Voss for her support and advice on crossing

*the finish line*. I'm thankful to Jamal Laoudi for his assistance with my final thesis experiments. Thanks are also in order for David Baran, Claire Bonial, Taylor Cassidy, Philip David, Arthur William Evans, Susan Hill, Reginald Hobbs, Judith Klavans, Stephen LaRocca, Douglas Summers-Stay, Stephen Tratz, Michelle Vanni, Garrett Warnell, and Stuart Young for their feedback on my dissertation. I'm grateful to Melissa Holland for giving me the opportunity to join such exceptional colleagues.

During my time in Pittsburgh, I greatly benefited from weekly meetings with the First Trinity Lutheran Student Fellowship. Thanks to Pastor Eric Andrae for his prayers and support throughout the entirety of my doctoral program.

I'm thankful to my family for their constant love and support. I could not have achieved this without my parents Susan and Kenneth; they encouraged me to pursue my childhood dreams from the beginning.

But I am most thankful to the person that helped me reach the end – my wife, Boyoung. To her I am most grateful for helping me to hit this milestone. Thank you for your love, enthusiasm, and unwavering patience as I pushed through deadlines and a mountain of work so that this dissertation could be complete. I could not survive without you holding the fort and preparing daily lunches and dinners for me. Thank you for helping me keep my focus, and for setting us up for a wonderful future.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

Chapter 1

# Introduction

People accomplish tasks by communicating, where they often use dialogue to achieve a mutual understanding of information [27]. Today, improvements in technologies like automatic speech recognition and speech synthesis have allowed spoken dialogue systems to run on devices as ubiquitous as smartphones and tablet computers (e.g., Siri on iOS). Spoken dialogue system researchers have traditionally focused on tasks tied to a knowledge base or computer application. These systems are equipped to handle structurally defined domains like travel booking where information is stored in static databases. More recently, there has been growing adoption of dialogue interfaces for educational tutoring systems and virtual assistants (e.g., [42]). All of the above systems operate in controlled conditions, where the system has full access to task-specific information (e.g., bus schedules in the Let's Go! bus information system [109], training tasks and goals in the Mission Rehearsal Exercise system [129]).

These applications face a common problem: robustness issues related to their spoken language understanding components failing to understand part or all of the user's utterance (i.e., *language miscommunication*). Language and speech interpretation failures are the leading cause of communication problems in traditional spoken dialogue systems. To this end there has been a substantial body of work that has enabled dialogue systems to detect and recover from language miscommunication (e.g., [10, 13, 16, 74, 75, 113, 124, 130, 141]).

In recent years, advances in robotics, computer vision, and navigation have brought robots out of labs and into real-world spaces. The emergence of mobile, physically situated agents like robots has also brought about a desire to communicate with them effectively. *Physically situated dialogue* is dialogue that refers

to the surrounding environment: a streaming source of information that is often crucial for processing plans and accomplishing tasks [11]. Dialogue systems can allow people to communicate with situated agents at a more abstract task level than traditional command-and-control devices. However, situated dialogue cannot be treated like traditional human-computer dialogue; often interactions will require reference to the agent's surroundings.

In this thesis, *situated interaction* is communication between two or more communicative partners that could use the following sources of context to accomplish tasks: (1) occupancy, (2) proximity, and (3) orientation. The agent in this thesis has access to these sources of context. *Occupancy* is awareness about the positions of objects, walls, space, and communicative partners. It requires reasoning about space and where space is occupied. *Proximity* describes how close the agent is to each object in the environment and each communicative partner. Proximity requires reasoning over position (the $(x, y)$ coordinates of partners and objects in the environment) and distance (how close or far an object is from the agent or other objects). *Orientation* provides information about the objects each partner is facing and the objects each object is facing. It requires reasoning about spatial terms like left and right. Other sources of context are outside the scope of this work.

## 1.1 Problem Space

The nature of situated interaction yields a relatively unexplored set of problems separate of traditional human-computer dialogue. Situated agents must monitor their environment. Tasks may fail due to dependencies on specific environment configurations, like when a robot's path to a goal is blocked. These environments are often changing, and people may make unrealistic assumptions about a robot's ability to monitor surroundings.

Situated agents also perceive and represent their surroundings differently than people; they sense the world using input modalities like cameras, microphones, and laser rangefinders [67]. This leads to people and agents having fundamentally different interpretations of the environment. While a person may have high confidence about what they see in their surroundings, agents always deal with uncertainties from sensory perception. Without help from people, an agent would need to act on poor representations that could lead to further system failures. Situated agents also need access to spatial skills [148] and may need to manage multi-party dialogue, which brings with it turn-taking challenges [11]. People have arguably

higher expectations for situated agents compared to traditional computer systems because they are embodied and co-present [63]. Co-presence can lead to people assuming that that situated agents share their perspective or awareness of deictic expressions about the environment, which may not be the case due to visual and sensor limitations.

*Situated grounding*[1] *problems* happen when an agent fails to uniquely map a person's instruction to its surroundings, even though the input may have been correctly recognized. Consider the case where a human operator tells a robot to search a nearby room (e.g., "Search the room on your left"). If the operator does not provide enough detail about the room (e.g., there are several rooms on the robot's left), there is a *referential ambiguity* problem; the agent could produce multiple navigation plans for the instruction. If the agent detects no rooms on its left, or all of the rooms are inaccessible, there is an *impossible-to-execute* problem; the agent cannot produce a single plan that uniquely maps the user's instruction to its surroundings. Instructions that resolve to a single plan are *executable*.

Human-human corpora demonstrate that people detect situated grounding problems and use strategies to recover from them. Below is an example of referential ambiguity from the SCARE Corpus, a corpus of human-human dialogues where one person (a direction giver, DG) is providing navigation instructions to another person (a direction follower, DF) situated directly in a virtual environment [128]. Both dialogue partners have access to the direction follower's view.

> *DG:* Face the left door to the left of you
> ***DF:* That one right there?**
> *DG:* No the middle one
> *DF:* Oh ok
> (SCARE Corpus, Dialogue S3)

In this example, the direction giver tells the follower to rotate and face a door in the virtual environment. The giver's instruction was vague enough to prompt the follower to confirm his initial resolution of the door referent, because there were multiple possible doors to go through. The giver notices the ambiguity and corrects his specification to a unique referent in the follower's view, and the navigation task

[1]Grounding here refers to *grounding in communication* [28], the process by which two dialogue partners accumulate information as part of common ground. It does not refer to *symbol grounding*, which has become a popular term in the robotics community for aligning words to sensory perception [47].

can continue. If the direction follower either did not confirm his initial selection, or did not detect the ambiguity in the first place, he would have failed to interpret the giver's intention. This would have led to either subsequent correction attempts by the giver until the follower performed the correct task, a misunderstanding detected a few turns later in the dialogue, or task abandonment.

People ask their dialogue partners questions to prevent communication breakdowns, and in navigation domains situated grounding problems are often resolved in this way. Two studies of human-human dialogue corpora report that questions about a previous dialogue partner's utterance represent 3-6% of all human-human dialogue turns [106, 111]. Questions are even more frequent in dialogue tasks with a direction giver-direction follower pair. A study conducted for this thesis revealed that in the SCARE corpus, roughly 9% of dialogue turns by the direction follower were questions directed to the giver (340 out of a total 3818 turns). Of these questions, 86% were related to situated grounding problems (reference resolution, action confirmation, or spatial relation resolution). In a different study, Tenbrink et al. [133] found that 60% of direction follower turns (in a human-human dialogue about navigating a 2-dimensional map) were questions about directions. A majority of these questions (57%) were about situated grounding problems (namely reference resolution).

The statistics above show that situated grounding problems occur in human-human dialogue, and that people have strategies for dealing with them. We can observe how people recover from these problems, and how they would like agents to recover from them. This would enable us to develop a distribution of recovery strategies. With the ability to detect and recover from situated grounding problems, spoken dialogue systems would be more resilient to miscommunication and operate in a wider range of scenarios with people.

We evaluate automatic detection and recovery from situated grounding problems in the virtual agent navigation domain, where agents are represented as robots. Interactions in the virtual agent navigation domain can be physically situated. This is true in scenarios where a person gives navigational directions, commonly termed *route instructions*, to a virtual agent. Mobile agents must master navigation if they are to work with people – successful dialogues often require a virtual agent to move around an environment based on a user's instructions. Route instructions have been thoroughly studied, both in human-human [2, 117, 128] and human-robot interactions (e.g., [18, 64, 81, 102, 120]).

The nature of route instructions also make them a good candidate for auto-

matic interpretation. Route instructions are sequential, meaning that the order in which instructions are presented is how they should most often be executed [71]. Evaluation is straightforward with route instructions; the agent is successful if it ended at the correct goal in the instruction (this can extend into subgoals in the instruction as well). The user's intention when giving a route instruction is largely matched if the agent arrives at a specified destination (correct subgoals may or may not be important). Route instructions also follow a predictable structure and can be classified into distinct categories like imperative commands and side-comments [102]. In essence route instructions form a "sub-language" of their original language, which prevents out of domain errors. Several groups have developed route instruction interpreters, some for written language [50, 61, 78, 121], some for typed language [119, 131], and some for spoken language [36, 112]. Vogel and Jurafsky developed a route instruction follower using examples from the HCRC Map Task Corpus [138]. There is a great deal of work addressing the route instruction interpretation problem.

The gap that this body of work does not cover is resolving interpretation failures. Without interacting with the user, the best such systems can do is estimate intention. Agents can overcome such problems by engaging in recovery strategies that present the problem or ask the user questions about situated grounding problems. Current dialogue systems do not use information about an agent's surroundings to detect communication problems. If the user's language was interpreted robustly, but an action is ambiguous or impossible-to-execute, current systems are not equipped to handle these types of problems. They are unable to communicate about the grounding problem with the dialogue partner, which can lead to confusion and task failure. This is not acceptable for mobile agents, where tasks like navigation have high costs for failure.

## 1.2 Thesis Statement

**A contextually aware, adaptive approach to recovering from situated grounding problems is novel, feasible, and useful.** Using this approach, an agent detects if an interpreted user instruction is ambiguous, impossible-to-execute, or executable. If there is a problem, the agent informs the user of the problem and, if appropriate, formulates a question to ask the user. This is accomplished by selecting a recovery strategy. The agent attempts to resolve the grounding problem with the user's response.

The approach is *contextually aware* because the agent combines evidence from its task planner, the environment, and the user's utterance to detect possible grounding problems. Identifying grounding problems requires access to the agent's surroundings so that it can try to map user instructions to the environment. Natural language research for route instructions has not demonstrated resilient ways of dealing with situated grounding problems; most have focused on learning to get the initial interpretation of the user's utterance as correct as possible. However, some grounding problems can only be addressed by asking for clarification, and the agent must be aware of its surroundings in these cases (e.g., repairing an ambiguous instruction or notifying the user that a requested move is impossible-to-execute).

The approach is *adaptive* because the agent learns to improve how to recover from situated grounding problems over time. Situated dialogue agents should handle a variety of grounding problems: misunderstood words and phrases, ambiguous instructions, and environment scenarios where a requested move is impossible-to-execute. We address the latter two. We detect situated grounding problems using a combination of static and dynamic information about the current context. The approach selects recovery strategies that previously yielded success. Some of these strategies involve just presenting the agent's understanding of the situation, while some ask the user questions to get the conversation back on track. The agent selects better recovery strategies over time using an instance-based learning approach. When there is a problem, the agent looks back on its history for similar interactions, and selects a strategy that resolved that type of problem in the past.

## 1.3 Thesis Contributions

The primary contributions of this thesis are the following:

- A representation for managing grounding in task-oriented spoken dialogue systems designed for situated agents, called *plan grounding* (Chapter 4);
- Experimentation with crowdsourcing to collect recovery strategies for situated grounding problems (Chapter 5), definition of recovery strategies for spoken dialogue systems in physically situated contexts (Chapter 7);
- A software infrastructure for detecting situated grounding problems that analyzes features from spoken language input, the agent's plan information, and the agent's surroundings (Chapter 6);
- A learning method that changes how a dialogue agent selects recovery strate-

gies for situated grounding problems over time; and a task-based evaluation of this method (Chapter 8). The agent uses its interaction history to select recovery strategies that were successful for similar interactions.

Chapter 2

# Miscommunication in Spoken Dialogue

In this chapter, we introduce miscommunication as it occurs in spoken dialogue and illustrate methods to detect and resolve miscommunication. We review relevant work from the spoken dialogue, human-computer interaction (HCI), human-robot interaction (HRI), and artificial intelligence (AI) communities. We first review existing sources of miscommunication. Then, we discuss past approaches to spoken dialogue error detection, which this thesis expands to handle situated grounding problems. Next we discuss past approaches to error recovery in spoken dialogue, which have mainly focused on repairing language misunderstandings and non-understandings. In this thesis, we expand this to situated problems that not only require access to the speech signal, but also the environment in which the agent operates.

## 2.1 Sources of Miscommunication in Dialogue

Miscommunication can be any event in a dialogue exchange where something problematic happens. Traditionally, miscommunication encompasses misunderstanding, non-understanding, and ambiguity in dialogue. Hirst and colleagues [51] defined differences between misunderstanding and non-understanding in human-human dialogue. *Misunderstanding* refers to when a dialogue partner interprets an utterance in a way that is not in line with what the speaker intended. Note that this dialogue partner still believes that communication occurred to some extent.

In human-computer dialogue, events where a spoken dialogue system interprets a user utterance with high confidence but that interpretation is incorrect are misunderstandings. However, *non-understanding* refers to the event where the dialogue partner fails to interpret an utterance at all. Non-understandings are noticed right away in human-human dialogues, while it may take additional turns for a dialogue partner to detect a misunderstanding. Non-understandings occur in a spoken dialogue system when the system fails to register any meaningful interpretation of a user utterance. When a misunderstanding occurs, one dialogue partner may initiate a repair, whereby the partner tries to find and amend an invalid assumption. When a non-understanding occurs, one partner may initiate a recovery by trying to get the conversation back on track [126].

### 2.1.1 Human-Human Miscommunication

According to Clark, referring to objects or ideas in the world is a collaborative process between a speaker and a listener [27]. The speaker of the concept is trying to convey that concept to the dialogue partner, while the listener is attempting to register the concept and display understanding to the speaker. Misunderstandings occur when the listener confidently interprets a concept from the speaker, but that interpretation does not match the speaker's intentions [51]. Because of this, the listener's interpretation is currently in that person's common ground, which is not aligned with the speaker. If the misunderstanding is detected, the dialogue partners collaborate to align their common ground. The misunderstanding could be caused by any number of factors, such as misrecognizing the speaker's speech, failing to register a gesture or gaze cue, or disambiguating a referent in a manner different from the speaker's intention. If the listener fails to register the speaker's signal (i.e., identifying that the speaker spoke), words, or intention, then a non-understanding has occurred.

As we introduced in Chapter 1, situated grounding problems are where one dialogue partner fails to uniquely map an instruction or concept to the environment in which one or more of the partners are situated. Two types of situated grounding problems are referential ambiguities and impossible-to-execute instructions. *Referential ambiguity* occurs when a speaker provides insufficient information to determine which specific referent among a set of candidates is the intended one. *Impossible-to-execute instructions* occur when the agent cannot produce a single plan.

### 2.1.2 Miscommunication in Spoken Dialogue Systems

Spoken dialogue systems most commonly operate in domains where users are accessing information or are commanding an agent to perform domain-specific tasks. The most common cause of spoken dialogue system miscommunication is the inaccuracy of the system's automatic speech recognizer (ASR). Speech recognition problems have several causes: speaker variation (and their associated acoustic properties), words the user says that are outside of the vocabulary of the system, and speech disfluencies like false starts, self-corrections, and other acoustic occurrences that cannot be lexically transcribed.

The role of ASR in spoken dialogue system success has received a considerable amount of attention. Several groups report that spoken dialogue systems performance is often dependent on the performance of the ASR [114, 140, 142]. [140] and [142] both report that speech recognition performance was the single best feature to forecast user satisfaction. Experimental evaluations in the field suggest users prefer systems with constrained vocabularies and system-led initiative, provided they are better at speech recognition [139].

Situated grounding problems, in terms of spoken dialogue systems for situated agents, have not received such attention. This makes them a good candidate for exploration. One study suggests that situated grounding problems like ambiguities occur often in situated human-robot dialogue [76].

## 2.2 Error Detection in Spoken Dialogue Systems

Spoken dialogue systems need to be able to detect miscommunication. Otherwise, systems will act on imperfect information. In the literature, misunderstandings are detected by monitoring the concepts under discussion in a dialogue task. Non-understandings are typically detected by identifying poor speech recognition performance or parse coverage.

Several groups have developed miscommunication detection methods for spoken dialogue systems. One method monitors results from an automatic speech recognizer, which can produce a value indicating its level of confidence in interpreting a word or phrase. A spoken dialogue system uses this information to determine whether or not it correctly interpreted a user's utterance. These *ASR confidence annotations* further help the system decide how to address a failed recognition if recognition confidence is below a threshold. Several groups have

studied ASR confidence annotation at the level of phonemes, words, and whole utterances [7, 26, 31]. Other groups have used other sources of information to detect errors. San Segundo et al. [113] used information from the language model and parser to train a neural network-based classifier that detected misrecognized words and out-of-scope phrases. Walker and colleagues [141] evaluated many features of the natural language understanding and dialogue management components of a telephony-based dialogue system to detect misunderstandings and other errors. Bohus and Rudnicky [13] studied several approaches to combining evidence from the speech recognizer, parser, and dialogue manager to estimate confidence about an input utterance. Litman et al. [75] considered prosodic features for detecting ASR errors.

Several groups have studied approaches to automatically detecting when the user notices occurrences of dialogue system misunderstandings or attempts to correct the system. Swerts et al. [130] found that user attempts to correct the system can be detected because they have prosodic properties that make them more difficult to understand than other utterances. They report high accuracy in detecting these occurrences by combining evidence from the prosodic features of the utterance, the speech reocgnizer, and past dialogue turns. Levow [74] demonstrated that prosodic features of the utterance can distinguish between the first user utterances to the system and later attempts to correct system-detected misunderstandings and non-understandings.

Detecting and diagnosing errors in spoken dialogue systems has received a considerable amount of attention and has established processes. However, all of these detection approaches process the spoken language input, and not any possible mapping to environments. Moreover these approaches all operate in non-situated domains. Situated grounding problems have not received this level of attention, but the success of ASR approaches could be extended by including plan and environment information, as we do in this thesis.

## 2.3 Dialogue Error Recovery

Communication strategies that allow dialogue partners to recover from errors are essential to allow derailed conversations to get back on track. In human-computer dialogue, error recovery strategies have been studied from the perspective of both the human and the dialogue system.

Several groups have attempted to understand human error recovery strategies by

conducting Wizard-of-Oz experiments where participants faced simulated speech recognition problems from a dialogue system. People use several strategies for repairing communication breakdowns. Zollo found that human wizards used several strategies that machines were not capable of using – like prompting the participant just about a single word or phrase [149]. Those strategies largely did not exist in dialogue systems at the time. Other strategies included explicit and implicit confirmation, asking to repeat a word or phrase, clarification requests about ambiguities, and disregarding any information under discussion that is not crucial to the dialogue task. Skantze [124] found that human wizards acting as systems in a map navigation domain avoided explicitly declaring a non-understanding and instead opted to move the dialogue forward by asking alternative questions related to the task.

The dialogue system research community has developed several recovery strategies for misunderstandings. These boil down to explicit and implicit confirmation strategies. Moreover these two recovery strategies are common in nearly all spoken dialogue systems capable of error handling [16]. There is also general agreement about the pros and cons of both strategies [10]. Explicit confirmations allow the system to constrain the user response to a confirmation (e.g., yes/no), but also take an additional dialogue turn to complete. Implicit confirmations display the system's understanding of a user utterance, and allow the user to interrupt the system (i.e., barge-in) if there is a misunderstanding. Without user intervention, the system will assume its interpretation is correct for implicit confirmations. Present-day dialogue systems are better equipped to handle explicit confirmations because their responses are easier to predict than responses to implicit confirmations. In addition, when there is a problem and the system uses implicit confirmation, the user may be unsure what to say to correct the system, which can lead to an interaction breakdown. Krahmer et al. [65] conducted an extensive analysis of explicit and implicit confirmations. They identified positive and negative signals users made when the dialogue system used these strategies, and were able to automatically detect dialogue system errors based on these cues.

Recovery strategies for non-understandings operate at a coarser level than misunderstandings because the system interprets little to no part of the user's utterance. Dialogue systems typically repeat the last phrase, indicate a failure to understand the user, and ask the user to repeat [10]. However, there have been attempts to better diagnose non-understandings. For example, Gorrell et al. [41] attempt to better understand the type of problem by classifying any known

information from the non-understanding into one of a set of help options. Those help options attempt to give the user more feedback on how to speak to the system given results from a domain classifier (e.g., a *"when"* versus *"where"* question). System developers of a large-scale bus information dialogue system suggested a method for better diagnosing non-understandings by extracting words from the non-understood input and offering the user alternative ways to answer the system that the system could anticipate [107].

Bohus [10] offers several human-computer recovery strategies for non-understandings. The most simplistic strategies include lexically entraining users to words and phrases the system can interpret, offering a *"help"* option, using alternative methods of input like DTMF (touch-tone), and backing off to system-directed control. Other strategies from his work include proposing an alternate task-related question, asking for a short answer, providing an example of what the system can understand (i.e., *"you-can-say"*), and offering advice on how to speak to a dialogue system (e.g., more slowly and softly). Skantze [126] has also investigated methods to overcoming errors in spoken dialogue that exist in human-human communication.

The subset of dialogue systems strategies that have been studied throughly in the literature mainly deal with recovering from errors related to spoken language interpretation. Recovery from speech recognition errors has been the primary focus. In this thesis, we study error recovery strategies for communication problems that happen even when the speech signal is interpreted correctly.

## 2.4 Chapter Summary

In this chapter we presented background material about miscommunication in dialogue and related work about detection and recovery from dialogue system errors. Studies exist for both human-human dialogue and spoken dialogue systems, but those for the latter focus on miscommunication stemming from speech and language interpretation problems. We showed that there is a gap in the literature regarding detection and recovery of situated grounding problems.

Chapter 3

# Spoken Language for Agents in Physically Situated Contexts

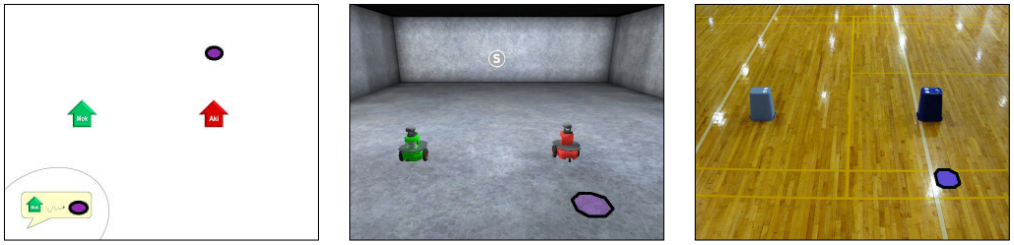
Spoken dialogue between people and physically situated agents like robots will require communication about space and distance to perform navigation tasks. It is therefore useful to understand the nature of the language people would use to give such instructions. We have studied how people give route instructions to agents across presentation formats (schematic, virtual, or natural) and how the complexity of the route influences the content of instructions [81]. In addition, we have publicly released the data we collected as the TeamTalk Corpus [83].

## 3.1 Presentation Formats for Agents

We performed a study to understand how people gave route instructions to a robot. Although many groups have confirmed the importance of landmarks in such instructions (e.g., [32, 59, 77, 79, 90]), it is unclear if their role changes based on the presentation format (e.g., two-dimensional schematics, three-dimensional virtual spaces, natural environments). Instructions given in a physical space may differ from those where only a three-dimensional virtual scene or schematic map are available. For example, landmarks may play a greater role in scenarios where people cannot establish a good sense of distance. The nature of instructions may change based on the complexity of the route that a robot needs to take. In this work, we collected spoken langauge route instructions in open spaces across these dimensions. This work provides useful information about navigation task design,

| Presentation Format | # Participants | # Route Instructions | Transcription Method |
|---|---|---|---|
| Schematic | 10 | 640 | In-house |
| Virtual (*distance*) | 7 | 448 | 6 MTurk, 1 In-house |
| Virtual (*no-distance*) | 7 | 445 | 6 MTurk, 1 In-house |
| Natural | 11 | 86 | In-house |

Table 3.1: Corpus information about the types of route instructions from participants.



(a) Schematic environment (b) Virtual environment (c) Natural environment

Figure 3.1: Stimuli from the (a) schematic, (b) virtual, and (c) natural scene recording sessions. All scenes had 2 robots: Mok (left) and Aki (right), and a designated goal location (circled and marked in purple). Mok was the moving robot [81].

as how we present robots to people can influence the overall interaction.

### 3.1.1 Method

We conducted three rounds of experiments where we instructed participants to "tell" a robot instructions that would move it to a target location. We presented scenarios to participants in one of three formats: two-dimensional schematic, three-dimensional virtual, and natural scenes. All scenarios had participants view a static depiction of two robots (named "Mok" and "Aki") with a specified goal location. After viewing the scene, participants provided a verbal route instruction that would move Mok to the goal location.

The experiment instructions told participants to act as observers in the scenarios, so they could not use themselves in the instructions. They were also told to assume that Mok could understand natural language and to use natural language in their instructions (i.e., no special phrasing was necessary). Participants were also informed that they could use the orientations of the robots in their instructions. The

experimenter instructed participants to first think about a scene (thereby avoiding self-correction when speaking), then record all of the instructions they would give to the robot to move to the destination. Participants recorded themselves while viewing the scene using a close-talking microphone.

For each experiment session, we varied the orientations and locations of the robots and goal location. Each stimulus presented the robots Mok and Aki in one of four orientations: pointing forward, right, left, or backward. There were also four possible goal locations, directly in front of, to the right of, to the left of, and behind Aki. Each possible configuration was presented to participants in a randomly generated order, yielding sixty-four route instructions per experiment.

Participants viewed either two-dimensional schematics, three-dimensional virtual scenes, or natural configurations of the stimuli.

- The two-dimensional schematics presented a birds-eye view of the scene, where the robots had clearly marked orientations.
- The three-dimensional virtual scenes showed an eye-level view of virtual Pioneer P2DX robots. In this variant (three-dimensional), half of the participants were also informed that the distance between the two robots was seven feet, which provided a sense of scale to the virtual scene. This gave participants a chance to use absolute distances in their instructions.
- Natural configurations saw the robots presented as plastic bins, with eyes to indicate orientation. Again participants viewed stimuli at an eye-level view. However, in this group, we reduced the number of orientations we presented for Mok and Aki (Mok was only positioned right and left, while Aki was kept in the same orientation for all trials). In the previous studies, the other orientations for Mok and Aki did not change how frequently people used landmarks in their instructions, and were removed for this task. Participants viewed configurations of the robots in-person, so only verbal instructions were collected.

Thirty-five participants were recruited for this experiment, with ten allocated to the schematic scene variant, fourteen to the virtual scene variant (seven had distance scale), and eleven to the natural variant. This yielded 640 instructions, 896 instructions, and 88 instructions, respectively, from the three variants (see Table 3.1). All transcriptions followed the CMU Communicator guidelines [9], with virtual scene and natural transcriptions collected using workers from Amazon Mechanical Turk [87]. In both the schematic scene and virtual scene variants, we

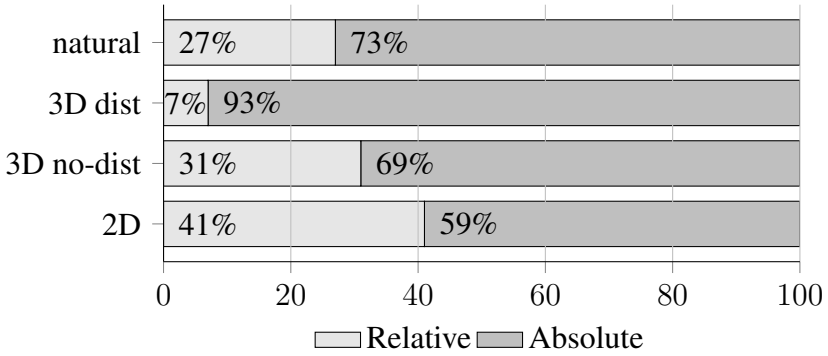



Figure 3.2: Proportions of relative steps to absolute steps for 2D schematics, 3D virtual scenes, and natural scenes (with a 1% margin of error).

logged the time between when participants viewed a scene and started recording themselves (i.e., *thinking time*).

### 3.1.2 Measurements

We measured the following outcomes in this study:

- The number of discrete actions per instruction (i.e., steps). A *step* was any action in an instruction that required Mok to move to a subgoal (e.g.,"turn right and go forward two feet" consists of two steps).
- We further annotated steps as either absolute or relative in nature. *Absolute steps* were explicit directions with metric or similar distances (e.g., "move forward two steps", "turn right", "go forward a foot"). *Relative steps* included Aki, the static robot and only landmark in the scene (e.g., "move forward until you reach Aki", "rotate left until you face Aki").
- Word count per instruction (excluding utterance-level restarts and mispronunciations)
- Thinking time (s) - the time participants took to formulate instructions

### 3.1.3 Results

We analyzed these measures in the four participant groups, two-dimensional schematics without distance information (i.e., "2D"), three-dimensional virtual scenes with distance information (i.e., "3D dist"), virtual scenes without distance information (i.e., "3D no-dist"), and natural scenes (i.e., "natural"). We found that

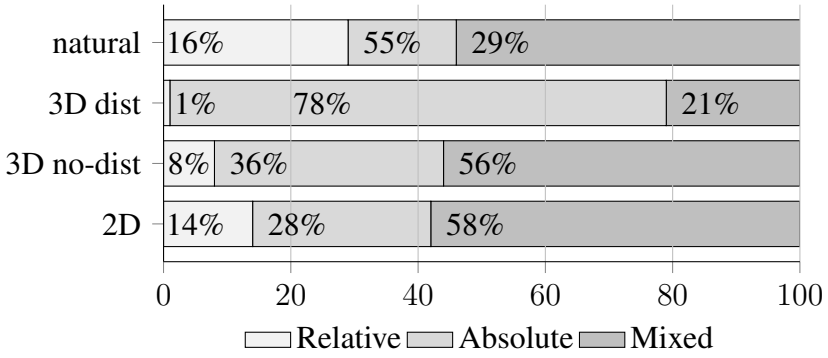



Figure 3.3: Proportions of instruction types for 2D schematics, 3D virtual scenes, and natural scenes.

sense of scale influenced participants' use of landmarks in their instructions. When fewer scale cues were available to participants, they tended to refer to landmarks more often. Figure 3.2 presents the proportions of absolute to relative instructions in the four groups. Participants from the two-dimensional schematics group used landmarks most often, while participants from the three-dimensional scenes group where we explicitly provided distances used them least often.

The instructions for the natural group did not inform them of any distances, since participants should have been able to infer the distances themselves. Somewhat surprisingly, their instructions had a proportion of landmarks (i.e., relative steps) that was closer to the two other conditions where participants did not have an explicit sense of scale. This result highlighted the prevalence of landmark use in natural environments even at close ranges. We also observed that absolute steps made up nearly 94% of the steps participants from the 3D dist group gave in their instructions. Comparing this result with those from the natural group suggests that when people can't determine a sense of scale for an environment, they can be primed to formulate instructions differently from when they can.

We observed differences in our measures when we varied the presence of a sense of distance in the instructions, which was exclusively for the virtual scene groups. Participants from the 3D no-dist group used Aki as a landmark in their instructions much more frequently than the 3D dist condition, confirming our hypothesis. The mean number of relative steps per instruction was around five times greater in the 3D no-dist group than the 3D dist group (1.0 relative steps per instruction compared to 0.2, $F[1,12] = 4.6$, $p = 0.05$). With respect to absolute steps, We observed that the 3D dist group used them significantly more (3.3 absolute

steps per instruction) than the 3D no-dist group (2.4 absolute steps per instruction, $F[1,12] = 5.5$, $p < 0.05$). These results suggest that between the two groups participants had very different strategies for producing instructions.

Route instructions were consisted either entirely of absolute steps, entirely of relative steps, or a mix of the two. Route instructions with landmarks also tended to include absolute steps, as shown across all conditions in Figure 3.3. Participants in the virtual scene groups distributed relative steps (i.e., those including landmarks) among their instructions. Across all conditions, there were more entirely absolute instructions compared to entirely relative ones. This suggests that sequences of absolute steps can be enough in close-range instructions, while relative steps depend on some metric distances to be sufficient.

The goal location resulted in different measures in the scenarios. When observing all instructions, participants used the fewest words per instruction (on average fourteen words fewer per instruction) when the goal was closest to Mok ($F[3, 1580] = 252.2$, $p < 0.01$). Thinking time was also lowest overall when the goal was closest to Mok (on average 1.3s lower). The goal location also impacted when participants included landmarks in their instructions. When a landmark was in between Mok's current location and the goal, partcipants referred to it much more often than otherwise. This yielded a high proportion of relative steps for these configurations. For example, in the 2D participant group, instructions for this type of configuration featured a propotion of 45% relative steps to 55% absolute steps (other configurations featured relative step proportions of 33-38%). This can help us determine how to design scenarios for future experiments to influence landmark use in participant instructions. There were no significant differences in word count or thinking time when comparing the configurations of the two robots.

## 3.2 The TeamTalk Corpus: Route Instructions for Robots

The TeamTalk corpus is a corpus of route instructions from the study presented in Section 3.1. The corpus has over 1600 route instructions in total, grouped into three categories, those for robots situated in (1) a schematic environment, (2) a virtual environment, and (3) a natural environment. This includes the audio recordings and corresponding transcriptions. The corpus also contains step annotations (absolute and relative) for each instruction. There were a total thirty-five participants that

provided instructions for the corpus, all fluent English speakers (some were non-native English speakers).

Participants provided instructions to a robot that needed to move to a marked location. The environment contained two robots and a symbolic destination marker, all within an open space. The corpus contains the collected speech, speech transcriptions, stimuli, and logs of all participant interactions from the experiment. Route instruction transcriptions are divided into steps and annotated as either metric-based or landmark-based instructions. This corpus captured variability in directions for robots represented in 2-dimensional schematic, 3-dimensional virtual, and natural environments, all in the context of open space navigation.

### 3.2.1 Robot Navigation with Route Instructions

There is general agreement that navigation is essential for mobile robots to accomplish tasks. Spoken language interaction is one way to move robots, by communicating about space and distance via *route instructions*. This section describes a corpus that we believe will help the human-robot dialogue community better understand the nature of spoken language route instructions when they are directed toward robots. The corpus contains over 1600 route instructions from a total of thirty-five participants, all fluent speakers of English (some were non-native English speakers).

Although dialogue will be a necessary aspect of understanding people's language interaction with robots, we must first capture their initial intentions when they give robots directions. We collected a corpus for this specific purpose: Given a static scenario with a robot capable of understanding natural language, what will people say to navigate the robot to a designated location? All directions required the robot to only move within an open space, and not around a more complex environment. There was only one landmark: another visible robot near the designated location. This environment setup allowed us to systematically vary the configurations of the robots in the scenarios and observe how people adjusted their route instructions to the changes.

Direction giving has been of enduring interest to the natural language processing and robotics communities; many groups have collected and released similar corpora to the TeamTalk corpus. Perhaps the best known is the HCRC Map Task corpus, a collection of dialogues that has one person (a *direction giver*) provide directions to another person (a *direction follower*) to move along a prescribed path on a shared

> Mok turn right / go forward seven feet till you run into Aki / turn right again / go forward three feet stop / turn left / go forward a foot

Figure 3.4: Example route instruction from the TeamTalk corpus with steps ('/').

map [2]. The SCARE [128], GIVE [39], and CReST [37] corpora are dialogue corpora with a similar navigation task, but present environments in different ways (for SCARE and GIVE the environment was a virtual world; for CReST the environment was a computer-generated schematic overhead view). The IBL corpus [70] captured people's spoken language directions to a static robot in a real-world miniature model of a city. The corpus collected for building the MARCO [79] route instruction interpreter captured people's typed instructions for navigating another person around a virtual environment. The TeamTalk corpus consists of spoken language directions within the confines of open space navigation, systematically varying location and robot orientation. None of the corpora above apply systematic variation to stimuli to elicit directions. In addition, our corpus only captures what the giver of directions says (similar to the IBL corpus but unlike the others), since the follower is described as an artificial agent.

### 3.2.2 Open Space Navigation Task

In the open space navigation task, participants provided verbal instructions that would let a mobile robot move to a specified location [81]. Participants viewed a static scene on a computer monitor with two robots, Mok and Aki, and a destination marker (Mok was the actor in all scenarios). The experimenter told participants to give instructions as if they were observing the environment but not situated in it. In other words, the robots could hear participants but not see them (so participants could not mention themselves in their instructions).

The experimenter indicated that Mok was capable of understanding natural language. Participants were free to assume they could use naturally occurring language when they provided instructions to Mok (i.e., no formally structured language was necessary). The experimenter showed participants the orientations of the robots and told them they could use these orientations in their instructions. The robots did not move in the scenes, so participants did not see how the robot responded to their instructions (participants received no feedback on their instructions).

All scenarios required participants to give all the necessary steps to move Mok to the goal destination in a single recording. Participants accomplished this by using a recording interface that did not disrupt their view of the robots' environment. They spoke their instructions through a close-talking headset microphone. The recording interface allowed participants to playback their instructions and re-record them if necessary. The experimenter also told participants to think about their instructions before recording; this was meant to deter participants from replanning their instructions as they spoke. We time-stamped and logged all interaction with the recording interface, and provide it as part of this corpus.

We varied the orientations of the two robots, Mok and Aki, and the location of the destination marker for the recordings. The robots were each in one of four orientations, directly pointing forward, right, left, or backward. The destination marker was in one of four locations, directly in front of, behind, to the left of, or to the right of Aki. Participants viewed environment representations in 2-dimensional schematics, 3-dimensional virtual scenes, or natural scenes. For the schematic and virtual environments, we varied the three factors using a full-factorial design, resulting in 64 configurations (randomly ordered) of the robots and destination marker per session. Each participant for these recordings provided 64 instructions. Participants each gave 8 instructions when viewing natural scenes (destination varied four ways, Mok's orientation varied two ways). We describe these environments in detail below.

#### Schematic Environment

A segment of participants viewed 2-dimensional schematic representations of the two robots and a destination marker. The schematics presented a birds-eye view of the scene, with the robots presented as arrows and the destination marker symbolized by a purple circle (See Figure 3.1(a)). The arrows indicated the perspective of the robots. There was no sense of scale in the environments; participants could not use metric information when instructing the robot.

#### Virtual Environment

Some participants gave instructions to a robot situated in a 3-dimensional virtual environment. We developed the environment using a virtual map builder and USARSim, a robot and environment simulator that uses the Unreal Tournament game engine [24]. The environment contained two Pioneer P2AT robots and a

transparent purple destination marker. The environment was set to a standing eye-level view, at a height of about 1.8 meters (see Figure 3.1(b)). Walls in the space were far enough away from the robots that participants did not use them as landmarks. The recording setup used two monitors, one to show a full-screen representation of the environment, and one for the recording interface.

Half of the participants in this study were told that the two robots were seven feet apart (i.e., the *distance* condition); the experimenter did not specify any distance to the remaining participants (i.e., the *no-distance* condition). The distance condition allowed participants to provide instructions to the robot using metric information. The environment did not provide any other indication of scale.

#### Natural Environment

The environment for this group of participants was similar to the virtual condition, but participants gave instructions in-person (see Figure 3.1(c)). The robots were represented as bins in the space; eyes on the top of the bins indicated each robot's orientation. Participants were standing in a gymnasium, not sitting at a computer. No logging information was collected, only verbal recordings.

### 3.2.3 Participation

Thirty-five self-reported fluent English speakers participated (ten viewed the schematic environment, fourteen viewed the virtual environment, eleven viewed the natural scene environment). There were twenty-two male and thirteen female participants, ranging in age from 19 to 61 ($M$ = 28.4, $S.D.$ = 9.9). We recruited participants by posting to the Carnegie Mellon Center for Behavioral Decision Research webpage[1]. Participants earned $10.00 for completing the task.

The corpus contains 1,619 verbal route instructions and is summarized in Table 3.1. On average there are 23.1 words per instruction ($S.D.$ = 18.2). Total word count was 37,442 with 751 unique terms (some terms are words; others are annotations). There are 640 route instructions directed to robots in the schematic environment (sixty-four recordings from each of ten participants). These instructions were transcribed in-house according to the CMU Communicator transcription conventions [9]. The corpus also contains logs of participants' interactions with the recording interface. A past study used these logs to derive the amount of

[1] http://www.cbdr.cmu.edu/experiments/

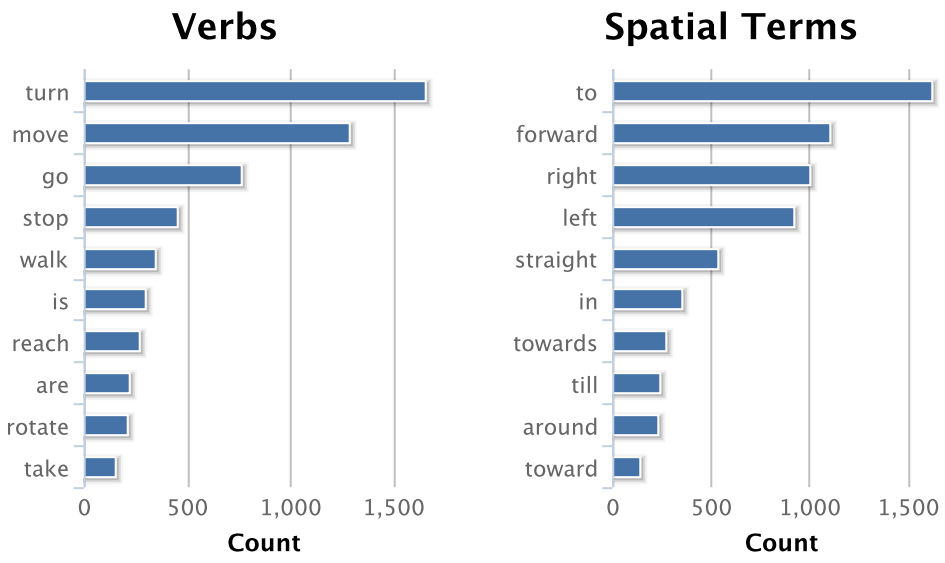



Figure 3.5: Histograms of the most frequent verbs and spatial terms from the TeamTalk corpus.

time participants took to formulate their instructions after viewing a scene (i.e., "thinking time") [81].

We collected more route instructions from the virtual environment scenarios (893 instructions[2], 64 from each of fourteen participants). This segment of the corpus also evenly divides instructions into those collected from participants that had an absolute sense of scale in the environment (labeled in the corpus as "distance" instructions) and those that did not (labeled in the corpus as "no-distance" instructions). Workers from Amazon Mechanical Turk[3] transcribed recordings from twelve participants using the same guidelines as before (see [87] for a description of the approach). The natural study yielded 86 route instructions[2], all of which were transcribed in-house.

### 3.2.4 Corpus Annotation

As the corpus was used for a study of direction giving [81], the route instruction transcriptions were annotated for discrete actions per instruction (i.e., 'steps') and

[2]Three virtual trials and two natural trials are missing due to a recording interface issue with two participants.

[3]https://www.mturk.com

disfluencies. We describe the annotation coding scheme below.

### Step Annotation

We divided participant recordings into discrete steps. In this corpus, we defined a *step* as a sequence of words that represented any single action that would move Mok (the acting robot) to a subgoal. The following example has two steps, divided by a forward slash '/':

*Mok turn left / and move forward three feet*

The first step in this instruction is a rotation, while the second moves Mok to the goal destination.

We annotated steps as one of two types, absolute steps and relative steps. An *absolute step* is a step that explicitly includes a measured distance or turn rotation. This includes simple turns (e.g., "turn right") that we assume to be rotations of ninety degrees. Both of the steps above are absolute steps. Discretized measures that aren't metric distances are also absolute steps (i.e., "move forward three steps"). Absolute step examples:

*move right four feet*
*move forward about seven feet*
*turn left ninety degrees*
*turn around*

A *relative step* is one that mentions a landmark as a subgoal or reference point. In this study, the only possible landmark was Aki, the static robot in all scenarios. Below are some examples of relative steps:

*go forward until you're in front of Aki*
*go forward half the distance between you and Aki*

A single annotator divided transcribed recordings into steps and labeled them as absolute or relative. Marge and Rudnicky [81] discuss the proportions of absolute and relative steps for the schematic and virtual scene subsets of the corpus. 58.9% of the steps in schematic scene segment of the corpus had absolute steps, while the remaining 41.1% were relative steps. We divided step analysis for the virtual scene instructions into *distance* and *no-distance* segments. The *no-distance* instructions had a similar breakdown to the schematic instructions; 68.5%

of the steps were absolute and 31.5% were relative. The *distance* instructions were markedly different in nature; most were absolute (93.5% compared to 6.5% relative). The study that used this corpus found that when aware of distance, participants included it in nearly all instructions.

### Disfluency Annotation

The schematic instructions in this corpus were transcribed in-house; the transcriptions include annotations for disfluencies. The three types of disfluencies we annotated were fillers (e.g., *uh* or *um*), mispronunciations (i.e., when the speaker does not pronounce an intended word) and false starts (i.e., when the speaker speaks then abruptly begins again).

The in-house transcriber annotated fillers by surrounding fillers by forward slashes, as shown below:

*I think **/uh/** you should turn left*

Mispronunciation annotations have the uttered word fragment and what the transcriber believed to be the intended word. Square brackets surround the entire mispronunciation annotation and within the brackets the parenthesized text has the intended word, as follows:

*move **[sev (seven)]** feet forward*

False starts indicate all words in an uttered phrase that were abruptly abandoned for a new phrase. Angled brackets surround the abandoned phrase, like below:

*<**turn right**> no turn left*

Often a mispronunciation and false start occur in the same uttered speech, these occurrences include both annotations. See below for an example:

*<**turn [ri (right)]**> no turn left*

The crowdsourced transcriptions did not have these annotations because the crowdsourced transcriptions were found to be word-accurate. Results pertaining to their accuracy correctly marking disfluencies is forthcoming.

### 3.2.5 Session Log Information

The experiment recording software logged all interface activity. The logs recorded four or more items per trial: presentation of a stimulus then starting, stopping, replaying, and accepting a recording (we define trial here as one of the sixty-four recordings per participant). The log information can be used as an indicator of cognitive load for participants as they formulated instructions. More specifically, the elapsed time between when participants viewed a scene and pressed the 'Record' button can measure load on participants as they formulated instructions. When 'thinking time' was long, we hypothesize that the participant incurred a high cognitive load while formulating an instruction for that scene.

## 3.3 Chapter Summary

This chapter described a set of experiments exploring human language use with physically situated agents. The results from this work shaped the experiments we performed in this thesis. In any scenario where we are collecting instructions from participants, we use 3D situations whenever possible. We presented the next-best presentation format to a natural one, 3D virtual scenes. In this thesis we not prime participants by providing an explicit sense of scale, but may provide implicit perspective by showing landmarks of known heights (e.g., chairs and tables) in the virtual scenes.

We discussed a framework for designing studies to collect spatial language, particularly in the form of route instructions. The results can inform the design decisions of robots that can interpret instructions that include spatial langauge. Results suggest that the composition of route instructions to robots are similar under the right circumstances in natural and virtual presentation formats. However, the more metric information people have about an environment (especially in one where they cannot develop a sense of scale for themselves), the more likely they are to include such information in their instructions. These observations can help inform the design of situated language understanding agents.

This chapter also described the TeamTalk corpus, a new corpus that contains the speech and transcriptions of route instructions directed to robots. Fluent speakers of English gave verbal directions to a robot in an open space that would allow the robot to move to a specified location. The corpus captures speakers' intentions when giving navigational directions and provides useful information for researchers

studying spatial language and route instructions. The most immediate impact of this corpus was to help build grammars for human-robot dialogue systems and for general language analysis (i.e., vocabulary building and language modeling, in combination with other resources). The TeamTalk Corpus can be found at `http://www.cs.cmu.edu/~robotnavcps`.

This body of work captured people's intentions when giving navigational instructions. This helps us understand how people give navigational instructions across presentation formats where the follower is considered an agent. In this thesis we investigate how agents can overcome situated grounding problems that happen when they receive these kinds of instructions from people.

Chapter 4

# Grounding as an Error Handling Process

Grounding is the process by which two or more dialogue partners align information through conversation. It has long been studied, both in experimental evaluations and via implementations with theoretical foundations. Most approaches to grounding in dialogue systems only incorporate information from the speech and language input. This chapter describes a model of grounding that also includes planning and environment information.

## 4.1 Related Work

Even with sophisticated error detection and recovery, dialogue systems further benefit from unified frameworks that learn patterns about decisions the dialogue system makes when handling errors. Typical spoken dialogue system authors compose heuristics for handling errors in the system by hand. These types of systems are only equipped to handle a limited range of possible error sequences. Some strategies that system authors may believe to be intuitive may not necessarily be optimal in practice. Statistical approaches exist, primarily using reinforcement learning techniques. These approaches typically work well for dialogue systems for limited domains like information access. In this thesis, we explore an instance-based learning approach due to the limited data availability of situated interactions, where dialogue handling decisions are continually reassessed as the dialogue agent records more interactions.

### 4.1.1 Models of Discourse Grounding: Theory

In human-human communication, Clark [27] proposed that people ground the information and experiences they share with dialogue partners. When a dialogue partner attempts to ground something under discussion, they are attempting to add it to the common ground they share with their dialogue partners (i.e., the information and experiences they share).

#### Contribution Model

Clark and Schaefer [29] formally defined grounding in the Contribution Model, which states that at any given point in the dialogue, dialogue partners are contributing to a shared discourse. At the same time, the contributor is attempting to ensure that dialogue partners register the contributor's intention and can add that to their common ground. There are two phases to every contribution: the *presentation phase*, where the contributor is presenting a signal (e.g., a spoken language utterance) to a partner, and the *acceptance phase*, where the partner attempts to understand the contribution, add it to their common ground, then display evidence (e.g., a signal) indicating that the contribution has been understood.

Along with the Contribution Model, Clark proposes several levels of understanding, ranging from understanding that a dialogue partner has uttered something to understanding the full goals and intentions of the dialogue partner [27]. These levels of understanding drive the type of recovery strategies humans use to recover from miscommunication. In this thesis, we deal with communication problems at the highest level of Clark's model, the Joint Projects level, where dialogue partners attempt to coordinate their actions to complete the current goal (i.e., project). We argue that situated grounding problems exist at this level. These problems occur because information about plans is not grounded between dialogue partners. In cases where a human is giving navigation commands to an agent, the agent interprets the user's language correctly, but fails to map a unique plan of action to execute in the environment (thus the two partners are unable to accomplish the *joint project* of moving the robot to a new location).

#### Conversation Acts Model

Traum developed the Conversation Acts model, which attempts to address some of the limitations of the Contribution Model [135]. This model includes Speech

Acts like turn-taking and grounding [118]. He defines several "grounding acts" and ways to handle sequences of them.

#### Exchange Model

Cahn and Brennan [20] present an extension/reformulation of the Contribution model designed for human-computer interaction. The grounding process is formalized in this model as building dialogue structures known as "exchange graphs" over the course of the dialogue. Exchanges are pairs of contributions (the contributions defined in the Contribution Model) linked by their complimentary roles: the first contribution proposes, and the second executes a jointly achieved task. An exchange serves as a mapping between contributions and dialogue tasks.

#### Other Models

Nakano et al. [95] developed a model of grounding for verbal and non-verbal signals that was designed for embodied conversational agents. Other attempts at extending the contribution model exist, but none take on statistical approaches to modeling grounding [20, 28, 88].

### 4.1.2 Grounding as Decision Making Under Uncertainty

While the models of grounding above provide insight into how humans execute grounding processes, there are scaling issues associated with adapting them for dialogue systems. Heuristics encapsulate the processes behind these models; such designs can become complex and fail to generalize. Paek and Horvitz propose a dialogue architecture, Quartet, designed specifically for managing grounding in computational systems [100]. This approach views conversation as taking actions under uncertainty, and, moreover, grounding as decision making under uncertainty.

To determine the next system action, Quartet calculates its beliefs about its grounding status across four of Clark's levels of understanding (from low to high these levels are *channel, signal, intention, conversation*). Its grounding status is a Bayesian network of five states: OKAY, CHANNEL FAILURE, SIGNAL FAILURE, INTENTION FAILURE, and CONVERSATION FAILURE. A probabilistic distribution for the likelihood of failure over these states is inferred from input at each dialogue step. The *channel level* models attention; it uses visual information from a camera to monitor whether or not the user is attending to the system. The *signal level*

models speech and language understanding; it uses speech recognition and natural language processing tools to monitor whether or not the user presented a signal to the system (in the form of speech, visuals, and/or text). The *intention level* assesses progress in completing tasks; it monitors its certainty about the user's current goal. Finally, the *conversation level* models the system's common ground and joint activity status with the user. The common ground is composed of contributions, i.e., mutual information between the user and system, inspired by Clark's Contribution Model. Without belief that a joint activity is underway, the system has no reason to continue with the dialogue.

Modules associated with the channel, signal, and intention levels contain a Decision-Based Transition Network (DTN), a finite state automaton with transitions between states associated with grounding actions [99]. When grounding problems arise, Bayesian inference determines the best transition to follow. Consider the DTN associated with the intention level, shown in Figure 4.1, which estimates the user's most likely goal given evidence. Set thresholds are checked against the probability value of the maximum likelihood goal. When the value is high, it can proceed with the task; otherwise it must initiate a grounding strategy, like confirming the goal or asking the user to elaborate, repeat, or provide more information. Confirmation transitions the DTN to what is presented in Figure 4.1 as the ACCEPT INPUT state, while the others transition the DTN toward asking the user a more general question (and thus embedding a side sequence in the dialogue). One of the limitations with this approach is defining the probability thresholds: the authors presented approaches that allow the user to define these thresholds (by asking questions about user preferences) or to collect values from psychological studies. One desirable technique that was not studied would be to learn threshold values from large amounts of user data based on task success.

When there is a grounding problem at the channel, signal, or intention level that requires more information to resolve, the system selects the best question to ask using a process known as Value of Information (VOI) [52]. VOI requests observations, in the form of signals, that best reduce the system's uncertainty. First, VOI calculates the expected utility of obtaining more information compared to the cost of asking about it [54]. When the benefit of gathering more information is greater than the cost of asking, it will prompt the user for more information. For example, in one of their applications known as the Bayesian Receptionist, if the user command had something to do with "shuttle", the system could ask a question resolving confidence about a recognized word (a signal level problem),

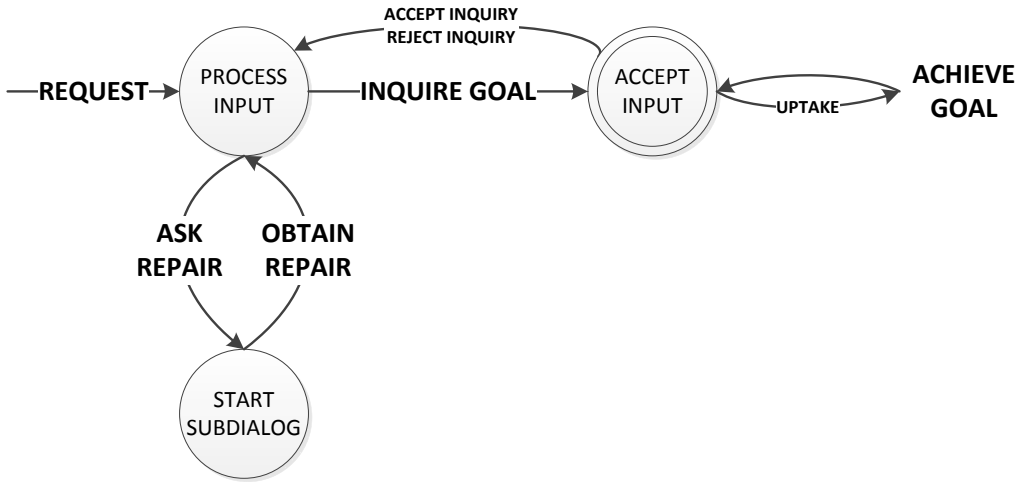



Figure 4.1: A Decision-Based Transition Network (DTN) for the Intention level of understanding in Quartet [55]. The DTN processes a user's request (e.g., "I'd like to book a shuttle"), along with any clarifications that might need to take place. Task execution here is equivalent to the uptake transition.

like "Did your request have anything to do with a shuttle?" Alternatively, if the system has belief that the user is not attending to the system, it can probe the user's attention (a problem at the channel level), like "Excuse me. I just asked you a question. Are you there?" The process will continue until the cost of gathering exceeds its expected utility. When this happens, the DTN will transition out of the side sequence.

After assessing the user's input and gathering more if necessary using VOI, the system must decide upon the next action. The overall control first obtains suggestions from each level of understanding on if that level believes there is a grounding problem, and what to do. Then the control selects an action based on the grounding status with the highest likelihood, coupled with it assessment of that level's reliability. For example, consider the situation where the system reports high confidence about assessing the user goal, but the user keeps correcting it. The overall control can reduce the reliability of the intention level module while looking for grounding problems at lower levels of understanding. This approach works well for making broad decisions about grounding. However, in complex tasks with many choices, the state space can grow to intractable levels. Their task model requires that the task decompose into increasingly more detailed steps and have likelihoods for each state reassessed given evidence.

### 4.1.3 Symbol Grounding: Theory and Applications

While discourse frames grounding as a top-down process, many in the robotics community take a bottom-up approach known as symbol grounding. Introduced by Harnad [47], symbol grounding permits agents to connect symbols to their low-level perceptual representations. In a robot system, for example, symbols can be words and the observables can be input streams from cameras and laser rangefinders. Harnad defines the symbol grounding problem as mapping symbols to meaning, where meaning is some internal representation that allows the agent to detect, identify, and execute tasks that refer to the symbols. Harnad argues that symbols themselves are not equivalent to their meaning. Consider the following phrases: "Barack Obama", "President of the United States", and "husband of Michelle Obama". Each of these phrases has the same referent (a particular individual that as of 2014 lives in the White House), but not the same meaning — each phrase reveals different information about the referent. Symbol grounding can be viewed as an internal negotiation between an agent's sensory projections and its symbols rather than a dialogue between two interlocutors. In Harnad's terms, symbols are objects that make up a symbol system, and a symbol system is a set of objects and rules governing those objects based strictly on their shapes.

Symbol grounding is the process by which an agent selects referents from a set of symbols. With symbol grounding, an agent is able to interact with the objects, events, properties, and states that comprise its surroundings. At the same time, the symbols it uses to represent the referents in its world should also be understandable by humans.

Harnad proposes a general solution to the symbol grounding problem: to derive direct connections between symbols and referents. The approach is a bottom-up process consisting of communication between an iconic representation, a categorical representation, and a symbolic representation. The iconic representation contains unique instances of the agent's sensory observations. The categorical representation learns the core, unchangeable, common features that make up general object and event categories using icons produced from agent's observations. Finally, the symbolic representation ties everything together; symbols that comprise a hierarchy of objects and events connect to the categories in the categorical representation. A daunting challenge when the symbol grounding problem was first proposed was how an agent could produce a categorical representation of objects and events. Today, one can train a classifier to detect different categories

based on features extracted from observations (for example, to detect doorways).

One noteworthy contribution of this approach is that it is bottom-up in design: once the categorical representation converges on a set of elementary symbols, more complex symbols can follow suit. Harnad claims that symbolic functions would then form the basis for an intrinsic, grounded symbol system. This system grounds the otherwise arbitrary shapes of symbols to icons and categories. With natural language as a symbol system, some elementary set of words would have groundings with the agent's perceptual features. All other word strings would inherit groundings from this elementary set. For example, the symbol "leopard" could inherit groundings from "cat" and "spots" if they were part of the elementary set. The elementary symbols would just need to be properly arranged in a taxonomy. In this thesis, the available sensory projects are an occupancy grid, location information, an apriori map of the environment, an ontology of environment surroundings, and the user's speech input.

Although symbol grounding is a reasonable approach in theory, it is unclear from Harnad's writings how to form a categorical representation. Historically, researchers have approached the symbol grounding problem for situated agents like robots in three ways [132]. The first method is to create symbol systems by hand that connect language to the agent's world. Each word maps to a predefined action and the agent's environment representation. While this method works well for spatial prepositions, which have predictable structure in the context of navigation, it relies heavily on heuristics. There is very little opportunity for learning to take place. The second method learns the meaning of words by directly linking them to the agent's perceptual features (e.g., camera images). This approach treats language input as sensory observations. The learning process uses various features intrinsic to the robot's observable signals. As such the approach can only produce a limited set of word groundings. In the third method, researchers pose learning symbol groundings as a form of translation from language to parts of the environment. This approach is flexible enough to learn symbol groundings for spatial prepositions, action verbs, and landmarks.

### 4.1.4 Clarification Requests

A spoken dialogue system is said to initiate a clarification request when it requests information immediately after detecting a partial or complete non-understanding. In the navigation domain, Tenbrink et al. conducted a study investigating the nature

of human-human navigational dialogue [133]. In these dialogues, direction givers directed followers around a 2-dimensional map. They found that clarification requests were essential to task success: 60% of the utterances from direction followers were clarification requests. Of these, a majority (57%) were for resolving a reference, which suggests that referential ambiguity is commonly addressed using clarification. Clarification requests they observed in addition to general reference resolution were "best guess" reference resolution (e.g., "like this?"), requests for repetition, spatial clarifications, and perspective clarifications. In my work, we will experiment with several of these clarification requests as recovery strategies for referential ambiguities.

Purver [105] developed an approach to generating clarification requests broad enough to cover a majority of questions about individual words and phrases. He composed a taxonomy of clarification requests, a semantic representation to handle them, and a proof-of-concept dialogue system. This work takes a linguistic view to handling clarification requests, where different types of requests have different surface forms (e.g., full reprises of what was heard by the listener and elliptical fragments). His representation for clarification requests, known as "contextual abstracts", were shown to be flexible enough to handle a variety of clarifications, but the process for selecting a request was heuristic. The handcrafted rules for this approach limit it to only clarifying individual words and phrases for specific cases. Such an approach could benefit from statistical methods that learn over time the optimal clarification requests to ask based on past experiences.

### 4.1.5 Learning Approaches for Dialogue Control

Heuristic methods are the most common ways to specify dialogue control policies. There is general agreement that explicit and implicit confirmations are strategies to handle misunderstandings, and there are benefits and limitations to both [65, 97]. These approaches manually establish thresholds that specify when to ask explicit and implicit clarification questions. Commonly these thresholds are determined from the speech recognizer's confidence annotator [62, 97, 145]. These thresholds are typically set by the system authors, but could be recalculated later, as in [12].

Statistical methods towards learning dialogue control policies typically cast the dialogue control problem as a Markov Decision Process (MDP) [58]. Dialogue researchers have used reinforcement learning to learn the decisions the system should make given the type of error the system detects and the possible recovery

strategies available [73, 115, 123]. To cast the dialogue control problem as an MDP, the states of the dialogue system should be abstracted into states that are general task components (i.e., dialogue states that behave similarly are grouped together) [10]. Actions in the dialogue system generally correspond to actions in the MDP. Lastly, rewards can be applied to desirable dialogue states (typically successful dialogue end states), and they can be tied to measurements of success appropriate for the dialogue system.

In his work, Bohus developed an online method for learning dialogue control policies that used MDPs in this fashion [10]. The system estimates the likelihood of success for selecting any given recovery strategy at runtime. Those likelihood estimates are then used to build a dialogue control policy for the current dialogue. An experimental evaluation demonstrated a spoken dialogue system learning a better recovery policy for an error than one initially set by the system authors. The learning method works for typical spoken dialogue system errors (i.e., features extracted from spoken language input like recognition confidence and parsing coverage).

## 4.2 Plan Grounding Model

In this thesis, we combine instance-based learning with a method for selecting grounding actions for plans; we call this the *plan grounding model*. The agent must select grounding actions to recover from situated grounding problems, such as resolving ambiguous references or impossible-to-execute actions. Those situated grounding problems relate to plans that the user and agent are trying to accomplish. Formally, a *plan* consists of an action with its corresponding referent(s). The key processes to the plan grounding model are summarized in Figure 4.2.

A successful user instruction resolves to an action with referents that the agent can uniquely identify. Each possible referent can be queried against the agent's knowledge base of objects in its surroundings.

We cast the problem as one where the agent's goal is to resolve the user's intended referent as efficiently as possible. In other words, the agent must attempt to confidently resolve references in the user's instruction to unique objects (i.e., a single match per referent) if they exist, and zero otherwise. In addition to interpreting the user's instruction, the agent must consider the user's speech and intention as sources of uncertainty. Speech uncertainty can be estimated from the confidence scores obtained from the agent's speech recognizer, while intention

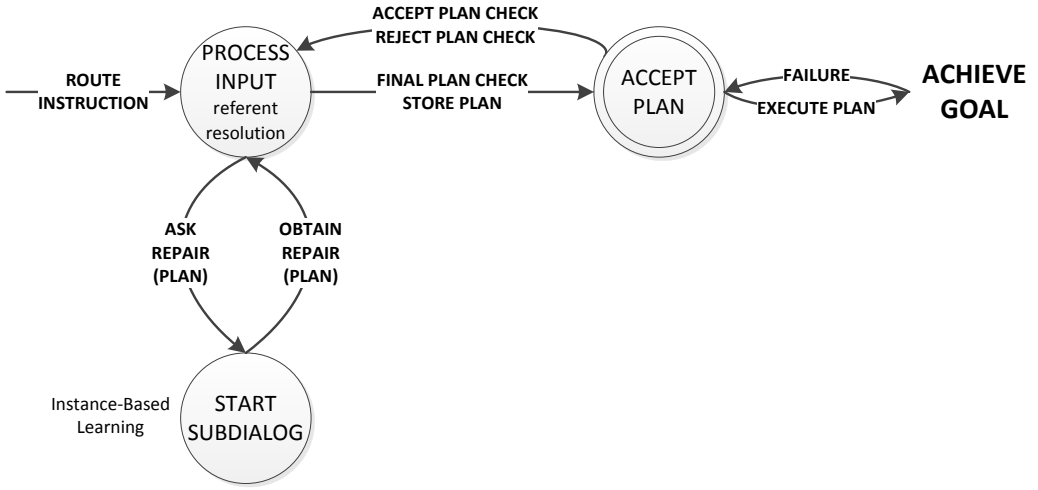



Figure 4.2: Plan error handling as a grounding process, extended from [55].

can be estimated from the agent's past interactions with users. An instance-based learning process stores all past interactions and estimates intention.

At its core, the plan grounding model decides *what to ask about* while also considering *how to ask about a problem* (if one exists). Consider the scenario where the user tells the agent to move to a "big blue box", but no such object exists in the environment. The choice of whether the agent should relax or remove the "blue" or "box" constraint depends on the context of the immediate situation, along with the words the agent should say.

Whenever a situated grounding problem arises, the agent must select the next question to ask considering the following factors: (1) *what to ask:* select the question that maximizes information gain and (2) *how to ask:* select the words and phrases that would most likely yield a successful interaction.

### 4.2.1 Question Content Selection

Determining *what to ask* means that the agent selects the next plan grounding action (asking about a grounding problem) that would effectively recover from the current situation. Moreover, the next best question is determined based on the agent's past history of interactions, combined with the current context. The context is a combination of the agent's present location, surroundings, and assigned task. The agent must often go through many possibilities to produce a question to ask the user. In turn the process selects the one question that would do the sufficient

work. We measure this by retrieving similar successful situations to the current observation. Only applicable ones to the current context can be used. We will elaborate on this approach with the $k$-nearest neighbors algorithm we selected in Chapter 8.

### 4.2.2 Measuring Costs of Adjusting Constraints

In some referent descriptions, the user is referring to one object out of many in the agent's surroundings. When the user's referent description is overspecified, that is, when it does not resolve to at least one match, the description needs to be altered and checked against the agent's surroundings again. If we view each word describing a referent as a type of constraint on the possible objects the user meant, then each constraint has different costs to remove or relax. In this thesis, the agent first defines a *state space* of possible instantiations of adjustments to the original referent description. The adjustments themselves serve as the *actions* of the state space.

The initial state space represents all combinations of the original (and longest) referent description. The original referent description is the start state. We can formally define the search space as follows:

State $s \in S$ = some combination of words based on the original referent description

Start state $start \in S$ = the original referent description

Action $a \in A$ = the removal/relaxation of words (*constraints*) from the original referent description

Path costs $c \in C$ = the number of removal/relaxation actions to reach the current state

Goal proximity heuristic $h$ =
$(Total\ \#Objects) - (\#Entries\ for\ Query\ q\ in\ State\ s)$

The agent can use Forward A* Search to find the cheapest action to take in the state space. An example state space for the referent "big blue box" can be found in Figure 4.3. Entering a state means that the agent will select that adjustment of the original referent description. The agent will then use that adjusted referent description in a recovery strategy for an otherwise impossible-to-execute

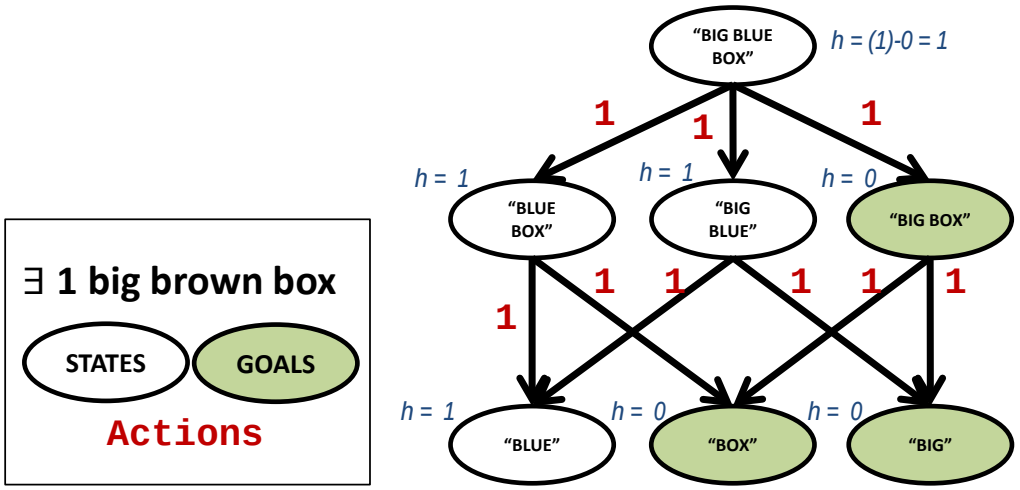



Figure 4.3: Refashioning referent descriptions as a search space.

instruction.

In the state space presented in Figure 4.3, each action has the same cost (1), while each state's heuristic value is dependent on the number of possible matches in the agent's surroundings. To improve performance, the agent's model should factor in the role of the word being removed from the referent description as part of the action cost. In the model, the removal costs are ranked using a data-driven method. In this approach, the model considers preferences from crowdsourced data and data collected in real time. Each removal can be one of two types: a *situated dimension* or an *basic object type*. A situated dimension can be a property associated with one of the following: an intrinsic property (color or shape), an egocentric proximity, proximity to an object, or the agent's history. The basic object type is a term describing the object as it exists in the agent's surroundings (e.g., desk, chair, door, etc.).

Using a cost assessment similar to the one proposed by Skantze [125], the agent first ranks the possible removals by how greatly each possibility reduces the search space upon confirmation or rejection. This results in possible constraint removals grouped by a search reduction cost value. Within each group, the candidate removals can then be ranked based on whether they are a situate dimension or basic object type. Finally, user preferences, whether heuristic or data-driven, can be considered. In general, there is a greater cost to removing the basic object type constraint than a situated dimension: removing an basic object type should open the space of possibilities much larger than a situated dimension.

### 4.2.3 Scenario Discussion

Costs are reordered at increasing levels of detail. The levels of detail are as follows from highest to lowest:

1. whether the constraint is being removed or relaxed,
2. whether the constraint is a situated dimension or basic object type,
3. if the constraint is a situated dimension, whether it is intrinsic (color or shape), egocentric proximity, object proximity, or history,
4. if the constraint is intrinsic, whether it is color or shape.

The agent presents a list of alternative referents to the user. Given the initial referent description provided by the user, enumerate the possible adjustments. Adjustments are either relaxations of a constraint (i.e., word) in the description (e.g., changing *big* to *medium* for the size property) or removals of constraints from the description. Details of the algorithm can be found in Algorithm 4.1.

**Scenario 1: Impossible-to-Execute Recovery**

Consider the original referent description "big blue box." Given the current state of the world, which is different from the one that the user perceived, the user actually meant a "medium blue box."

The agent's actual surroundings have the following objects:

- big brown box
- medium blue lamp
- big blue chair
- big blue door
- medium blue box **(target)**
- medium blue chair

The agent first processes the route instruction in the plan grounding model, and initiates a subdialog with the user (changing from Process Input state to Start Subdialog state). Next, the agent follows the steps for resolving the costs of relaxing or removing constraints of the original query "big blue box":

- Step 1: Rank relaxations by informativeness.

**Algorithm 4.1:** Resolving impossible-to-execute instructions by refashioning the original referent description.

**Input**: Initial referent description words and current environment context
**Output**: Refashioned referent description, if a match is found

1 Rank relaxations and removals by the informativeness, that is, how greatly they reduce the search space. Group adjustments by their informativeness scores.
2 Rank adjustments within groups by situated dimension versus basic object type.
3 Within groups, rank adjustments by situated dimension type. Given results from a data collection study [85], participants preferred using intrinsic most, followed by egocentric proximity, object proximity, and history. We define the costs as higher to remove or relax the situated dimensions that participants preferred. Thus the costs in this category, from highest to lowest are: removal or relaxation of history, object proximity, egocentric proximity, and intrinsic.
4 Rank by color versus shape. Given results from the crowdsourced data collection study referenced above, participants preferred using color over shape, so costs should be applied inversely: cost to remove or relax color is higher compared to shape.
5 Rank by whether each change is a relaxation or removal of a constraint. We define costs that remove constraints as higher than those that relax constraints.
6 Decide whether to list the top-ranked reformulation of the original referent description or present a list. The agent checks the adjustment possibilities in the top-ranked group. Then the agent determines whether the possibilties from each adjustment present different possible candidate referents to the user. If they do, the agent must decide whether to present the list or just the top-ranked one. Currently, this is done heuristically using the following guideline: if the list is less than $n$ (where $n = 5$), and all possibilities come from adjustments have the same informativeness, present them all. Otherwise, present the top $n$ possibilities from the highest ranked group.

**return** Refashioned referent description.

$cost(rm\_BIG) = 1$ object matches "BLUE BOX", 5 don't.
If confirmed, the referent is resolved, otherwise, one candidate is removed from possibilities.
$cost(rm\_BLUE) = 0$ objects match "BIG BOX", 6 don't.
If confirmed, the referent is not resolved, otherwise zero candi-

dates are removed from possibilities.
$cost(relax : BIG \rightarrow MEDIUM) = 1$ object matches to
"MEDIUM BLUE BOX", 5 don't.
If confirmed, the referent is resolved, otherwise, one candidate is removed from possibilities.
$cost(rm\_BOX) = 2$ object matches "BIG BLUE", 4 don't.
If confirmed, two possibilities remain, otherwise, two are removed.

Rank from best to worst:
$cost = 1 : rm\_BIG, relax : BIG \rightarrow MEDIUM$
$cost = 2 : rm\_BOX$
$cost = \infty : rm\_BLUE$

- Step 2: Rank adjustments by situated dimension vs. basic object type within group.
  $\rightarrow$ no effect, both constraints are dimensions in top group.
- Step 3: Rank adjustments by situated dimension within group.
  $\rightarrow$ no effect, both dimensions are intrinsic.
- Step 4: Rank adjustments by color vs. shape (color higher, shape lower).
  $\rightarrow$ no effect, both dimensions are shape.
- Step 5: Rank adjustments within top group by relaxation vs. removal. Higher cost to remove a constraint than relax one. Costs adjust as follows:
  $\rightarrow cost = 1 : relax : BIG \rightarrow MEDIUM, rm\_BIG$
  $cost = 2 : rm\_BOX$
  $cost = \infty : rm\_BLUE$
- Step 6: Decide whether to present a list or just the top-ranked constraint adjustment. Since both changes in the top-ranked group result in the same referent candidate, there is no value to applying both changes. **Final decision: Apply-Repair**$(relax : BIG \rightarrow MEDIUM)$

This decision is then propagated up the plan grounding model and presented as a confirmation question to the user: "I don't see a big blue box. Did you mean a medium-sized blue box?"

**Scenario 2: Impossible-to-Execute Recovery**

Consider the original referent description: "blue box on the right by the door". The actual intended referent the user meant was: "gray box on the agent's right by the door".

The agent's actual surroundings has the following objects:

- blue box on the left by the lamp
- gray box on the right by the door **(target)**
- purple box on the right by the chair
- brown box on the left by the lamp
- blue chair on the right by the lamp
- blue chair on the left by the door
- black lamp on the right by the door
- blue door on the left by the box
- brown door on the right by the box

After processing the input, the agent assigns costs to possible adjustments to the original referent description:

- Step 1: Rank relaxations by informativeness.

  $cost(rm\_BLUE) = 1$ object matches "BOX ON THE RIGHT BY THE DOOR", 8 don't.
  If confirmed, the referent is resolved, otherwise, one candidate is removed from possibilities.
  $cost(rm\_BOX) = 0$ objects match "BLUE $x$ ON THE RIGHT BY THE DOOR", 9 don't.
  If confirmed, the referent is not resolved, otherwise zero candidates are removed from possibilities.
  $cost(rm\_ONTHERIGHT) = 0$ objects match "BLUE BOX BY THE DOOR", 9 don't.
  If confirmed, the referent is not resolved, otherwise zero candidates are removed from possibilities.
  $cost(rm\_BYTHEDOOR) = 0$ objects match "BLUE BOX ON THE RIGHT", 9 don't.
  If confirmed, the referent is not resolved, otherwise zero candidates are removed from possibilities.

- Step 2-6: Skipped because only one relaxation or removal is possible. **Final decision: Apply-Repair**$(rm\_BLUE)$

This decision is then propagated up the plan grounding model and presented as a confirmation question to the user: "I don't see a blue box on the right by the door. Did you mean a gray box on the right by the door?"

**Scenario 3: Impossible-to-Execute Recovery**

Same environment as Scenario 2:

Instead the original referent description is: "blue box by the door" and the actual intended referent is "blue box by the lamp".

- blue box on the left by the lamp **(target)**
- gray box on the right by the door
- purple box on the right by the chair
- brown box on the left by the lamp
- blue chair on the right by the lamp
- blue chair on the left by the door
- black lamp on the right by the door
- blue door on the left by the box
- brown door on the right by the box
- Step 1: Rank relaxations by informativeness.

  $cost(rm\_BLUE) = 1$ object matches "BOX BY THE DOOR", 8 don't.
  If confirmed, the referent is resolved, otherwise, one candidate is removed from possibilities.
  $cost(rm\_BOX) = 2$ objects match "BLUE $x$ BY THE DOOR", 7 don't.
  If confirmed, two possibilities remain, otherwise two candidates are removed.
  $cost(relax : BYTHEDOOR \rightarrow$
  $NextNearestObject : BYTHELAMP) =$
  1 object matches "BLUE BOX BY THE LAMP", 8 don't.
  If confirmed, the referent is resolved, otherwise, one candidate is removed from possibilities.

$cost(rm\_BYTHEDOOR) = 1$ object matches "BLUE BOX",
8 don't.
If confirmed, the referent is resolved, otherwise, one candidate is removed from possibilities.

Rank from best to worst:
$cost = 1 : rm\_BLUE, rm\_BYTHEDOOR,$
$\quad relax : BYTHEDOOR \rightarrow BYTHELAMP$
$cost = 2 : rm\_BOX$

- Step 2: Within group, rank adjustments by situated dimension vs. basic object type.
  $\rightarrow$ no effect, all in group 1 are situated dimensions.
- Step 3: Rank by situated dimension within group. If we assume there is greater cost to remove more desirable properties based on the data collection study, then we rank object proximity adjustments as having lower cost than intrinsic property adjustments.

  Rank from best to worst:
  $cost = 1 : rm\_BYTHEDOOR, relax : BYTHEDOOR \rightarrow$
  $\quad BYTHELAMP,$
  $rm\_BLUE$
  $cost = 2 : rm\_BOX$
- Step 4: Rank by color vs. shape.
  $\rightarrow$ No effect.
- Step 5: Rank by relaxation vs. removal of constraint. If we assume relaxations have lower cost than constraint removal, the ranks are adjusted as follows:

  Rank from best to worst:
  $cost = 1 : relax : BYTHEDOOR \rightarrow BYTHELAMP,$
  $\quad rm\_BYTHEDOOR, rm\_BLUE$
  $cost = 2 : rm\_BOX$
- Step 6: Decide top-ranked adjustment vs. list. The agent's model now checks the value to presenting a list of alternative referent descriptions or only one in the top-ranked group:

  $relax : BYTHEDOOR \rightarrow BYTHELAMP \wedge$
  $\quad rm\_BYTHEDOOR \rightarrow$ resolve to same object,

while $rm\_BLUE \rightarrow$ resolves to a different object.

Since there are two possible objects in the list, and the list size is less than five, the agent will present a list of options to the user. **Final decision: Apply-Repair**$(relax : BYTHEDOOR \rightarrow BYTHELAMP \wedge rm\_BLUE)$.

The system will output the phrase: "I don't see a blue box by the door, but I did find a blue box on my left by the lamp and a gray box on my right by the door."

### 4.2.4 Question Presentation

The primary factor for presenting questions (i.e., *how to ask*) has to do with their length: The agent's model must reduce the amount of **effort** the user and agent together would spend on getting to a mutually grounded referent. This is achieved by listing options to the user. This action grounds what the agent is capable of understanding and what it knows about in the environment. The agent crafts questions such that users' answers are short.

### 4.2.5 Canceling the Task

The user must have the capability to decide whether to continue attempting or completely abandon a task. A rephrasing of the original instruction represents an attempt to issue the same task. Issuing a halt or cancel command can permit the agent to abandon the task, without any further subdialogue.

## 4.3 Chapter Summary

In this chapter we presented related work about grounding in dialogue systems, and the plan grounding model used in this thesis. Studies exist for both human-human dialogue and spoken dialogue systems, but those for the latter focus on miscommunication stemming from speech and language interpretation problems. We presented several models of grounding. Clark's Contribution Model [27], which accounts for the joint projects dialogue partners attempt to accomplish in dialogue, is a good fit for handling situated grounding problems. Next we discussed statistical approaches for dialogue control, which have focused on reinforcement learning methods. Those methods operate in domains where massive amounts of data could

be collected (e.g., telephony systems), and not in situated interaction scenarios. We also discussed symbol grounding, requesting clarification, and machine learning approaches for dialogue action selection.

We described a model of grounding that also includes planning and environment information. This *plan grounding model* attempts to repair problems that relate to plans that a user and agent are trying to accomplish. A successful user instruction resolves to an action with referents that the agent can uniquely identify. Whenever a situated grounding problem arises, the agent uses previous successful interactions to select recovery strategies. This can occur for both referential ambiguity and impossible recovery. Finally, we described an approach that expands how an agent can handle an impossible-to-execute command. In this approach, the agent uses a search-based constraint relaxation technique to identify alternative referents to one that was originally overspecified.

Chapter 5

# Miscommunication Recovery in Physically Situated Dialogue

We describe an empirical study that crowdsourced human-authored recovery strategies for various problems encountered in physically situated dialogue [85]. The purpose was to investigate the strategies that people use in response to requests that are referentially ambiguous or impossible to execute. Results suggest a general preference for including specific kinds of visual information when disambiguating referents, and for volunteering alternative plans when the original instruction was not possible to carry out.

## 5.1 Miscommunication in Physically Situated Domains

Physically situated dialogue differs from traditional human-computer dialogue in that interactions will make use of reference to a dialogue agent's surroundings. Tasks may fail due to dependencies on specific environment configurations, such as when a robot's path to a goal is blocked. People will often help; in navigation dialogues they tend to ask proactive, task-related questions instead of simply signaling communication failure [124]. They supplement the agent's representation of the environment and allow it to complete tasks. The current study establishes an empirical basis for grounding in physically situated contexts. We had people provide recovery strategies for a robot in various situations.

The focus of this work is on recovery from *situated grounding problems*, a

type of miscommunication that occurs when an agent fails to uniquely map a person's instructions to its surroundings [84]. A *referential ambiguity* is where an instruction resolves to more than one possibility (e.g., "Search the room on the left" when there are multiple rooms on the agent's left); an *impossible-to-execute* problem fails to resolve to any action (e.g., same instruction but there are no rooms on the agent's left). A common strategy evidenced in human-human corpora is for people to ask questions to recover from situated grounding problems [133].

Dialogue divides into two levels: that of managing the actual dialogue—determining who has the floor, that an utterance was recognized, etc.—and the dialogue that serves the main *joint activities* that dialogue partners are carrying out, like a human-robot team exploring a new area [6]. Most approaches to grounding in dialogue systems are managing the dialogue itself, making use of spoken language input as an indicator of understanding (e.g., [10, 126]). Situated grounding problems are associated with the main joint activities; to resolve them we believe that the recovery model must be extended to include planning and environment information. Flexible recovery strategies make this possible by enabling dialogue partners to coordinate their joint activities and accomplish tasks.

We cast the problem space as one where the agent aims to select the most efficient recovery strategy that would resolve a user's intended referent. We expect that this efficiency is tied to the cognitive load it takes to produce clarifications. Viethen and Dale [137] suggest a similar prediction in their study comparing human and automatically generated referring expressions of objects and their properties. We sought to answer the following questions in this work:

- How good are people at detecting situated grounding problems?
- How do people organize recovery strategies?
- When resolving ambiguity, which properties do people use to differentiate referents?
- When resolving impossible-to-execute instructions, do people use active or passive ways to get the conversation back on track?

We determined the most common recovery strategies for referential ambiguity and impossible-to-execute problems. Several patterns emerged that suggest ways that people expect agents to recover. Ultimately we intend for dialogue systems to use such strategies in physically situated contexts.

## 5.2 Existing Work

Researchers have long observed miscommunication and recovery in human-human dialogue corpora. The HCRC MapTask had a direction giver-direction follower pair navigate two dimensional schematics with slightly different maps [2]. Carletta [21] proposed several recovery strategies following an analysis of this corpus. The SCARE corpus collected human-human dialogues in a similar scenario where the direction follower was situated in a three-dimensional virtual environment [128].

The current study follows up an initial proposal set of recovery strategies for physically situated domains [82]. Others have also developed recovery strategies for situated dialogue. [66] developed a framework for a robot mapping an environment that employed conversational strategies as part of the grounding process. A similar study focused on resolving misunderstandings in the human-robot domain using the Wizard-of-Oz methodology [63]. A body of work on referring expression generation uses object attributes to generate descriptions of referents (e.g., [40, 43]). Viethen and Dale [137] compared human-authored referring expressions of objects to existing natural language generation algorithms and found them to have very different content.

Crowdsourcing has been shown to provide useful dialogue data: Manuvinakurike and DeVault [80] used the technique to collect game-playing conversations. [144] and [91] have used crowdsourced data for training, while others have used it in real time systems [56, 68].

## 5.3 Method

In this study, participants came up with phrases that a search-and-rescue robot should say in response to an operator's command. The participant's task was to view scenes in a virtual environment then formulate the robot's response to an operator's request. Participants listened to an operator's verbal command then typed in a response.

Scenes displayed one of three situations: *referential ambiguity* (more than one possible action), *impossible-to-execute* (zero possible actions), and *executable* (one possible action). The instructions showed some example problems. All situations involved one operator and one robot.

Figure 5.1: An example trial where the operator's command was *"Move to the table"*. In red is the robot (*centered*) pointed toward the back wall. Participants would listen to the operator's command and enter a response into a text box.

### 5.3.1 Experiment Design

After instructions and a practice trial, participants viewed scenes in one of 10 different environments (see Figure 5.1). They would first watch a fly-over video of the robot's environment, then view a screen showing labels for all possible referable objects in the scene. The participant would then watch the robot enter the first scene. The practice trial and instructions did not provide any examples of questions.

The robot would stop and a spoken instruction from the operator would be heard. The participant was free to replay the instruction multiple times. They would then enter a response (say an acknowledgment or a question). Upon completion of the trial, the robot would move to a different scene, where the process was repeated.

Only self-contained questions that would allow the operator to answer without follow-up were allowed. Thus generic questions like "which one?" would not allow the operator to give the robot enough useful information to proceed. In the instructions, we suggested that participants include some detail about the environment in their questions.

Participants used a web form[1] to view situations and provide responses. We

[1] See http://goo.gl/forms/ZGpK3L1nPh for an example.

| Trial Group | # Participants | # Ambiguous | # Impossible | # Executable |
|---|---|---|---|---|
| 1 | 15 | 9 | 9 | 7 |
| 2 | 15 | 16 | 6 | 3 |
| Total | 30 | 25 | 15 | 10 |

Table 5.1: Distribution of stimulus types across the two trial groups of participants. Trials either had referential ambiguity, were impossible-to-execute, or executable.

recorded demographic information (gender, age, native language, native country) and time on task. The instructions had several attention checks [101] to ensure that participants were focusing on the task.

We created fifty trials across ten environments. Each environment had five trials that represented waypoints the robot was to reach. Participants viewed five different environments (totaling twenty-five trials). Each command from the remote operator to the robot was a route instruction in the robot navigation domain. Trials were assembled in two groups and participants were assigned randomly to one (see Table 5.1). Trial order was randomized according to a Latin Square.

### Scenes and Environments

Scenes were of a 3D virtual environment at eye level, with the camera one to two meters behind the robot. Camera angle issues with environment objects caused this variation.

Participants understood that the fictional operator was not co-located with the robot. The USARSim robot simulation toolkit and the UnrealEd game map editor were used to create the environment. Cepstral's SwiftTalker was used for the operator voice.

Of the fifty scenes, twenty-five (50%) had referential ambiguities, fifteen (30%) were impossible-to-execute, and ten (20%) were executable controls. The selection was weighted to referential ambiguity, as these were expected to produce greater variety in recovery strategies. We randomly assigned each of fifty trials a stimulus type according to this distribution, then divided the list into ten environments. The environments featured objects and doorways appropriate to the trial type, as well as waypoints.

| Dimension | Property | # Scenes |
|---|---|---|
| Intrinsic<br>(i.e., "perceptual feature") | On no dimension does the target referent share an intrinsic property value with any other object of its type. The two intrinsic properties are color and size. | 20 |
| History<br>(i.e., "conceptual feature") | The robot already visited the referent once. | 14 |
| Object Proximity<br>(i.e., "functional relation") | The referent has a unique, nearby object that can serve as a "feature" for reference purposes. | 21 |
| Egocentric Proximity<br>(i.e., "spatial relation") | The referent has a unique spatial relationship relative to the robot. The relation is prototypical, generally falling along a supposed axis with the robot. | 20 |

Table 5.2: Ambiguous scene referent description space. The number of scenes was out of 25 total. We relate the current terms to general types defined by Carlson and Hill [23].

*Referential Ambiguity* We arranged the sources of information participants could use to describe referents, to enable analysis of the relationship between context and recovery strategies. The sources of information (i.e., "situated dimensions") were: (1) *intrinsic properties* (either color or size), (2) *history* (objects that the robot already encountered), (3) *egocentric proximity* of the robot to candidate referents around it (the robot's perspective is always taken), and (4) *object proximity* (proximity of candidate referents to other objects). Table 5.2 provides additional details.

Scenes with referential ambiguity had up to four sources of information available. Information sources were evenly distributed across five trial types: one that included all four sources, and four that included all but one source of information (e.g., one division excluded using history information but did allow proximity, spatial, and object properties, one excluded proximity, etc.).

*Impossible-to-Execute* The impossible-to-execute trials divided into two broad types. Nine of the fifteen scenes were impossible because the operator's command did not match to any referent in the environment. The other six scenes were impossible because a path to get to the matching referent was not possible.

*Executable* Ten scenes were executable for the study and served as controls. The operator's command mentioned existing, unambiguous referents.

**Robot Capabilities**

Participants were aware of the robot's capabilities before the start of the experiment. The instructions said that the robot knew the locations of all objects in the environment and whether doors were closed or open. The robot also knew the color and size of objects in the environment (*intrinsic properties*), where objects were relative to the robot itself and to other objects (*proximity*), when objects were right, left, in front, and behind it (*spatial terms*), the room and hallway locations of objects (*location*), and the places it has been (*history* – the robot kept track of which objects it had visited). The robot could not pass through closed doors.

### 5.3.2 Hypotheses

We made five hypotheses about the organization and content of participant responses to situated grounding problems:

- *Hypothesis 1:* Participants will have more difficulty detecting impossible-to-execute scenes than ambiguous ones. Determining a robot's tasks to be impossible requires good *situation awareness* [96] (i.e., an understanding of surroundings with respect to correctly completing tasks). Detecting referential ambiguity requires understanding the operator's command and visually inspecting the space [127]; detecting impossible commands also requires recalling the robot's capabilities and noticing obstacles. Previous research has noted that remote teleoperators have trouble establishing good situation awareness of a robot's surroundings [19, 25]. Moreover, obstacles near a robot can be difficult to detect with a restricted view as in the current study [1, 4].
- *Hypotheses 2a and 2b:* Responses will more commonly be single, self-contained questions instead of a scene description followed by a question (2a for scenes with referential ambiguity, 2b for scenes that were impossible-to-execute). This should reflect the principle of *least collaborative effort* [27], and follow from Carletta's [21] observations in a similar dataset.
- *Hypothesis 3:* Responses will use the situated dimensions that require the

least cognitive effort when disambiguating referents. Viethen and Dale [137] suggest that minimizing cognitive load for the speaker or listener would produce more human-like referring expressions. We predict that responses will mention visually salient features of the scene, such as color or size of referents, more than history or object proximity. [35] found that color and shape draw more attention than other properties in visual search tasks when they are highly distinguishable.

- *Hypothesis 4:* In cases of referential ambiguity where two candidate referents are present, responses will confirm one referent in the form of a yes-no question more than presenting a list. Results from an analysis of task-oriented dialogue suggests that people are efficient when asking clarification questions [110]. Additionally, Clark's *least collaborative effort* principle [27] suggests that clarifying one referent using a yes-no confirmation would require less effort than presenting a list in two ways: producing a shorter question and constraining the range of responses to expect.
- *Hypothesis 5:* For impossible-to-execute instructions, responses will most commonly be ways for the robot to proactively work with the operator's instruction, in an effort to get the conversation back on track. The other possible technique, to simply declare that the problem is not possible, will be less common. This is because participants will believe such a strategy will not align with the task goal of having the robot say something that will allow it to proceed with the task. Skantze found that in human-human navigation dialogues, people would prefer to look for alternative ways to proceed rather than simply express non-understanding [124].

### 5.3.3 Measures

The key independent variable in this study was the stimulus type that the participant viewed (i.e., referential ambiguity, impossible-to-execute, or executable). Dependent variables were observational measurements, presented below. We report Fleiss' kappa score for inter-annotator agreement between three native English speaking annotators on a subset of the data.

*Correctness* ($\kappa = 0.77$): Whether participants correctly determined the situation as ambiguous, impossible, or executable. Annotators labeled correctness based on the content of participant responses. This measure assessed participant accuracy for

detecting situated grounding problems. Either *correct* or *incorrect*.

*Sentence type* ($\kappa = 0.82$): Either *declarative*, *interrogative*, *imperative*, or *exclamatory* [30].

*Question type* ($\kappa = 0.92$): Sentences that needed an answer from the operator. The three types were *yes-no questions*, *alternative questions* (which presented a list of options and includes *wh-* questions that used sources from Table 5.2), and generic *wh- questions* [30].

*Situated dimensions in response* ($\kappa = 0.75$): The capability (or capabilities) that the participant mentioned when providing a response. The types were *intrinsic* (color or size), *object proximity*, *egocentric proximity*, and *history*.

*Projected belief* (impossible-to-execute trials only, $\kappa = 0.80$): The participant's belief about the next task, given the current operator instruction (projected onto the robot). The types were *unknown* (response indicates participant is unsure what to do next), *ask for more* (ask for more details), *propose alternative* (propose alternative object), *ask for help* (ask operator to physically manipulate environment), and *off topic*.

### 5.3.4 Participation

We recruited 30 participants. All participants completed the web form through the Amazon Mechanical Turk (MTurk) web portal[2], all were located in the United States and had a task approval rate $\geq$95%. The group included 29 self-reported native English speakers born in the United States; 1 self-reported as a native Bangla speaker born in Bangladesh. The gender distribution was 15 male to 15 female. Participants ranged in age from 22 to 52 (*mean:* 33 years, *std. dev.:* 7.7). They were paid between $1 and $2 for their participation. We collected a total of 750 responses.

[2]https://www.mturk.com

| Problem Type | Sample Crowdsourced Responses |
|---|---|
| Referential Ambiguity | ▹ *Do you mean the table in front of me?*<br>▹ *Should I go to the small or big table?* |
| Impossible-to-Execute | ▹ *There is not a lamp behind me. Would you like for me to go to the lamp in front of me?*<br>▹ *Do you mean the lamp in front of me?* |

Table 5.3: Participants composed recovery strategies in response to operator commands that were referentially ambiguous or impossible-to-execute.

## 5.4 Results

We analyzed the measures by tabulating frequencies for each possible value. Table 5.3 presents some example responses.

### 5.4.1 Correctness

In general, participants were good at detecting situated grounding problems. Out of 750 responses, 667 (89%) implied the correct scene type. We analyzed correctness across actual stimulus types (ambiguous, impossible-to-execute, executable) using a mixed-effects analysis of variance model[3], with participant included as a random effect and trial group as a fixed effect.

Hypothesis 1 predicted that participants will do better detecting scenes with referential ambiguity than those that were impossible-to-execute; the results support this hypothesis. Actual stimulus type had a significant main effect on correctness ($F[2, 58] = 12.3$, $p < 0.001$); trial group did not ($F[1, 28] = 0.1$, $p = 0.72$). Participants had significantly worse performance detecting impossible-to-execute scenes compared to ambiguous ones ($p < 0.001$; Tukey HSD test). In fact, they were four times worse; of the impossible-to-execute scenes, participants failed to detect that 22% (50/225) of them were impossible, compared to 5% (17/375) of scenes with referential ambiguity. Of the 150 instructions that were executable, participants failed to detect 11% (16/150) of them as such.

[3]This approach computed standard least squares regression using reduced maximum likelihood [49].

### 5.4.2 Referential Ambiguity

We analyzed the 358 responses where participants correctly detected referential ambiguity.

Hypothesis 2a predicted that participants would more commonly ask single, self-contained questions instead of describing the scene and asking a question. We assessed this by counting sentence types within a response. Responses that had both a declarative sentence and an interrogative would fit this case. The results confirmed this hypothesis. Only 4.5% (16/358) of possible responses had a declarative and an interrogative.

Hypothesis 3 predicted that participants would use the situated dimensions that require the least cognitive effort when disambiguating referents. More specifically, the most common mentions will be those that are visually apparent (intrinsic properties like color and size), while those that require more processing would have fewer mentions (history and to a lesser extent object proximity and egocentric proximity). We measured this by tabulating mentions of situated dimensions in all 358 correct participant responses, summarized in Figure 5.2. Multiple dimensions could occur in a single response. The results support this hypothesis. By far, across all ambiguous scenarios, the most mentioned dimension was an intrinsic property. More than half of all situated dimensions used were intrinsic (59%, 242/410 total mentions). This was followed by the dimensions that we hypothesize require more cognitive effort: egocentric proximity had 30% (125/410) of mentions, object proximity 10% (39/410), and history 1% (4/410). Of the intrinsic dimensions mentioned, most were only color (61%, 148/242), followed by size (33%, 81/242), and using both (5%, 13/242).

Hypothesis 4 predicted that participants would ask yes-no confirmation questions in favor of presenting lists when disambiguating a referent with exactly two candidates. The results suggest that the opposite is true; people strongly preferred to list options, even when a confirmation question about one would have been sufficient. Of the 285 responses that were correctly detected as ambiguous and were for scenes of exactly two possible referents, 74% (212/285) presented a list of options. Only 14% (39/285) asked yes-no confirmation questions. The remaining 34 questions (12%) were generic wh-questions. These results held in scenes where three options were present. Overall 72% (259/358) presented a list of options, while 16% (58/358) asked generic wh-questions and 11% (41/358) asked yes-no confirmations.

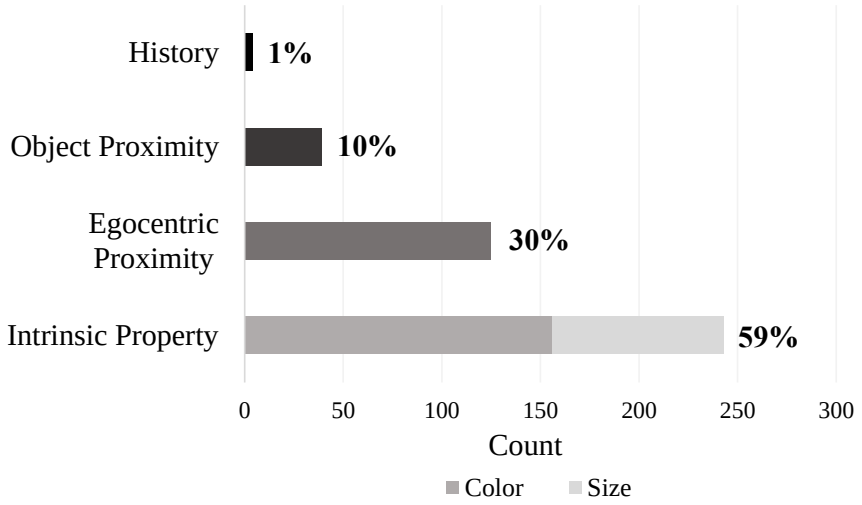



Figure 5.2: Counts of situated dimensions in recovery strategies for scenarios with referential ambiguity.

### 5.4.3 Impossible-to-Execute

We analyzed the 175 responses where participants correctly identified impossible-to-execute situations.

Hypothesis 2b predicted that participants would more often only ask a question than also describe the scene. Results confirmed this hypothesis. 42% (73/175) of responses simply asked a question, while 22% (39/175) used only a declarative. More than a third included a declarative as well (36%, 63/175). The general organization to these was to declare the problem then ask a question about it (89%, 56/63).

Hypothesis 5 predicted that responses for impossible-to-execute instructions will more commonly be proactive and make suggestions, instead of simply declaring that an action was not possible. Table 5.4 summarizes the results, which confirmed this hypothesis. The most common belief that participants had for the robot was to have it propose an alternative referent to the impossible one specified by the operator. The next-most common was to have the robot simply express uncertainty about what to do next. Though this belief occurred in about a third of responses, the remaining responses were all proactive ways for the robot to get the conversation back on track (i.e., propose alternative, ask for more, and ask for help).

| Projected Belief | Count | Percentage |
|---|---|---|
| Propose Alternative | 72 | 41% |
| Unknown | 56 | 32% |
| Ask for More | 42 | 24% |
| Ask for Help | 5 | 3% |
| Total | 175 | 100% |

Table 5.4: Projected belief annotations for the 175 correct detections of impossible-to-execute scenes.

## 5.5 Discussion

The results largely support the hypotheses, with the exception of Hypothesis 4. They also provide information about how people expect robots to recover from situated grounding problems.

*Correctness* Participants had the most trouble detecting impossible-to-execute scenes, supporting Hypothesis 1. An error analysis of the 50 responses for this condition had participants responding as if the impossible scenes were possible (62%, 31/50). The lack of good situation awareness was a factor, which agrees with previous findings in the human-robot interaction literature [19, 25]. We found that participants had trouble with a specific scene where they confused the front and back of the robot (9 of the 31 impossible-executable responses were for this scene). Note that all scenes showed the robot entering the room with the same perspective, facing forward.

*Referential Ambiguity* Results for Hypothesis 2a showed that participants overwhelmingly asked only a single, self-contained question as opposed to first stating that there was an ambiguity. Participants also preferred to present a list of options, despite the number of possible candidates. This contradicted Hypothesis 4. [110] found that in task-oriented human-human dialogues, clarification requests aim to be as efficient as possible; they are mostly partially formed. The results in our study were not of real-time dialogue; we isolated specific parts of what participants

believed to be human-computer dialogue. Moreover, Rieser and Moore were observing clarifications at Bangerter and Clark's [6] dialogue management level; we were observing them in service of the joint activity of navigating the robot. We believe that this difference resulted in participants using caution by disambiguating with lists.

These results suggest that dialogue systems should present detection of referential ambiguity implicitly, and as a list. Generic *wh-* questions (e.g., "which one?" without presenting a follow-on list) are less desirable because they don't constrain what the user can say, and don't provide any indication of what the dialogue system can understand. A list offers several benefits: it grounds awareness of surroundings, presents a fixed set of options to the user, and constrains the range of linguistic responses. This could also extend to general ambiguity, as in when there are a list of matches to a query, but that is outside the scope of this work. Lists may be less useful as they grow in size; in our study they could not grow beyond three candidates.

The data also supported Hypothesis 3. Participants generally preferred to use situated dimensions that required less effort to describe. Intrinsic dimensions (color and size) had the greatest count, followed by egocentric proximity, object proximity, and finally using history. We attribute these results to the salient nature of intrinsic properties compared to ones that must be computed (i.e., egocentric and object proximity require spatial processing, while history requires thinking about previous exchanges). This also speaks to a similar claim by [137]. Responses included color more than any other property, suggesting that an object's color draws more visual attention than its size. Bright colors and big shapes stand out most in visual search tasks; we had more of the former than the latter [35].

For an ambiguous scene, participants appear to traverse a *salience hierarchy* [51] whereby they select the most visually salient feature that also uniquely teases apart candidates. While the salience hierarchy varies depending on the current context of a referent, we anticipate such a hierarchy can be defined computationally. Others have proposed similar processes for referring expression generation [43, 136]. One way to rank salience on the hierarchy could be predicted mental load; we speculate that this is a reason why history was barely mentioned to disambiguate. Another would be to model visual attention, which could explain why color was so dominant.

Note that only a few dimensions were "competing" at any given time, and their presence in the scenes was equal (save for history, which had slightly fewer due

to task design constraints). Egocentric proximity, which uses spatial language to orient candidate referents relative to the robot, had a moderate presence. When intrinsic properties were unavailable in the scene, responses most often used this property. We found that sometimes participants would derive this property even if it wasn't made prototypical in the scene (e.g., referring to a table as "left" when it was in front and off to the left side of the robot). This suggests that using egocentric proximity to disambiguate makes a good fallback strategy when nothing else works. Another situated dimension emerged from the responses, disambiguation by location (e.g., "Do you mean the box in this room or the other one?"). Though not frequent, it provides another useful technique to disambiguate when visually salient properties are not available.

Our findings differ from those of Carlson and Hill [23] who found that salience is not as prominent as spatial relationships between a target (in the current study, this would be the robot) and other objects. Our study did not direct participants to formulate spatial descriptions; they were free to compose responses. In addition, our work directly compares intrinsic properties for objects of the same broad type (e.g., disambiguation of a doors of different colors). Our findings suggest the opposite of [93], who found that when pointing out an object, describing its position may be better than describing its attributes in human-robot interactions. Their study only had one object type (cube) and did not vary color, size, or proximity to nearby objects. As a result, participants described objects using spatial terms. In our study, we explored variation of several attributes to determine participants' preferences.

*Impossible-to-Execute* Results supported Hypothesis 2b. Most responses had a single sentence type. Although unanticipated, a useful strategy emerged: describe the problem that makes the scene impossible, then propose an alternative referent. This type of strategy helped support Hypothesis 5. Responses for impossible scenes largely had the participant proactively presenting a way to move the task forward, similar to what [124] observed in human-human dialogues. This suggests that participants believed the robot should ask directed questions to recover. These questions often took the form of posing alternative options.

### 5.5.1 Limitations

We used the Amazon Mechanical Turk web portal to gather responses in this study. As such we could not control the participant environment when taking the study, but we did include attention checks. Participants did not interact with a dialogue system. Instead we isolated parts of the interaction that were instances of where the robot would have to say something in response to an instruction. We asked participants to provide what they think the robot should say; there was no ongoing interaction. However, we maintained continuity by presenting videos of the robot navigating through the environment as participants completed the task. The robot was represented in a virtual environment, which prevents us from understanding if there are any influencing factors that may impact results if the robot were in physical form or co-present with the participant.

Participants' prior notions of robot intelligence could have influenced the structure of their recovery strategies. If a participant thought highly of robots, they may have included more human-like conversational strategies (e.g., *the red or blue one?*), while others with a lower notion of robot intelligence could have used more rigid responses (e.g., *did you mean the red box or the blue box?*). We did not survey participants about their prior experience with robots, so we could not analyze the results along this dimension. However, we expect that this variable is minimized by the listing of the robot's capabilities in the task instructions; participants knew the types of information the robot could handle.

## 5.6 Chapter Summary

Recovery strategies allow situated agents like robots to recover from misunderstandings by using the human dialogue partner. In this chapter, we conducted a study that collected recovery strategies for physically situated dialogue with the goal of establishing an empirical basis for grounding in physically situated contexts. We crowdsourced 750 written strategies across 30 participants and analyzed their situated properties and how they were organized.

We found that participants' recovery strategies minimize cognitive effort and indicate a desire to successfully complete the task. For disambiguation, there was a preference for strategies that use visually salient properties over ones that require additional mental processing, like spatial reasoning or memory recall. For impossible-to-execute scenes, responses more often presented alternative referents

than just noting non-understanding. We should note that some differences between our findings and those of others may in part rest on differences in task and environment, though intrinsic variables such as mental effort will likely persist over different situations.

In this thesis, we use these data to model salience ranking in similar contexts. We assess the hypothesis that participants' preferences from this study will enhance performance in a spoken dialogue system that deploys similar strategies.

Chapter 6

# Detecting Situated Grounding Problems with TeamTalk

This chapter describes the infrastructure and representation that permits a spoken dialogue system to detect miscommunication in physically situated contexts. First, we scope the problem space for this thesis by defining a navigation domain ontology. We describe the objects, actions, paths, and other information relevant to the navigation task. Next, we describe TeamTalk, the research platform used in this thesis. Finally, we describe our approach to detecting situated grounding problems with the Situated Reasoner, a software extension to TeamTalk.

## 6.1 Virtual Agent Navigation Domain

There are a vast amount of tasks where agents can interact with the physical world [11]. In this work, we define the core capabilities for an agent operating in the navigation domain. Constraining the problem domain helps focus our efforts on where situated grounding problems occur and what the agent can do to repair these problems. Ultimately this allows us to identify opportunities for the agent to learn and improve miscommunication recovery.

Below we define the problem domain for this thesis, virtual agent navigation (the virtual agent is rendered on-screen as a robot), and the core primitives in the environment that the agent should be able to access in order to detect and recover from situated grounding problems. Primitives allow the agent to have fundamental knowledge for accomplishing tasks in the domain, but at the same time allow us

to investigate research questions related to this thesis. Defining primitives also allows us to design controlled experimental evaluations because we can constrain the task to the agent's capabilities. Starting the agent without prior knowledge is beyond the scope of this thesis; some *a priori* knowledge is assumed so that we may evaluate the dynamics of human-robot dialogue.

The virtual agent navigation domain involves two dialogue partners, a person (the direction giver), and a virtual agent (a direction follower). The direction giver-follower paradigm in dialogue has seen a wide adoption, originating with the HCRC Map Task Corpus [2] and continuing with other human-human and human-computer dialogue corpora (e.g., [83, 89, 128]). The direction giver assigns the follower navigation tasks in the environment by communicating about space, referents, and distance as route instructions. The nature of the task prohibits the direction giver from accomplishing tasks without the direction follower. Those tasks become the goals the direction follower must accomplish to achieve mutual task success. The direction follower performs the giver's requested tasks, asking for clarification or help as needed.

The agent represents its surroundings by maintaining static and dynamic knowledge about the environment. Static information is stored in a knowledge base with a structured ontology. The agent can retrieve information like the location of an object or its color by issuing a query to the knowledge base. We developed the ontology with the Protégé Ontology Editor [60]. Dynamic information is accessed in real time using the agent's current position and spatial orientation. This information is directly accessible to the agent at any time.

Primitives in the virtual agent navigation domain are divided into four primary categories: (1) objects and their properties, (2) actions and their parameters, (3) paths, and (4) existential knowledge (i.e., factual knowledge about the environment or relevant to the dialogue). These primitives were expanded using navigation dialogues from the SCARE corpus (three dialogues, with a total of 780 dialogue turns). The process included checking the feasibility of storing information from these instructions against the primitives below, and expanding them as necessary. From these categories, we defined an initial set of core heuristics about the environment that the dialogue agent should know *a priori*.

### 6.1.1 Object

An object in this domain is any referable object that can be named (see Table 6.1).

### 6.1.2 Action

An action is a task the agent can perform in the navigation domain. Below each action are the various parameters for that action (see Table 6.2).

### 6.1.3 Path

Paths is an action complete with parameters, where those parameters may contain objects (see Table 6.3).

### 6.1.4 Existential Knowledge

These are knowledge elements that are not otherwise defined in Objects, Actions, or Paths (see Table 6.4).

### 6.1.5 Core Heuristics

The core heuristics in Table 6.5 represent the initial heuristics that govern the agent's actions in the environment. We associate object properties with paths and actions.

## 6.2 Research Platform: TeamTalk with USARSim

In this section we describe the architecture of the TeamTalk spoken dialogue framework. To enable communication with virtual agents we used the USARSim simulator, which we also describe.

### 6.2.1 TeamTalk Human-Robot Dialogue Framework

TeamTalk[1] is a framework for spoken dialogue interaction in human-robot teams situated in real-world and virtual-world spaces [48, 86]. It creates a spoken language interface between one human operator and a team of one or more robots. The framework encompasses over a decade of research and development geared towards human-robot dialogue. The virtual environment and agents (presented as robots) are rendered in a 3D virtual environment, with 3D models for robots and

[1] http://wiki.speech.cs.cmu.edu/teamtalk

| Object (parameter) | Description |
|---|---|
| `Name` | The uniquely defining variable name of the object (e.g., BOX_1). |
| `Object Type` | The type of object (e.g., BOX). |
| `Location(x,y)` | The reachable $(x, y)$ location for the object. |
| `Center(x,y)` | The center $(x, y)$ of the object. |
| `Front(x,y)` | For objects that can be passed through (e.g., doorways), an $(x, y)$ location representing a move through it. |
| `Back(x,y)` | Alternate side of `Front(x,y)`. |
| `Area` | The space (room or hallway) that the object occupies. |
| `Intrinsic properties` | Observable attributes associated with the object. |
| → `Color` | Color of the object. |
| → `Size` | Size of the object in cubic centimeters. |
| → `Shape` | One or more shapes associated with the object. |
| → `Openness` | Object openness state (OPEN/CLOSED). |
| `Boolean properties` | TRUE or FALSE properties associated with the object. |
| → `canBePassedThrough` | Whether or not the object can be passed through. |
| → `isVisible` | Whether or not the object has been seen by the agent since the start of the current dialogue. |
| `Spatial relations` | Spatial relations between the object and environment. |
| → `Nearest object` | The nearest object to this one. |

Table 6.1: Objects and their static properties in the virtual agent navigation domain ontology. Arrows (→) indicate subtypes of a higher-level type (e.g., `Size` is a subtype of `Intrinsic properties`).

other objects. This framework uses the Olympus/RavenClaw architecture for task-oriented spoken dialogue. As part of this thesis, we extended several capabilities to TeamTalk that enable execution of a wider range of physically situated tasks and improved detection of situated grounding problems. The framework has been used for task-oriented dialogue in the search-and-rescue and navigation domains. Commands can be issued to an individual robot or a team of robots.

The TeamTalk virtual platform features two views of the interaction, a live map and a 3D view of the agent's environment. The TeamTalk GUI displays the agent's current location and the areas it has traversed, as shown in Figure 6.1. The map information is displayed as a two-dimensional occupancy grid that continually updates as the virtual agent navigates its surroundings. Multiple configured systems

| Action (parameter) | Description |
|---|---|
| `Go/walk/move-to` | Request to move to a location. |
| → `(object)` | A move to an object. |
| → `(object, intrinsic property)` | Object with a property intrinsic to that object. Either color or size. |
| → `(object, spatial relation)` | e.g., in front, behind, left, right |
| → `(object, nearby object)` | Refer to an object by its proximity to other objects. |
| → `(object, combination)` | Refer to an object by some combination of intrinsic property, spatial relation, or nearby object. |
| → `(direction, distance)` | A move either forward or backward a specified number of meters. |
| `Go-through/move-through` | Request to move through a referable space, like a doorway. |
| → `(object)` | Only objects that can be passed through qualify. |
| → `(object, intrinsic property)` | An object with an intrinsic property. |
| → `(object, spatial relation)` | e.g., in front, behind, left, right |
| → `(object, nearby object)` | Refer to an object by its proximity to other objects. |
| `Turn/rotate/take` | General rotation action. |
| → `(spatial relation)` | e.g., around, left, right |
| → `(spatial relation, degrees)` | Rotate a specified number of degrees. |
| `Stop/halt/wait` | General stop. |

Table 6.2: Actions in the virtual agent navigation domain ontology. Arrows (→) indicate different parameters of the action.

on the same network can view the virtual environment and agents. Figure 6.2 presents a detailed view of one such robot in a virtual environment (a virtual version of the Pioneer P2-DX robot). The Appendix features several new virtual environments created for this thesis.

TeamTalk's robot communication layer supports moving and turning commands for robots around a virtual space. The framework translates spoken language into a combination of the low-level moves and turns necessary to accomplish a navigation task. Messages containing low-level behaviors are passed between TeamTalk

| Path | Description |
|---|---|
| `Boolean` | TRUE/FALSE properties of paths. |
| → `isExecutable` | There is only one possible path. |
| → `isAmbiguous` | One of multiple paths to the goal. |
| → `isImpossible` | There is no way to complete the path. |
| → `isComplete` | Produces a plan from the start $(x, y)$ to the goal $(x, y)$. |
| → `isNotPassable` | Part of the path is blocked. |
| → `isNotFound` | A referent mentioned as part of the path does not exist. |

Table 6.3: Paths in the virtual agent navigation domain ontology. Arrows (→) indicate different properties of the path.

| Existential Value (parameter) | Description |
|---|---|
| `Exists(object[number` $n$`])` | There exist $n$ objects of some basic type (e.g., door) in the agent's surroundings. |
| `Exists(object[property` $p$`])` | There exists an object with property $p$. |
| `Exists(object,`$(x, y)$`)` | There exists an object at $(x, y)$. |

Table 6.4: Existential knowledge in the virtual agent navigation domain ontology.

and agents with communication libraries developed by the Boeing Company and Carnegie Mellon.

The current system supports interactions for robot navigation. Commands to TeamTalk include moves of specific distances (e.g., "Move forward five meters") and moves to defined objects in the agent's ontology (e.g., "Go to the box"). Details about objects are also possible as part of a navigation command (e.g., "Move to the white table in front of you"). Users can also specify turns (e.g., "Turn right"). TeamTalk also supports naming locations (e.g., "Name this location Alpha") and querying the whereabouts of agents in the environment.

Since TeamTalk uses RavenClaw for dialogue management, there are built-in capabilities to request missing information to complete a detected command (e.g., the concept `destination` for a `GoTo` command). If a user asks the agent to move forward but does not indicate a distance, RavenClaw will prompt for the distance. Like all RavenClaw-based systems, TeamTalk requires a task tree definition to specify domain-specific aspects of dialogue management. We describe the task

| Rule | Description |
|---|---|
| `PathNotPassable: ObjectNotPassable` | If an object in the current path is not passable, then the current path is not passable. |
| `PathIsImpossible: UnknownObject` | If an object of some basic object type is mentioned, and there exists no object of that type, then the path is impossible. |
| `PathIsImpossible: UnresolvedProperty` | If an object is mentioned with a property, and there exists no object with that property, then the path is impossible. |
| `PathIsImpossible: SpatialRelation` | If an object is mentioned with a spatial relation that does not exist in surroundings, then the path is impossible. |
| `PathIsAmbiguous: MultiplePaths` | If there are multiple paths generated for the instruction, then the path is ambiguous. |

Table 6.5: Core heuristics in the virtual agent navigation domain ontology.

tree definition developed for this thesis in Section 6.2.3.

TeamTalk stores environment information in a knowledge base with a structured ontology [112]. Pappu and Rudnicky developed an instance of the OWL representation of the Protégé [60] ontology for the TeamTalk system. It allows dialogue agents to store past actions and the locations of objects in the agent's environment [103, 104]. Past interactions are stored as *episodes* that represent a time-aligned sequence of dialogue commands and corresponding actions taken by the agent. This thesis extended the ontology to include a structure for representing physical situations and a history logging component for commands.

TeamTalk represents the agent's environment in the TeamTalk ontology as a semantic map. The map contains instances of objects from the ontology. Table 6.1 shows object properties in the ontology. We use the presence of objects in the agent's semantic map to detect situated grounding problems. For instance, the system tabulates the doors in the surroundings when an action referring to one is mentioned. The locations of these objects are used to calculate spatial relations and proximity to the agent. The agent also refers to objects in its semantic map when executing recovery strategies for situated grounding problems (e.g., accessing the current position of the agent and the spatial orientation of several matches to clarify

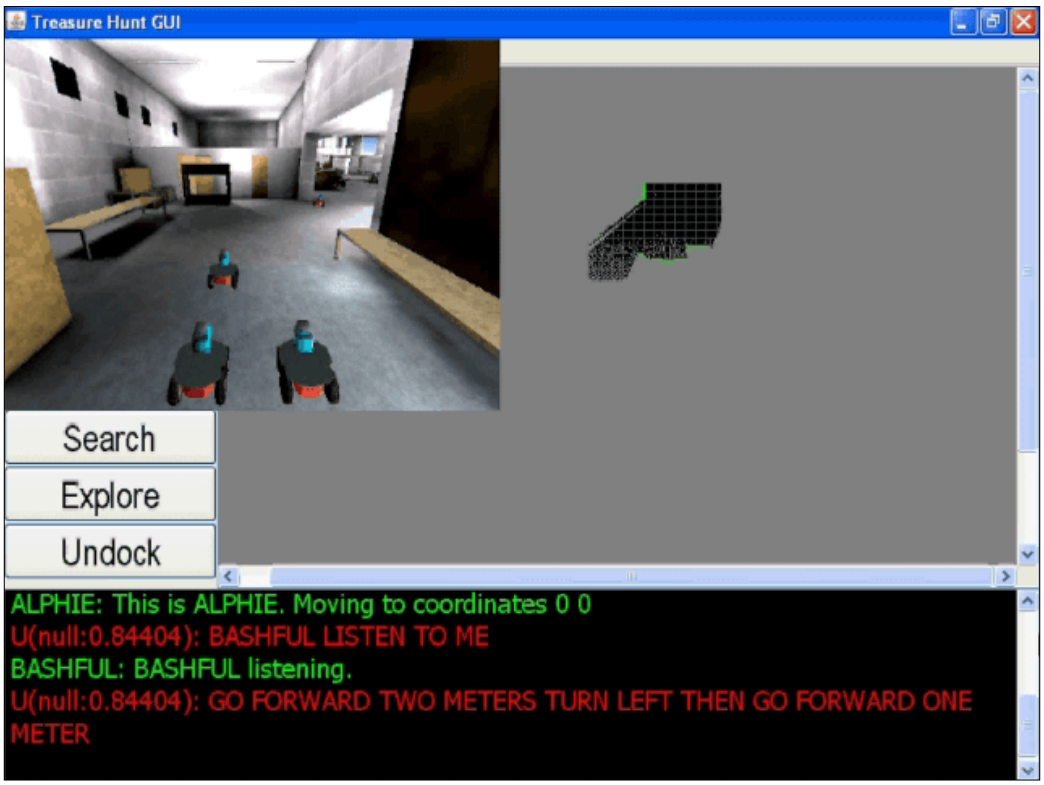



Figure 6.1: The TeamTalk GUI displaying a 3D virtual environment with a team of robots (left) and map (right). The dialogue history is displayed in the bottom of the window.

which referent the user intended from a list). Objects and their static properties were populated to serve as *a priori* knowledge for the agent, but their visibility was initially hidden.

### 6.2.2 USARSim and MOAST

TeamTalk supports spoken dialogue with virtual agents with the USARSim simulation software [24]. USARSim[2] (Unified System for Automation and Robot Simulation) is a high fidelity simulator designed for conducting research with mobile robots. It supports the rendering of virtual environments and robots with the Unreal Tournament 3D game engine. USARSim is actively supported by the robotics community, as it serves as the test platform for two robotics competitions (RoboCup Rescue Virtual Robot Competition and the IEEE Virtual Manufacturing

[2]http://sourceforge.net/projects/usarsim

Figure 6.2: A USARSim-based robot situated in a virtual environment. The TeamTalk system creates a conversational interface to this robot.

Automation Competition).

The Unreal Tournament game engine and USARSim simulator have been used to build stimuli for the corpus collection study described in Section 3.1. The purpose was to collect spoken language in the robot navigation domain.

In TeamTalk's virtual configuration, robots are simulated using the MOAST "SIMware" software [5]. MOAST is tightly integrated with USARSim, as it serves as a proxy for robotics middleware in the current infrastructure. MOAST supports the use of a path planner and simulated SLAM (Simultaneous Localization and Mapping). In this thesis, the virtual agent has an $A*$ path planner that references locally generated SLAM information. A MOAST robot enters a virtual environment by 'spawning' as a process, joining the virtual environment, and establishing a network connection to TeamTalk. The agent makes no assumptions about its occupancy grid as it traverses an environment. It must build one from scratch once it enters an environment.

### 6.2.3 TeamTalk Software Architecture

Figure 6.3 presents an overview of the TeamTalk architecture. TeamTalk features a pipeline architecture that incorporates a layer responsible for maintaining a dialogue with the user (human interaction layer) and a layer responsible for robot control (robotics layer) [86]. Figure 6.4 presents these layers. We now describe the two layers in detail.

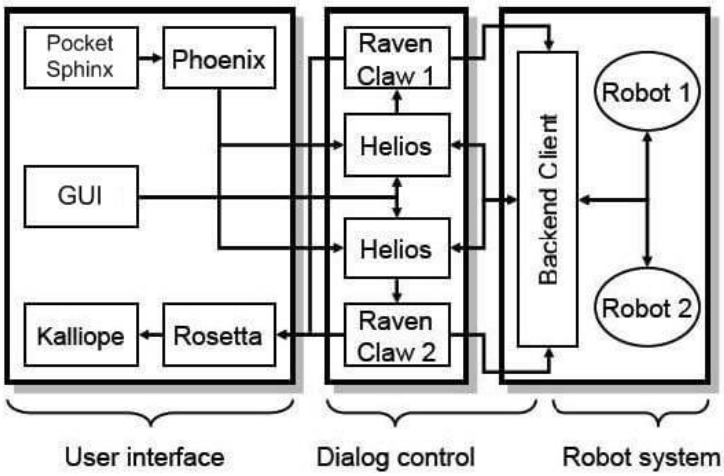



Figure 6.3: Overview of the TeamTalk architecture [86].

### Human Interaction Layer

The front end for TeamTalk handles user input and displays the robots' map. Optionally, a separate window can display the 3D rendering of robots in the virtual environment. The TeamTalk GUI shows an occupancy grid of the agent's environment; this expands over time as the agent sends back mapping updates. The GUI also displays the agent location and the recent conversation history (5 lines of dialogue). The dialogue interface starts once the operator presses the *Start Session* button on the GUI. Commands may either be spoken using a headset microphone or by typing directly into the GUI. A speech recognizer will interpret spoken commands and pass them along for processing. Responses from the agent are synthesized speech.

### Infrastructure Configuration: Human Interaction

The human interface component is as an instance of the Olympus Spoken Dialogue Framework [15]. Olympus packages all the software components necessary for a task-based dialogue system. Among the components are those to keep track of the conversation state, record and decode speech with Carnegie Mellon's PocketSphinx speech recognition engine [57], annotate speech for confidence, parse the input to extract its semantics, generate language, and produce synthesized speech. As part of the Olympus Spoken Dialogue Framework, all communication between components is accomplished with the Galaxy Communicator architecture [8].

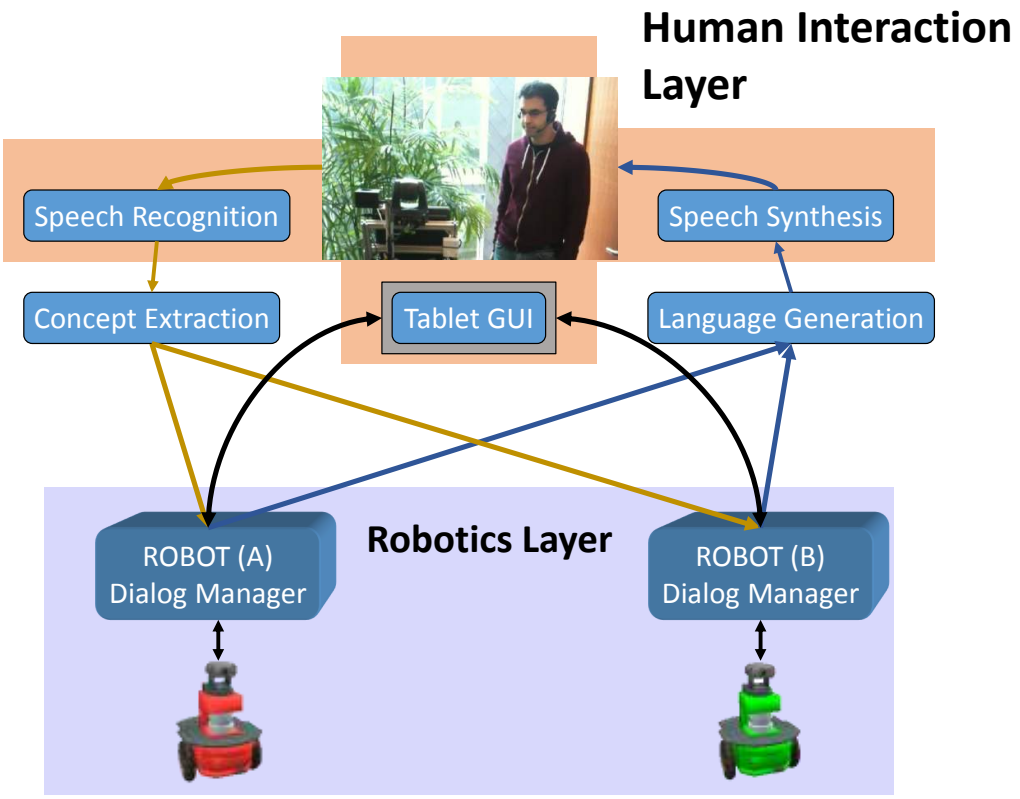



Figure 6.4: Layers of interaction in TeamTalk.

Olympus uses several components to interpret spoken language. An audio server detects voice activity, endpoints when to begin and end a speech recording, sends recordings to the PocketSphinx decoder, and collects recognition results. The Logios toolkit automatically creates a dictionary and language model based on the input grammar specified by the developer [146]. The Phoenix parser extracts semantics from decoded speech using a context-free grammar.

The RavenClaw dialogue management framework internalizes conversation state processes for each agent (i.e., real or simulated robot) [14]. The dialogue manager models the state of the conversation and tracks the next task the agent should perform, asking for clarification when needed. For robot navigation, we defined a set of *Dialogue Agencies*, or tasks, that an agent can perform. All agent (i.e., robot) capabilities are associated with dialogue agencies. Dialogue agencies are organized into a *dialogue task tree* that hierarchically represents possible agent actions as tasks and subtasks.

Every task an agent can do has certain requirements and prerequisites. For

example, to move a specified distance, the agent needs to also know the heading. These requirements are part of the dialogue task. The building blocks of the dialogue task are called *fundamental dialogue agents*, they are a specific type of handler for dialogue. A *dialogue task* is specified at implementation time as an ordered list of dialogue agents, dispatching inputs to appropriate dialogue agencies as the dialogue progresses.

There are four types of fundamental dialogue agents in RavenClaw. A *Request agent* asks and listens for certain user inputs that are relevant for the current task. An *Inform agent* generates a prompt that either greets the user or responds to input. *Expect agents* define the possible natural language bindings to *concepts*, or the variables RavenClaw manages during the conversation. An example concept could be the object in a movement command. Finally, *Execute agents* send robot navigation commands, composed of moves and turns, to the *domain reasoner*. The domain reasoner is part of the Robotics layer of the TeamTalk framework. Concepts can either be basic types like strings and integers, or complex types like arrays, frames and structs.

We developed handlers in the dialogue task tree so that TeamTalk could handle incomplete and noisy input for physically situated tasks. The `Handle Missing Action` agency handles spoken language that contains an object, but no action. The agency confirms whether or not the user intended to move to the object recognized; this enables the agent to recover from incomplete speech inputs. Conversely, the Request agents in the tree automatically prompt for missing objects if an action is specified. The `Move` agency handles moves that mention an object; in cases where the object is missing, the agency prompts for an object. The `Ask Disambiguate` agency presents a domain-specific list of objects in cases of referential ambiguity. Impossible-to-execute situations where a list of alternatives gets generated uses the same agency.

Syntactic bounds are placed on what words the speech recognizer's language model can interpret. The scope of user input is further restricted to certain semantic concepts using grammar mappings. Grammar mappings bind user input to concepts. For example, if a user tells the agent to move to the *box*, RavenClaw's grammar mapping binds user input to the semantic concept *landmark*. The recognizer vocabulary must have an entry for the word *box* for this to be possible.

TeamTalk's domain reasoner communicates with the agent. When the dialogue manager decides on the course of action that an agent should take, TeamTalk passes that message to the domain reasoner. This is usually from user-generated task

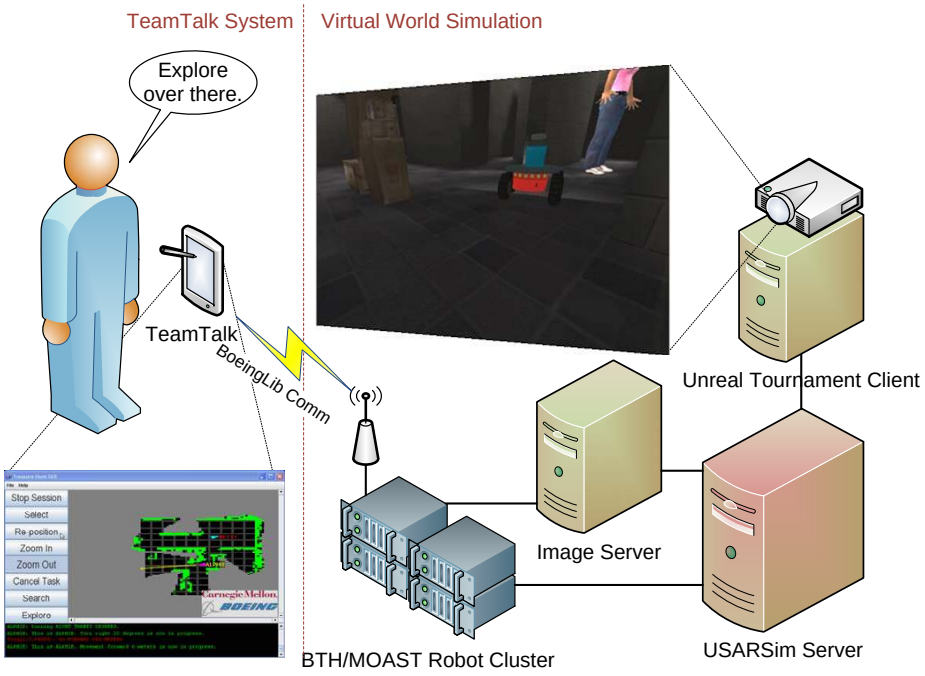



Figure 6.5: Overview of the TeamTalk and USARSim system setup [86].

instructions or responses to clarification.

During any stage of the dialogue, the agent may need to speak to the user. Rosetta, the natural language generation component of Olympus, produces natural language text given current dialogue context obtained from RavenClaw. Generation is domain-specific and specified as a series of templates with variables. Kalliope, the text-to-speech controller, converts the text to speech.

### Robotics Layer

Once an agent's dialogue manager determines a relevant task request, it will send a message to the domain reasoner. The domain reasoner features the communication protocols necessary to send move and rotation commands to the agent hardware (real or virtual).

### Infrastructure Configuration: Robotics

Figure 6.5 shows an overview of USARSim integration with TeamTalk. The TeamTalk front end runs on a machine with speech understanding hardware. The USARSim server hosts the environment and all participants; this includes each

agents and human user that participates. Viewing the virtual environment requires a client running the Unreal Tournament game engine software. Simulated agents run as individual processes on one or more Unix-based machines. In this thesis, we had one machine capable of running every part of the simulation architecture; this was possible because the Windows native machine hosted a Unix-based virtual machine (Ubuntu).

MOAST is configured to launch a user-configurable number of agents in the virtual environment. To observe the scene, an Unreal Tournament client must connect to the USARSim server. Commands from TeamTalk are sent to MOAST so that agents can accomplish the task in the virtual environment. In this work, all communications are between a human operator and a single agent.

## 6.3 Identifying Miscommunication in Physically Situated Dialogue

In the previous section, we described TeamTalk, the spoken dialogue framework used in this thesis. We now describe the sources of miscommunication and our approach to detecting them. This is the first step to handling miscommunication in physically situated dialogue; the second is to initiate recovery strategies, which we describe in Chapter 7.

### 6.3.1 Levels of Understanding in Human-Computer Dialogue

One goal of this thesis is to detect occurrences of miscommunication in physically situated dialogue. Detecting this requires monitoring for possible problems at various points of Clark's levels of understanding [27]. Figure 6.6 presents Clark's levels as adapted for human-computer communication by Bohus [10] and Paek and Horvitz [98]. Situated grounding problems happen at the highest level (the JOINT PROJECT level) because the agent generates too many plans (i.e., referential ambiguity in the context of performing an action in the environment) or none at all (i.e., impossible-to-execute instructions). Detecting these types of problems in situated dialogue has been limited to object manipulation [34].

Lower levels of understanding have received greater attention from the literature. The CHANNEL level consists of the transfer of the acoustic signal from the user to the agent through interaction. The agent must correctly endpoint the user's

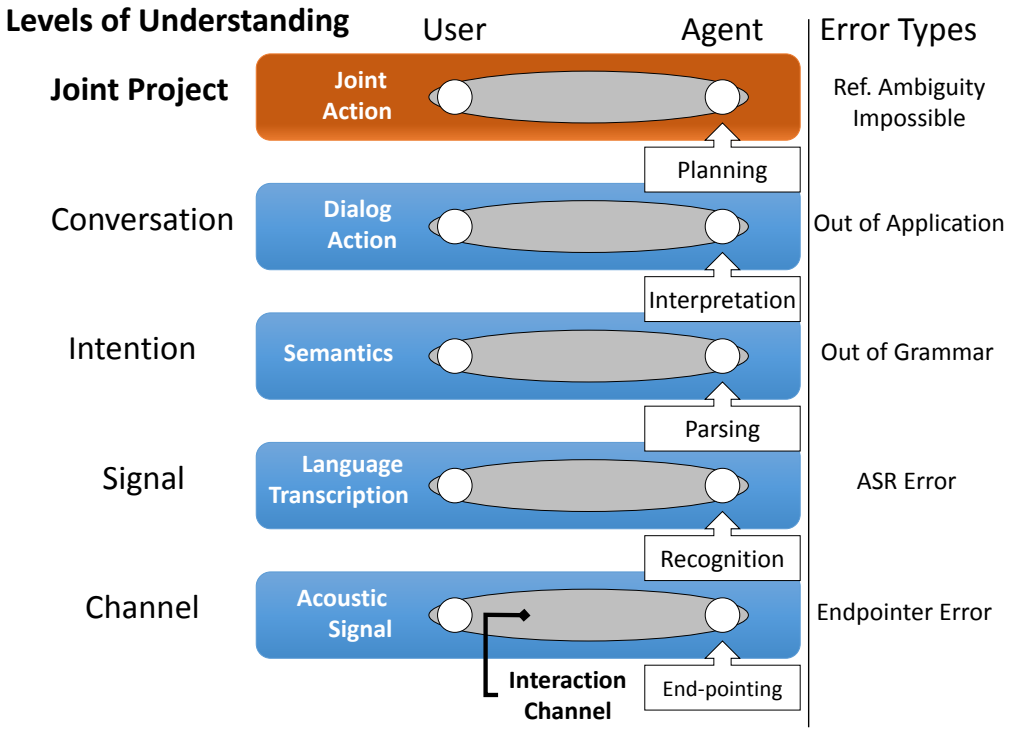



Figure 6.6: Levels of understanding in human-computer dialogue (extended from [10, 27, 98]). This thesis contributes to the JOINT PROJECT level of understanding by handling errors like referential ambiguities and impossible-to-execute instructions. The white circles indicate the user-agent transfer for that particular level.

utterance, and failure to do so, like cutting off a user mid-sentence, results in an endpointer error. Raux and Eskenazi developed a method to optimize endpointing for spoken dialogue systems using dialogue features [108]. The SIGNAL level involves the communication of words from the user to the agent. The agent's speech recognizer is responsible for transcribing the acoustic signal into words, and misinterpretation of words results in recognition errors. The INTENTION level describes the transfer of the user's intentions (i.e., semantics at the level of requested actions and other responses that drive the dialogue) to the agent. The agent's parser attempts to extract this information from recognized speech, and any words that fail to parse are out of grammar errors. Traditionally spoken dialogue systems use context-free grammars to define the semantics of any expected interactions with users. Bohus [10] and Skantze [126] used features from the SIGNAL and INTENTION levels to detect miscommunication in spoken dialogue systems. The

CONVERSATION level describes the transfer of dialogue actions from the user to the agent. Dialogue actions include commands, confirmations, and requests for information. Any dialogue action that falls beyond the scope of the actions the agent can do is an out of application error.

Most dialogue systems are capable of handling errors up to the CONVERSATION level of understanding. The JOINT PROJECT level describes the overall joint action taking place in the dialogue at that moment, whether the user is requesting one or the agent. According to Clark [27], at this level, joint actions are considered joint projects because one speaker is conveying an action both dialogue partners undertake. They must coordinate their actions in order to accomplish the project. In terms of human-computer dialogue, joint projects require the agent to plan out a series of actions to accomplish what the user requests, and in turn the agent expects the user to monitor the situation to ensure the agent is accomplishing the plan correctly.

### 6.3.2 Core Capabilities for Detecting Miscommunication

Any exchangeable information during the course of a dialogue can be used for grounding, and by extension, to detect miscommunication. That information can ground who has the conversational floor, along with the words, actions, and plans under discussion. The agent must maintain and update its notion of physically situated context over the course of interactions with people. This requires monitoring the surroundings, user input, and any uncertainties associated with them. The grounding actions dialogue partners take involve expressing some form of context as part of the grounding process. Clark's levels of understanding [27] align with different sources of context. For example, presenting confirmations of words happens at the SIGNAL level, while clarifying actions and their referents happens at the JOINT PROJECT level. As such, an agent must have several core capabilities to detect situated grounding problems at the JOINT PROJECT level.

We describe core capabilities as task-independent and physically situated *skills* that the agent uses to detect and repair problems at the JOINT PROJECT level. Managing and maintaining context requires task-independent skills (e.g., awareness of too few/too many choices to resolve a referent, knowing when to appeal to human for help, awareness of poor speech recognition confidence) and those skills specific to physically situated tasks (e.g., spatial reasoning, color detection). Task-independent skills should be activated from within the dialogue manager and

physically situated skills from within the domain reasoner.

### Task-Independent Skills

In this thesis we define task-independent skills for detecting situated grounding problems. If a plan request from some spoken language input resolves to more than one possibility, the system assumes there to be a referential ambiguity. Table 6.5 presents this as one of several heuristics. Resolving the referential ambiguity boils down to presenting one or more possible choices to the user. Detecting multiple possibilities and presenting the list of options to the user is a task-independent skill for referential ambiguity recovery.

Alternatively, if a plan request has no possible matches, the system assumes the request was impossible to execute. A behavior observed in the crowdsourcing study in Section 5.5 from the previous chapter motivated us to design another task-independent skill, that of refashioning an impossible command to a less restrictive one with one or more possibilities. Detecting an impossible instruction and presenting alternatives is a task-independent skill for impossible recovery.

When the agent faces multiple possibilities, distinctive properties of the choices make presenting a list viable. Determining the saliency of properties in the list (e.g., color in the virtual agent navigation domain) is useful in these situations. A model of saliency for a domain can inform a dialogue agent that some questions are better to ask than others. Determining the saliency of properties in a candidate list is a task-independent skill for both referential ambiguity (describing the ambiguous options using distinctive properties) and impossible recovery (describing alternative options using distinctive properties).

In summary, task-independent capabilities for dialogue in this thesis are: (1) determining the number of matches to a referent, (2) presenting a list of choices to the user, (3) refashioning otherwise impossible requests and presenting alternatives and (4) building a notion of saliency about the properties of candidates in a list.

### Physically Situated Skills

Task-dependent skills can vary; in this thesis we consider only those for physically situated tasks. Skills for the airline reservation task, for instance, may or may not transfer to that of robot navigation. An agent performing a room reservation task must deal with database lookup errors and date-time conflicts. Physically situated skills include awareness of proximity of some object to other objects or to the

agent itself ("the door near you"), awareness of spatial properties of a referent to the agent ("the box on your left"), awareness of the location/color/size of an object, and temporal information if it influences task execution. When the agent must appeal to the user for help, determining the right question can be a combination of task-independent and task-dependent skill.

Physically situated dialogue interactions are the focus of this thesis. Specifically, within the robot navigation domain described in this thesis, the core capabilities are: (1) occupancy, (2) proximity, (3) orientation, (4) color, and (5) geometry. *Occupancy* is awareness about the positions of objects, walls, and space. It requires reasoning about space and where space is occupied. *Proximity* describes how close the agent is to each object in the environment. Proximity requires reasoning over position (the $(x, y)$ coordinates of dialogue partners and objects in the environment) and distance (how close or far an object is from the agent or other objects). Understanding spatial terms requires awareness of one's own position, and how surrounding objects represent the front, back, left, or right of the agent. *Orientation* provides information about the objects each partner is facing and the other objects each object is facing. It requires reasoning about spatial terms like left and right. *Color* requires awareness about the colors of objects, and whether different objects of the same type differ. Logic about distinctiveness can influence the usage of color. For example, when disambiguating a referent, usefulness of using color deteriorates when more objects of the same basic object type also have the same color. Finally, *geometry* awareness requires reasoning about the shape and size of objects. Logic about relative large and small sizes can play a role in the saliency of an object. Distinctive shapes also require reasoning about the geometry of objects. Both color and geometry are encoded in the robot's environment, but are entered *a priori*. Automatic methods of color and object recognition, along with other sources of context, are outside the scope of this work.

### 6.3.3 Maintaining Environment Context

Context management is achieved using task-independent skills (via RavenClaw's grounding manager) and task-dependent skills (via any dialogue system's domain reasoner or application backend). The core task-independent skills depend on the processing of task-dependent skills to function. For example, being able to determine that a referent "box by the bed" is ambiguous requires knowledge about proximity (e.g., if there are multiple boxes by a single bed, the agent must ground

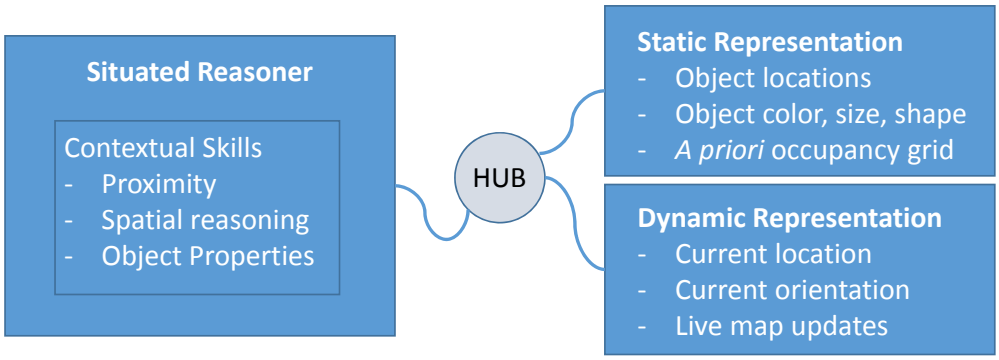



Figure 6.7: The Situated Reasoner combines static and dynamic information to detect situated grounding problems.

the box using and alternative yet distinctive property).

## 6.4 Detecting Grounding Problems with the Situated Reasoner

The *Situated Reasoner* is a novel extension to the TeamTalk dialogue framework that enables the agent to complete tasks relevant to physically situated interactions, especially in the context of navigation. We now describe the Situated Reasoner, which merges sources of context, derivable skills, and language processing. With this information and processing, the agent can accomplish tasks that were previously not possible with the TeamTalk framework.

### 6.4.1 Overview

One of the primary contributions of this thesis is the development of the *Situated Reasoner*, a software extension to TeamTalk's domain reasoner for physically situated contexts. Figure 6.7 presents the Situated Reasoner as it references static and dynamic contextual information. The Situated Reasoner is a suite of functions and algorithms for executing physically situated tasks. It contains the functionality necessary for detecting situated misunderstandings in the robot navigation domain (namely ambiguous requests like *go to the door* when there are multiple doors and impossible-to-execute ones like *go through the door* when it is not open). This includes the dialogue task tree to interpret navigation instructions and the

domain reasoning necessary to process commands in situated interactions. More specifically, given a route instruction command to a robot (like a move command to a landmark *go to the box*), this component will query a separate ontology that represents the robot's surroundings. The goal is to see how many of the landmark type are in the environment, and if there are multiple of a given type, detect that issue and report it to the user. In addition, the Situated Reasoner provides all the necessary message passing to trigger RavenClaw-relevant dialogue actions.

The Situated Reasoner also is capable of resolving spatial relationships between the robot and landmarks in the environment. Given the robot's current position in the environment and it's relative rotation to NORTH, the component calculates if a given landmark is "in front", "behind", "to the right", or "to the left" of the robot itself. The component calculates the *total relative rotation* the robot must make to face a given landmark, and determines the spatial relation. This component is also responsible for enabling the agent to reason about proximity relationships between objects (e.g., identifying the "box near the bed" in the environment) and between objects and the robot (e.g., "the box near you"). The approach is to determine proximity via a backoff model where "near" either means objects within a personal distance (approx. 1m) of each other, within the same room (rooms are entries in the ontology), and social distance (approx. 2m). It consists of functions and classes written in C++.

### 6.4.2 Representing Context in Physically Situated Dialogue

We now present a representation of context that ties together spoken language information with an agent's surroundings and plan information. While some context can be represented statically and retrieved from a structured knowledge base, dynamic information must be recomputed at various moments in a dialogue.

#### Language Context

Language context encapsulates all of the information extracted from the speech signal. Features from the language context include information about the nature of the language used, semantic structures extracted from the language, and confidence about how well spoken language was interpreted. Determining the referable objects in a spoken dialogue system requires defining them in the speech recognition vocabulary and tying words to objects they represent (i.e., *symbol grounding*). In turn, actions associated with those objects define the space in which those

objects can be used. Once spoken language is automatically transcribed, the parser produces a semantic representation. The dialogue manager determines relevant actions and any objects or terms relevant for those actions. The process of binding spoken language to the *concepts*, or main variables discussed during the dialogue, is a form of grounding. For a move command, this may be an attempt to bind an observed word to the *landmark* concept for a *move-to-goal* action.

### Physically Situated Context

Physically situated context is comprised of information about the agent's surroundings and automated planner. Robots collect this information using sensory devices like cameras and laser rangefinders. Virtual agents can simulate the sensors that robots traditionally have. We describe two types of physically situated context, static and dynamic. *Static context* usually does not change and can be stored as entries in a knowledge base. *Dynamic context* changes as an agent moves around an environment.

### Static Physically Situated Context

Static context is information that can be stored in a knowledge base and retrieved with queries. In this thesis, we populate an ontology with static information like the names of objects, their known locations (these do not change), and their color, shape, and size. Since objects do not change position in this work, the object or objects adjacent to an object are also static context. Whether or not the object can be passed through (e.g., an open doorway) is static, along with whether or not that object is open or closed. The agent's dialogue history is also static as it is logged for future access. All static context elements share several properties: they can either be private to an agent or shared with a user, and they can either be visible or hidden from the agent.

We use the Protégé Ontology Framework to store static context. The ontology uses the classes presented in Figure 6.8 for objects in the agent's environment and past interactions in the agent's history. Instances that represent individual objects populate the classes. Those instances have the properties defined for the classes. For instance, while all *single state* and *dual state* share the color property, only *dual state* objects have an openness property that indicates whether they are open or closed. Figure 6.9 presents an example instance of a *lamp* object. For any given object, the ontology supports the *name* of the object, its $(x, y)$ location, the

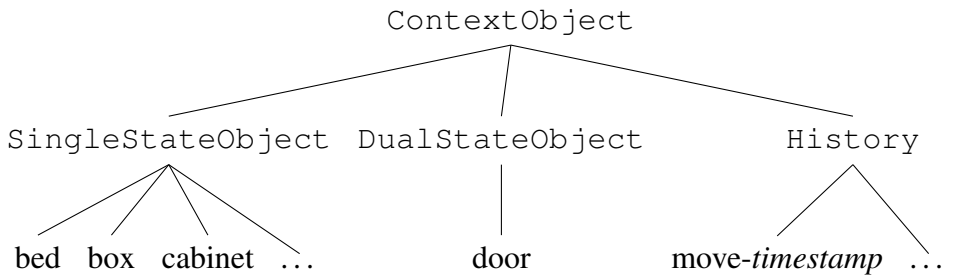



Figure 6.8: Ontology classes in the virtual agent navigation domain.

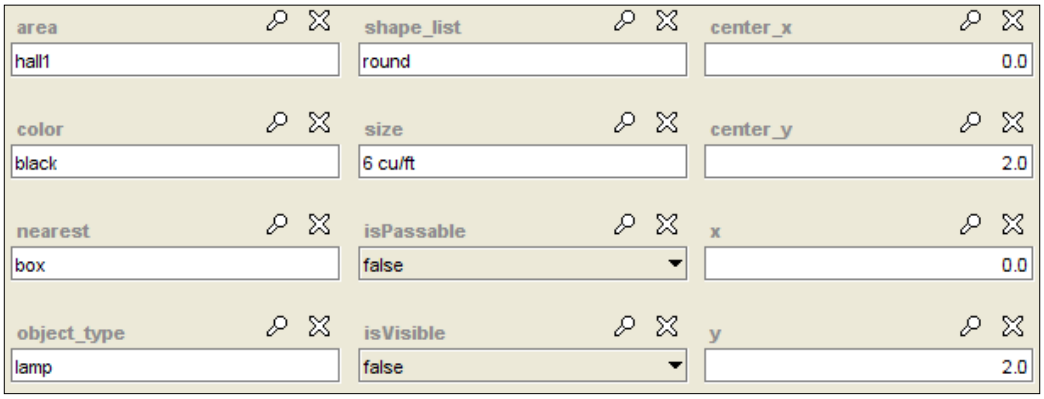



Figure 6.9: An instance of a lamp object in the Protégé Ontology Editor.

*area* where it is located, the *color*, *basic object type*, *shape*, and *size* of the object, *nearest object*, along with boolean flags for its *openness* and *visibility*; all of which are specified in Table 6.1.

Walls do not change in the agent's environment. This assumption allowed us to define another type of static context, an *a priori* occupancy grid that the agent could use for grounding plans. The $16 \times 25$ occupancy grid assigns a binary value to each point in order to indicate whether it is occupied or free space. We used Lester's implementation of $A*$ pathfinding [72] to process the grid in real time. When the agent receives an unambiguous navigation command, the $A*$ pathfinding algorithm checks to see if a path is possible to the candidate goal destination. When there is no possible path, the agent detects these as a type of impossible-to-execute instruction.

The *a priori* occupancy grid also plays an important role in restricting which objects the agent can see as it moves throughout an environment. If a wall blocks a straight path to an object, then the object is not visible to the agent. However, when it can generate a straight path to an object, the agent records it as visible. Each

instance of an object has a binary value associated with its visibility. Before the agent enters an environment, all objects are marked as not visible.

### Dynamic Physically Situated Context

Dynamic context must be calculated in real time based on the current state of the agent's environment. As the agent moves in the environment, streaming sensory input causes this information to change. Whenever the user requests dynamic contextual information, or if the agent needs it to accomplish a task, it must be recomputed. The agent cannot assume that dynamic properties related to the environment are the same from one particular point in time to another. Examples include the agent's location and orientation, and area (i.e., a unique identifier for a room or hallway). Physically situated grounding involves processing spatial relations between the agent and other objects, proximity relations between objects, and its inferring of proximity between other objects and itself (e.g., object nearest the agent).

Temporal properties can change the context of a situation as well. In traditional dialogue systems, a common temporal property is the time of day. It can change what it means to "book a room for today" or "book a room for now" in the hotel reservation domain. In a physically situated domain like navigation, its memory of past actions (i.e., history) serves as a temporal sequence of moves. The location at time step $t0$ changes relations the agent may have with its surroundings, compared to $t1$ (after it initiates a task).

While the agent does have *a priori* knowledge about walls in the environment, it does not have that information about objects. It must collect occupancy information about objects in real time. This is important because some objects could obstruct the agent's path to a goal. We simulate this information from mapping data collected by the agent in real time. The agent's MOAST simulation software uses an $A*$ path planner that operates over streaming SLAM mapping information (this path planner is separate of the $A*$ pathfinding algorithm mentioned earlier that uses the *a priori* occupancy grid). This dynamic path planner uses position information to find a short path to goal locations (i.e., $(x, y)$ points in the map). The agent uses MOAST obstacle avoidance libraries to prevent collisions with virtual objects. This means that tasks may become impossible whenever the agent finds no possible path because of a newly observed obstacle.

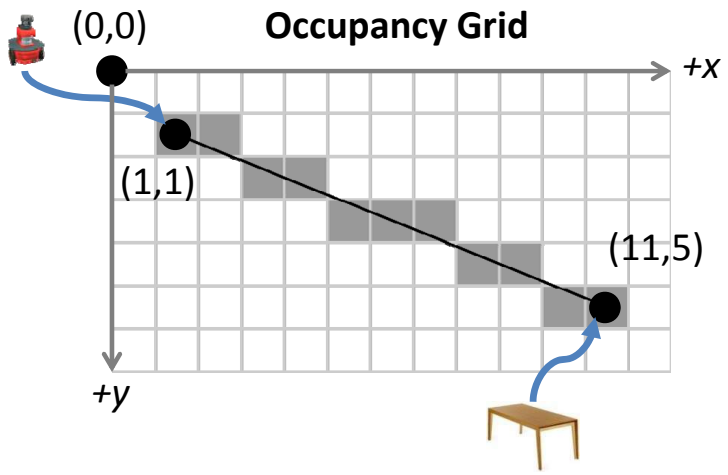



Figure 6.10: The agent calculates field of view between itself and referents using Bresenham's line algorithm [17].

**Field of View**

Visibility is an object property that can be stored as static information. In this work, we store it as a boolean flag. However, it must be reassessed in real time until a state change occurs, i.e., when the object becomes visible to the agent. We make the assumption that the ontology represents full knowledge of the environment, but restrict the agent's discourse to only those objects that it can see. Field of view simulates which objects the agent can and cannot see. We accomplish this using a combination of constraints applied to the agent's surroundings. First, when an agent enters an environment, it can only see objects in front of it within a 180-degree field of view. Whenever the agent updates its position, we recalculate object visibility. We use Bresenham's line algorithm to calculate line of sight between the agent in its current $(x, y)$ location and the $(x, y)$ locations of all surrounding objects [17]. The algorithm is a computationally efficient method for tracing lines in a graph. As the algorithm traces a line of sight, the point for each part of the line is checked in the *a priori* occupancy grid. The occupancy value for that location determines if that space is occupied by a wall. Whenever the agent calculates its line of sight between itself and an object, if a wall blocks it, the object's entry remains marked as not visible. Figure 6.10 illustrates Bresenham's line algorithm being calculated between the agent and a table in its surroundings.

### Representing Context Elements

Context elements that either represent static or dynamic information should be modifiable during any point of a dialogue with a physically situated agent. In this work, the algorithms for computing dynamic information are represented as functions that can be accessed by the dialogue manager in real time. Whether the current point of the dialogue is an initiation of a new instruction, an acknowledgment of an existing one, or reporting a complete status on a task, contextual information is available for the agent to access. Static information is retrievable by querying Protégé's knowledge base and processing the reply.

### Approach: Iterative Scenario-Building

To develop this representation, we first defined relevant scenarios by sampling existing corpora and simplifying samples into test cases. We looked at dialogues from the SCARE Corpus [128] and the HCRC Map Task Corpus [2]. Some of this work can be found in Chapter 7.1, where we analyzed recovery strategies that were appropriate for physically situated dialogue. We focused on how the direction follower (DF) in these dialogues conveyed situated grounding problems, and the strategies they would take to recover. This analysis also helped us determine the types of problems encountered and the reasoning capabilities necessary to detect and recover. Another extension of our recovery strategy analysis from Chapter 7.1 was to assess the necessary reasoning capabilities for detecting these types of problems. Reviewing the corpora helped us determine the task-independent and physically situated skills described previously in Section 6.3.2. We identified excerpts of dialogues where the DF was using one or more of those capabilities.

Test-driven development enabled us to iteratively design and develop the necessary modules to process language in physically situated domains. We used route instructions from the TeamTalk Corpus [83] to develop the parse structures. The agent could understand the most frequently observed route instruction phrases from the corpus. In total, we used a test suite of over 200 route instructions from the TeamTalk corpus to improve the agent's natural language understanding capabilities.

Test-driven development also allowed us to extend the existing dialogue software to develop physically situated skills using static and dynamic environment information. For instance, for egocentric spatial reasoning, we first developed thresholds for the spatial terms *right*, *left*, *front*, and *back* by considering various di-

alogue scenarios where they would be relevant. Then, we implemented the rotation calculations that enable the thresholds to be calculated given the agent's current position and angle of rotation. Finally, we added egocentric proximity reasoning that enabled natural language interpretation and generation of spatial terms relevant to objects that surround the agent. Section 6.4.4 presents all physically situated skills in more detail.

### 6.4.3 Task-Independent Skills

Dialogue agents gain task-independent skills by interpreting both their language and background knowledge.

#### Sources of Context

The number of possible referent candidates that match the instruction determines the general situated grounding problem, regardless of domain. If there are zero matches for a referent, the instruction was impossible-to-execute. If there is more than one, the instruction was referentially ambiguous.

Actions, as defined for the dialogue manager, often require specific information to succeed. The system can prompt the user for any missing information if is needed to accomplish the task. For example, if the user indicates a move, the agent can prompt for the landmark object. This happens automatically with RavenClaw based on the *Request agents* defined for various actions within the navigation dialogue task.

#### Disambiguation Process

When the agent finds a referent to be ambiguous, it can query its ontology for all known information about matching objects. At the same time, for each match, it cross-references the static information it collects and uses it to compute spatial reference and other dynamic, situated properties of the objects. Once the agent builds a sufficient understanding about the ambiguous matches, it can then formulate a disambiguation strategy by selecting situated properties to describe the matches. In Chapter 8, we describe a data-driven approach to formulating these lists based on the nature of the situated property (one or more of *color*, *size*, *egocentric proximity*, or *object proximity*).

#### Computing Distinctiveness

A crucial skill for delivering an effective disambiguation strategy is to describe objects in a way that is efficient and useful. One critical element to this is that each possible referent can be uniquely distinguished in the environment. Put another way, a distinguishable characteristic of an object is one or more situated properties that set it apart from any other object in the agent's view. An effective strategy is one that can deliver this information in a compact way.

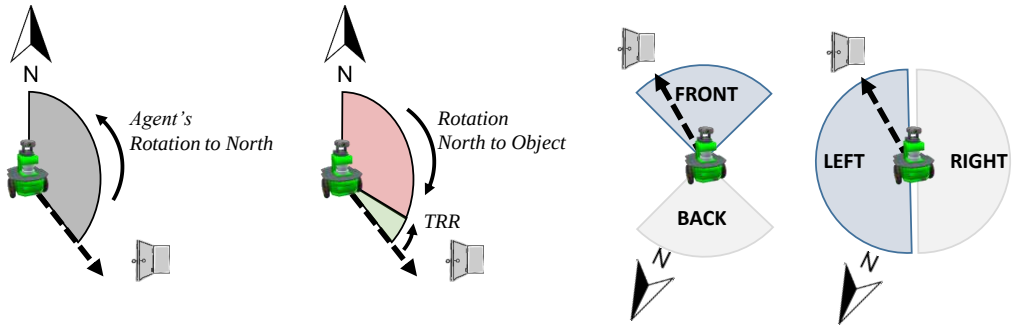
We implemented a method for computing minimally distinctive properties for situations that were referentially ambiguous. Given a list of matching candidates, the agent first determined which situated property types were defined for each one. These could either be *color*, *size*, *egocentric proximity*, or *object proximity*. For each situated property type, the agent checked if more than one candidate had the same value. Situated property types that were both defined and that did not have multiple candidates with the same property value were marked as distinctive. For each possible match, either a heuristic method (calculated using the most frequently used strategies from Chapter 5) or a learning-based method (see Chapter 8) determined which among the possible distinctive situated properties were used in a disambiguation strategy.

#### Refashioning an Overspecified Instruction

Sometimes the agent receives instructions that overspecify a referent. In these cases, the requested action is impossible because there are no matching referents in the environment. The agent can propose alternatives using the constraint relaxation techniques described in Section 4.2.2. In effect, the agent takes a proactive approach for an otherwise impossible-to-execute instruction. Each property that is used to describe an object, including the object type itself, places a constraint on the possible objects that could match an intended instruction. By relaxing or removing these constraints, the agent can generate alternative plans. Whenever there are ambiguous alternative plans, it can refer to its distinctiveness and disambiguation skills to present a list of alternatives.

### 6.4.4 Physically Situated Skills

Physically situated skills rely on static and dynamic information. Given a command that references an object and a search space of objects from the agent's




(a) (b) (c) (d)

Figure 6.11: The agent calculates egocentric spatial relations by calculating its total relative rotation between itself and a door referent, using cardinal direction NORTH as a global point of reference. The algorithm (a) determines the agent's rotation relative to NORTH and (b) the subsequent rotation to the door, resulting in the calculation of *total relative rotation* between itself and the door. The agent determines the door to be (c) in front and (d) to its left using spatial thresholds with the *total relative rotation* value.

knowledge base that match the basic object type, narrow the search space if a spatial relation or proximity relation is provided. In this section we present skills for processing egocentric spatial relations, proximity relations, and referencing intrinsic properties.

### Egocentric Spatial Relations

The agent achieves an understanding of spatial information by calculating egocentric proximity between itself and surrounding objects with real-time position information. Figure 6.11 presents the approach the agent takes to determine egocentric spatial relations between itself and a referent. The agent uses its current $(x, y)$ position, the $(x, y)$ positions of all matching referents, and the angle of rotation of the robot relative to the cardinal direction NORTH. With this information it calculates its *total relative rotation*, the amount of rotation (in radians) that it would need to rotate to face each referent. Positive radians refer to space on its left side, negative its right. This rotation is then classified as either being "to the right" or "to the left" of the agent, and optionally also "in front" or "behind" it.

The general idea of this approach is that the agent simulates a mental rotation from its current position to the referent, using thresholds to determine spatial relations. For instance, if the instruction is "Move to the box on the right", the

**Algorithm 6.1:** The egocentric spatial reasoning algorithm determines spatial relations between the agent and a referent.

**Input**: $(x, y, \theta)$ of agent and $(x, y, \theta)$ of a referent
**Output**: Spatial relation (one or more of *front*, *back*, *left*, *right*)

1 Calculate the amount (in radians) the robot must turn to reach NORTH.
2 Translate the robot location to the origin $(0, 0)$, apply that translation to the referent.
3 Calculate the angle between the robot (pointing toward NORTH) and the referent.
4 Calculate the *total relative rotation* (TRR), which is simply the sum of the two angles obtained from line 1 and line 3.
5 Calculate if the rotation puts the referent relative to the agent in the desired spatial region. **return** spatial relations within thresholds.

agent first retrieves all visible boxes in its static knowledge base. It then restricts the list to only those on its right by computing *total relative rotation*. Algorithm 6.1 presents this approach. If the *total relative rotation* meets either of these thresholds: $(0 < TRR < \pi)$ OR $(-2\pi < TRR < -\pi)$ then the object is LEFT of the agent. If $(-\pi < TRR < 0)$ or $(\pi < TRR < 2\pi)$, then the object is RIGHT of the agent. An object is said to be DIRECTLY IN FRONT of the agent if $(TRR = 0)$, $(TRR = 2\pi)$, or $(TRR = -2\pi)$. An object is IN FRONT of the agent if $(-\pi/4 < TRR < \pi/4)$, $(TRR < -7\pi/4 < 2\pi)$, or $(-2\pi < TRR < -7\pi/4)$. An object is DIRECTLY BEHIND the agent if $(TRR = \pi)$ or $(TRR = -\pi)$. An object is BEHIND the agent if $(3\pi/4 < TRR < 5\pi/4)$ or $(-5\pi/4 < TRR < -3\pi/4)$. These thresholds were fixed throughout our experiments.

The algorithm is useful for both spatial interpretation and spatial description. To reduce occurrence of unintended impossible instructions, the algorithm attempts to find as many spatial relations that could match it as possible. For instance, if the agent is both in front and to the right of a referent, a mention of either a front or right relation would match. Alternatively, when it must produce a description about a referent that uses its egocentric proximity, it uses as restrictive a distinction as possible. In our algorithm, the agent first checks if its *total relative rotation* results in a front or back relation. These have more restrictive thresholds than left and right. If not, it determines if the relation is left or right.

**Algorithm 6.2:** The proximity reasoning algorithm determines *nearness* relations between a referent and its proximity object.

**Input**: $(x, y)$ of a referent and $(x, y)$ of a proximity object
**Output**: Proximity relation of *nearness*, if appropriate

1 Check that there is only one proximity object. For *near* and *nearest* cases, if more than one of a basic object type that matches the proximity object in the environment, find only the ones that are near referents.
2 Given a search space of referent matches, locate only the ones with the proximity relation. We assume there is a hierarchy of proximity: *Personal distance* $>>$ *Social distance*.
3 For each referent match, count how many of them are $< 1$ meter (within range of *personal distance*) from the proximity object. If there are a nonzero number of matches, **return** all *nearness* relations.
4 For each referent match, count how many referents are $< 2$ meters (within range of *social distance*) from proximity object. If there are a nonzero number of matches, **return** all *nearness* relations.

### Proximity of Surrounding Objects

Our approach to proximity leverages the existing body of work on proxemics [45]. We make the simplifying assumption that the proxemics thresholds we set pertain to American and European cultures. Given a possible referent and an object in close proximity (we term this a *proximity object*), find any referent matches that have the desired proximity relation. For example: "Go to the box NEAR the bed" changes to the task of finding all boxes near beds, where $box = referent$; $bed = proximity\_object$. The agent uses the $(x, y)$ positions of referent matches and proximity object matches in the environment to determine *nearness* relations.

The proximity reasoning algorithm can be found in Algorithm 6.2. For any proximity calculations that refer to the agent itself as the proximity object (e.g., *the box nearest you*, the agent replaces the proximity object with its own $(x, y)$ location.

### Intrinsic Object Properties

Intrinsic object properties are obtained when the agent queries its ontology for information about its surroundings. Queries are specific to referents mentioned in the commands that the agent receives. Responses to queries take the form of a string object that contains the *basic object type*, *color*, *size*, and *shape*. These properties

are stored as text strings in the ontology. Figure 6.9 shows example entries for each of these properties for a lamp object. When the agent receives a reference to an object that mentions an intrinsic property (e.g., *black lamp*), it checks if there is a complete match in the reply from the ontology. If the reference mentions an intrinsic property that does not match any entries in the agent's ontology (e.g., *blue lamp*), any action associated with the command would be impossible. The agent would initiate the constraint relaxation techniques described in Section 4.2.2.

### 6.4.5 Evaluation

In addition to the test-driven development approach described in Section 6.4.2, we also conducted evaluation using the instruction set used in Chapter 5. These 50 simulated route instructions were used to evaluate the implementation of the TeamTalk system with the Situated Reasoner component versus without it. TeamTalk without the Situated Reasoner was only able to correctly detect 11 of the 50 instructions as executable, ambiguous, or impossible-to-execute. The Situated Reasoner enabled the system to correctly detect 48 of the 50 instructions.

## 6.5 Chapter Summary

In this chapter, we described the infrastructure and representation for detecting situated grounding problems. We defined the problem space using a domain ontology for virtual agent navigation. Next we described the research platform used in this thesis, the TeamTalk dialogue framework featuring the USARSim robot simulation software. We then described the core capabilities that the agent must have in order to handle situated grounding problems. In the next section, presented a rich representation of context for physically situated dialogue that combines spoken language information with static and dynamic sources of environment information. We presented the Situated Reasoner, the software component that reasons over this information and detects miscommunication events in physically situated dialogue. Finally, we described emergent task-independent and physically situated skills when these sources of information are leveraged together.

Chapter 7

# Recovery Strategies for Physically Situated Contexts

Recovery strategies are actions a dialogue agent can take to repair grounding problems. In this chapter, we first describe an analysis of human-human navigation corpora, identifying where people following directions would initiate recovery strategies. These recovery strategies were revisited after we collected and analyzed a dataset of recovery strategies for physically situated dialogue (i.e., the crowdsourcing study described in Chapter 5). In the second part of this chapter, we present recovery strategies an agent can use to recover from situated grounding problems. Recovering from *referential ambiguities* requires an agent to reason over multiple plan possibilities. In contrast, recovering from *impossible-to-execute* instructions should allow an agent to express what it cannot execute, estimate the user's intention, and propose alternative plans.

## 7.1 Understanding Human Miscommunication Behaviors

In order to understand the variation of strategies that people take when following route instructions, we examined existing human-human dialogue corpora in the navigation domain [82]. In these corpora, a pair of dialogue partners discussed navigation tasks. One person is assigned as a direction giver (DG) and the other a direction follower (DF). Either can ask questions of the other, but the DG delivers spoken language instructions that the DF attempts to follow.

### 7.1.1 Human-Human Navigation Dialogue Corpora

The purpose of this work was to develop a taxonomy of questions and a frequency analysis of those question types. The taxonomy laid the groundwork for developing a set of recovery strategies agents can use to recover from situated grounding problems. This work also gave us a better understanding of the types of situated grounding problems that happen. Common problems that happen in these corpora may also be common in human-robot dialogues. In the scenarios we developed for this thesis, the physically situated dialogue agent is an instruction follower, so our analysis is primarily on follower behaviors (e.g., questions they ask the DG, expressions they say to convey what they know or don't know, statements of what they plan to do). In addition we examined these corpora for the most common factors that lead to DF questions.

We will discuss examples of miscommunication occurrences in two human-human dialogue corpora, the HCRC Map Task Corpus and the SCARE corpus.

#### HCRC Map Task Corpus

The HCRC Map Task corpus consists of 128 unscripted English human-human dialogues (64 participants) in the navigation domain [2, 82]. Participants were tasked with verbally discussing how to replicate a path on one participant's (complete) map on the partner's (incomplete) map. The experimenters assigned participants to either the role of the *direction giver* or the *direction follower*. Both dialogue partners had a 2-dimensional schematic map of a fictional area, containing landmarks in various places. The key variation between the dialogue partners was that the direction giver had a labeled path on his map from start to finish, while the direction follower only had the start point labeled. The follower needed to trace the path that the direction giver described on his own map. They could not see each other's maps.

Since the participant task in this corpus is navigation, there are clear measures of communicative success. For subgoals, one measure of success is if the direction follower successfully reached the landmark or landmarks that the direction giver described. The overall task was a success if the direction follower arrived at the finish point on the direction giver's map.

This corpus fits our interests because recovery strategies for resolving miscommunication are very common in this corpus; in fact, the Map Task's dialogue act coding scheme contains labels for such strategies [22]. The *check* code for a

turn indicates when a dialogue partner needs to confirm something with the other that he is not certain about. The *query-yn* and *query-w* codes label questions one dialogue partner asks of the other (yes/no questions or otherwise). These codes can be used to identify recovery strategies as they occur in the dialogues.

#### SCARE Corpus

The SCARE corpus (Situated Corpus with Annotated Referring Expressions) contains fifteen English dialogues with direction giver-direction follower pairs [128]. In this corpus, the DG guides the DF through a virtual world (the DF was situated directly in the environment). The DG had an annotated schematic map of the scene and sat in front of a monitor that showed the DF's view of the environment (the DF was in another room). Partners shared information not only by spoken dialogue but also by monitoring the DF's gaze and position in the virtual world. The corpus allowed us to actively track these behaviors as it had recordings of the participants' audio and visual channels (video).

The DG gave navigation and manipulation instructions to the DF throughout the session. Some tasks required the DF to push buttons, while others required the DF to pick up objects and hide them in other places. This corpus has many instances where the follower asks questions about navigation and manipulation tasks that the giver assigns. We hypothesize that these examples can help inform the strategies that dialogue agents use when they listen to human instructions.

### 7.1.2 Recovery Strategies from Dialogue Corpora

We observed instances of questions about route instructions in the Map Task and SCARE corpora from direction followers. Below we provide examples of dialogues where the DF uses recovery strategies to recover from referential ambiguity and impossible instructions. We identified the physically situated skills used by the DF in the dialogues (either *egocentric proximity*, *proximity to object*, *proximity to self*, or *intrinsic object property*).

#### Strategy: Confirming a single referent (Referential Ambiguity)

From these examples, we learn that disambiguating a referent using the DF's own location can quickly alert the DG of a problem and prompt repair.

***DF:*** **it's the one I'm closest to this one?**
*DG:* that one that's correct
(SCARE, Dialogue S4. Skill: *Proximity to self*)

***DF:*** **the button farthest from you you wanna press?**
*DG:* no no no
(SCARE, Dialogue S13. Skill: *Proximity to self*)

*DG:* that one
***DF:*** **this one right in front of me?**
*DG:* yes
(SCARE, Dialogue S13. Skill: *Proximity to self*)

In the following dialogues, we observe the DF making reference to proximal relations between one referent and other referable objects in the space. In one case, the lack of a referable object makes it distinctive (*by itself*), but that reasoning requires knowledge that other referents have proximal objects.

***DF:*** **on the right wall?**
*DG:* yeah
(SCARE, Dialogue S3. Skill: *Proximity to object*)

*DG:* I would like for you to press the one all the way on the left
***DF:*** **meaning the one that's like not on the same wall as the others?**
*DG:* that's correct
(SCARE, Dialogue S4. Skill: *Proximity to object*)

*DG:* that's right so see this button on your left?
***DF:*** **the one that's by itself on the wall?**
*DG:* yeah
(SCARE, Dialogue S4. Skill: *Proximity to object*)

*DG:* ok see that door in front of you?
***DF:*** **next to the stairs?**
*DG:* yeah
(SCARE, Dialogue S12. Skill: *Proximity to object*)

In the next few dialogues, the DF makes use of egocentric spatial reasoning. The

DF takes his or her own perspective in these situations.

*DG:* um and then curve to the east above a bakery
***DF:* no so you do you want me to go right round pine forest?**
*DG:* no just 'til you're underneath it and then go south when you're underneath it
(Map Task, Dialogue Q1EC4. Skill: *Egocentric proximity*)

*DG:* oh yeah there are three doors um let's see there the door that is not on the wall that has a cabinet so
***DF:* the middle door?**
*DG:* right yeah there we go
(SCARE, Dialogue S2. Skill: *Egocentric proximity*)

*DG:* ok so you're in here and go through the door all the way on the
***DF:* right?**
*DG:* um straight in front of you
(SCARE, Dialogue S4. Skill: *Egocentric proximity*)

*DG:* it's the yep it's that one
***DF:* you want the one on the left that we want?**
*DG:* yeah they want the one next to the door
(SCARE, Dialogue S5. Skills: *Egocentric proximity, proximity to object*)

In the following dialogues, the DF mentions an intrinsic property of a referent to uniquely distinguish it with the DG.

*DG:* ok g- um go just go through the door
***DF:* back through the blue door?**
*DG:* yeah
(SCARE, Dialogue S2. Skill: *Intrinsic object property − color*)

*DG:* the one closest to you is cabinet four
***DF:* the one that's open?**
*DG:* yeah
(SCARE, Dialogue S3. Skill: *Intrinsic object property − openness*)

*DG:* and then go back through
***DF:* the blue room?**
*DG:* yeah the blue room
(SCARE, Dialogue S3. Skill: *Intrinsic object property − color*)

***DF:* the thing with the *S* on it right?**
*DG:* yeah
(SCARE, Dialogue S5. Skill: *Intrinsic object property − label*)

**Strategy: Listing options and asking a question (Referential Ambiguity)**

An alternative strategy to simply asking a confirmation question is to present a list of choices. Instead of expecting a yes-no answer, the DF presents a list and expects the DG to select one. The HCRC Map Task has few situations where objects in the maps were referentially ambiguous. In the next few dialogues from the SCARE corpus, DF partners presented a list of options, and used either egocentric spatial reasoning or an intrinsic property to disambiguate.

*DG:* yeah there'll be another door there
***DF:* this one or the left?**
*DG:* the one on the r- um take the one straight ahead
(SCARE, Dialogue S2. Skill: *Egocentric proximity*)

*DG:* and go forward go left or yeah that one straight
***DF:* ok this door or that door?** (pointing with shared gaze)
*DG:* that that one
(SCARE, Dialogue S3. Skill: *Egocentric proximity*)

*DG:* and close them
***DF:* this button here or second from the left?**
*DG:* mhm
(SCARE, Dialogue S13. Skill: *Egocentric proximity*)

*DG:* ok so we'll go to the next one
***DF:* ok the next button or next room?**
*DG:* the next task
(SCARE, Dialogue S5. Skill: *Intrinsic property − basic object type*)

**Strategy: Proposing an alternative plan (Impossible-to-Execute)**

Nearly all situations with the SCARE corpus lacked impossible-to-execute problems; this was due to the nature of the task (navigation without impossible tasks) and the channel of communication (shared gaze enabled the DG to ground what the DF could view). However, the HCRC Map Task reveals several occurrences of impossible-to-execute recovery. In general, the DF has trouble interpreting the initial instruction of the DG and attempts to refashion it. Direction followers primarily take the strategy of proposing an alternative referent, similar to our observations in our own crowdsourced data collection (Chapter 5). The next few dialogues present instances of a DF initiating a recovery from an impossible-to-execute instruction.

*DG:* just go a couple of steps
***DF:* have I to pass pass by the waterhole?**
*DG:* it's stony desert that's what I've got here
(Map Task, Dialogue Q1EC2. Skill: *Proximity to object*)

*DG:* we are going to go due south straight south and then we're going to g– turn straight back round and head north past an old mill on the right hand side
***DF:* due south and then back up again?**
*DG:* yeah
(Map Task, Dialogue Q1EC1. Skill: *Refashioning alternative plan*)

*DG:* on a– this map the roman baths are there are at the top below the lake and then it's the antelope
***DF:* so you want me to go between the antelopes and the position of the roman bath?**
*DG:* well it one is just below the other and it's come down to the way you'll be facing it'll be your left the roman baths and the antelopes will be to your left
(Map Task, Dialogue Q1 EC2. Skills: *Refashioning alternative plan, Proximity to object*)

*DG:* so go south down past the walled city underneath your ghost town and then slightly up sort of northeast direction between the flat rocks and the stone creek

***DF:* it'd be better if I s– s– between the walled city and the ghost town no to go underneath it above it or have you got something else there?**

*DG:* I've not got anything betw– between the carved wooden pole and the walled city

(Map Task, Dialogue Q1EC5. Skill: *Proximity to object*)

Although the human-human dialogues above have complex language structure, we can still use them to identify recovery strategies that an agent could use. Moreover, observations from human-human dialogues validate the physically situated skills necessary for recovering from situated grounding problems. In the next section, we develop a taxonomy of recovery question types.

### 7.1.3 Tabulating Question Types in the SCARE Corpus

The tasks associated with the SCARE corpus closely resemble the tasks that the agent faces in this thesis. As we reviewed the dialogue transcriptions from this corpus, we developed a set of question types by annotating the topic of each question. The goal was to identify what people were asking about. There were two iterations to this process. First, we annotated the transcribed dialogues for questions asked by the DF. Occasionally some DF turns were not clearly statements nor clearly questions; we annotated these for audio and video review later.

In a second pass over the dialogues, we reviewed the annotations while viewing the experiment videos. This second pass allowed us to validate that initial question annotations were correct and to review points in the dialogues that were not clear from the transcriptions alone. For each question, we also coded the evidence contributing to the follower asking the question (e.g., DG utterances, salient properties of the DF's virtual surroundings, etc.), the DG's answer to the question (if any), and the task that needed to be completed at the time the follower asked the question.

Given the nature of the task that dialogue partners performed, there were only a few major question types.

- *Referential* questions asked about a referent in the environment (e.g., a button on the wall, a door, a cabinet, etc.). Examples: "The middle door?", "Which one?"

| Question Type | Frequency | Percentage of Total Questions |
|---|---|---|
| Referential | 168 | 49.4% |
| Spatial | 40 | 11.8% |
| Action | 104 | 30.5% |
| Plan | 8 | 2.4% |
| Experiment | 22 | 6.4% |
| General | 27 | 7.9% |
| Total # Confirmation Questions | 285 | 84% |
| Total # DF Questions | 340 | 100% |

Table 7.1: Question type frequencies in the SCARE corpus. Questions could have multiple annotations. For all questions, we also counted whether they were confirmations (i.e., expecting yes or no answer). We observed 3,818 direction follower turns.

- *Spatial* questions asked about space or locations in the virtual world. Examples: "Right?" (the direction), "Where do I go?"
- *Action* questions explicitly asked about performing an action or task. Examples: "So I'm not going out the door I came in?", "You want me to press the button?"
- *Plan* questions asked about multiple actions or steps to an instruction. Example: "Around the corner and everything?"
- *Experiment* questions were specific to the properties of experiment setup.
- *General* questions were either non-understandings or off-topic. Examples: "What?", "Oh really?"

Table 7.1 presents frequencies of question types from the SCARE corpus. DF questions were quite frequent, making up roughly 9% of all turns held by the DF. These results show that questions initiated by the DF are a major component to the grounding that is happening between the DG and DF. Referential questions account for nearly half of all questions asked by the DF. Of the questions being asked by the DF, many have to goal of attempting to ground a referent based on the DG instruction. This observation logically follows from the nature of the task; instructions from the DG require navigation of a space with objects. In addition, the space is a virtual environment, so the DG is more likely to use landmarks to

instruct than metric distances (see findings from Chapter 3 on route instructions across presentation formats). Action questions were also important; these were often situations where the DF was confirming the potential execution of a task. Spatial questions had a presence as well, suggesting that spatial reasoning is an essential skill for physically situated agents. All three of these types of questions represent ways to recover from situated grounding problems.

In addition to knowing *what* people were asking about in these scenarios, we also sought to understand, for any question, *why* people were asking about something. For each of these question types, we also coded whether or not the question expected a *yes/no* answer. These question intention types helped us understand the motivation for why people were asking questions. We developed two primary categories:

- *Confirmation* questions confirm something that the follower has already internalized. These are usually in the form of a yes/no question. Examples: "The right one?", "This door?"
- *Clarification* questions ask about something related to missing parameters to an action or when the follower is uncertain about the next action to do. The follower displays no evidence of an initial hypothesis when asking these questions. Examples: "Which one?", "The right or left door?"

Our preliminary analysis of these two question types showed that dialogues have a mix of confirmations and clarifications, so physically situated dialogue agents should be able to ask both types of questions. The results from Table 7.1 show that confirmation questions were more common, representing 84% of all DF questions. We believe this is due to the nature of the SCARE task. *Yes/no* confirmations suggest a choice of at most two referents, or a confirmation of one highly likely one. The shared gaze between DF and DG clears up many potential ambiguities, and reduces the chance for multi-referent ambiguities that would be better sorted out using a list. People can also make more assumptions about their partner's capabilities if that partner is a human. Relevant capabilities include scene recognition, object recognition, gaze processing, and language understanding. Assumptions about an agent's capabilities can vary from person to person for any number of reasons, including their own beliefs about artificial agents and past history dealing with technology. Based on this point and the results obtained from Chapter 5, we decided to use lists as the agent's recovery strategy for disambiguation when possible. Lists provide an opportunity for the agent to

ground what it believes are possible matches for an ambiguous referent and offers a clear list of choices to a human dialogue partner.

## 7.2 Recovery Strategies for Agents

A recovery strategy takes the form of either a display of understanding (i.e., an explicit statement by the agent about what it knows or doesn't know), a question about an utterance, or both. In previous work we began to define recovery strategies for situated grounding problems [82]. We investigated two types of recovery strategies, those for disambiguating referents and those for describing, and potentially refashioning, impossible-to-execute instructions. We defined these recovery strategies using two sources of information: (1) occurrences of recovery strategies in human-human corpora and (2) the contextual information available to the agent.

### 7.2.1 Recovery Strategies for Referential Ambiguity

Recovery strategies to resolve referential ambiguity occur when there are multiple possibilities for converting an instruction to a plan. TeamTalk can display the agent's diagnosis of the problem by mentioning that multiple plans are possible. Questions associated with resolving referential ambiguities are presented below.

The types of questions that will resolve ambiguity will be referent clarification questions using the sources of information described in Chapter 6. In that chapter, we described the agent's physically situated capabilities as egocentric spatial reasoning, proximity reasoning, and intrinsic property (i.e., basic object type, shape, size, and color) awareness. We developed these recovery strategies by observing occurrences of questions by direction followers in the Map Task and SCARE corpora, along with the crowdsourced dataset of human-authored recovery strategies described in Chapter 5. Table 7.2 presents the possible recovery strategies that the agent can take. They make use of physically situated skills mentioned in Section 6.4.4. Most of these strategies are straightforward. A combination of multiple strategies occurs in two cases: (1) the only way to distinguish two or more referents is by mentioning multiple properties and (2) the user mentions one of the properties in its referent, which is maintained in any follow-up questions by the agent for grounding purposes (i.e., showing that the agent understood the user's request).

| Recovery Strategy | Example Dialogue |
|---|---|
| Disambiguation by Egocentric Spatial Relation | *Human:* Move to the box.<br>*Agent:* I see more than one box. Did you mean the box **on my right** or the box **on my left**? |
| Disambiguation by Color | *Human:* Move to the box.<br>*Agent:* I see more than one box. Did you mean the **red** box or the **blue** box? |
| Disambiguation by Proximity to Object | *Human:* Move to the box.<br>*Agent:* I see more than one box. Did you mean the box **near the chair** or box **by itself**? |
| Disambiguation by Size | *Human:* Move to the box.<br>*Agent:* I see more than one box. Did you mean the **big** box or the **small** box? |
| Disambiguation by a Combination of Above Sources | *Human:* Move to the door in front of you.<br>*Agent:* I see more than one door. Did you mean the **brown** door **in front of me** or the **blue** door **in front of me**? |

Table 7.2: Recovery strategies for referentially ambiguous situations.

The agent often can choose between multiple sources of context to disambiguate. In Chapter 8, we describe an approach to recovery strategy selection, whereby the agent learns to select some recovery strategies over others based on previous successful recoveries.

### 7.2.2 Implementation

We implemented these recovery strategies with a `Library Agency` for disambiguation in the RavenClaw dialogue manager. `Library Agencies` exist in RavenClaw's error handling decision process. When RavenClaw detects a situated grounding problem, it will select a recovery strategy to execute. Selection involves first computing all distinctive relations; these are the only ones that are usable in the immediate context. If there is more than one possible strategy, the agent can either use a heuristic model or a learned one to select the appropriate content. The

| Recovery Strategy | Example Dialogue |
|---|---|
| Declare impossible due to ontology mismatch | *Human:* Go through the table.<br>*Agent:* I cannot go through that. |
| Propose alternative plan | *Human:* Go to the blue box.<br>*Agent:* I don't see the blue box. **Choose from the following:** Brown box or red box. |
| Declare impossible due to object state | *Human:* Go through the door.<br>*Agent:* I cannot go through that. |
| Declare impossible due to planning failure | *Human:* Move to the cone.<br>*Agent:* I cannot find a plan to get there from my current location. |
| Declare impossible due to obstacle interference | *Human:* Go to the table.<br>*Agent:* I tried but I could not find a plan to get there from my current location. |

Table 7.3: Recovery strategies for impossible-to-execute situations.

agent generates the situated content of the strategy using respective handlers in the Situated Reasoner (described in Section 6.4.4) that refer to the agent's static and dynamic contextual information.

### 7.2.3 Recovery Strategies for Impossible-to-Execute Situations

Recovery strategies to resolve impossible-to-execute instructions occur when there are no possibilities for converting an instruction to a plan. TeamTalk can display the agent's diagnosis of the problem by mentioning that no plans are possible. Questions associated with resolving impossible-to-execute instructions are presented below.

Recovery strategies for resolving impossible-to-execute instructions may be proposing an alternative instruction or declaring a problem about the current instruction. Table 7.3 presents the impossible-to-execute recovery strategies that the agent can take. We developed these recovery strategies using examples from the HCRC Map Task Corpus and the crowdsourced dataset described in Chapter 5.

These strategies vary in how they attempt to ground a problem with the user. For the *matching-ontology* recovery strategy, Here RavenClaw will attempt to locate a match to the recognized concept (table) to feasible actions for that object in the navigation domain ontology. In the example provided in Table 7.3, the agent cannot move through an object that does not change openness state, so the interpreted move is impossible. This can occur in situations where the agent's recognizer confuses *to* and *through*. For the *propose alternative plans* strategy, the agent uses the constraint relaxation algorithm described in Section 4.2.2. The agent tries to relax or remove a constraint associated with the referent, then present alternatives. When there is more than one option upon refashioning, the agent will make use of its disambiguation strategies. Those disambiguation strategies will often have multiple choices in which to describe alternatives; this can be decided by either a heuristic model or a learned one. For the *declare object state* strategy, the agent receives an action that in some situations (e.g., moving through a doorway) is possible, but in others is not. The agent will declare the problem as impossible because the state of the object does not align with the requested action.

The remaining impossible-to-execute recovery strategies rely on the agent's *a priori* and real-time mapping information. If the agent receives a request to move to a location that it knows, according to its *a priori* occupancy grid, is not possible to move to, it will declare the move impossible. This *declare planning failure* strategy is triggered by a failure response from the *a priori* $A*$ pathfinding algorithm described in Section 6.4.2. Finally, the *declare obstacle interference* strategy can occur in real time only when the agent is already executing a task. According to the agent's surroundings when it received the request, the task was executable. However, the agent detects a change in the enviornment or an otherwise unknown object in real time that prevents task completion. The agent will declare the plan as impossible due to the unforeseen obstacle. This makes use of the agent's $A*$ path planner that operates over SLAM mapping data accumulated during the session. This is described in Section 6.4.2 as a source of dynamic context information, as opposed to the static *a priori* occupancy grid.

### 7.2.4 Implementation

We implemented this as part of a `Library Agency` in RavenClaw. The dialogue agent will present either the problem only or the problem and a list of alternatives. If possible, the agent will present any of the problematic phrases in the instruction

to give the user a chance to realize a possible self-made error and/or amend the instruction.

## 7.3 Chapter Summary

This work helped us get a sense of the distribution of questions direction followers asked in navigation tasks. As shown in Table 7.1, referential questions were the most common question type in the SCARE corpus, followed by action questions. Although the domain of the task was navigation, people generally asked questions about landmarks rather than general spaces in the world. Participants were not given a sense of scale in the environment, so heavy landmark use was to be expected, according to our presentation formats study. We note that more than half of the questions only required a yes or no answer, suggesting that the direction followers had a preference for crafting questions that required only simple answers. These factors impacted the design of recovery strategies for this thesis. This corpus provided examples of problems we can expect to encounter in physically situated dialogue with agents.

This chapter also described the recovery strategies defined for this thesis. We implemented these by using observations from human-human dialogue, a crowdsourced data collection study, and the agent's representation of physically situated context. The strategies were implemented as a task-independent Library Agent in RavenClaw.

Chapter 8

# Learning to Improve Recovery Strategy Decisions

Dialogue systems can improve how they select recovery strategies over time using previous situated interactions (i.e., dialogue history). However, data collection with physically situated agents is difficult to parallelize; contextual features vary greatly based on the situation. Agents need to build a history one situation at a time. The scale of data is therefore much lower than what can be collected with mainstream dialogue systems, like those in commercial settings. Over time these agents accumulate successful situations. This chapter describes an approach to improving recovery strategy selection using *instance-based learning* [92], where all of the agent's previous successful situations serve as training examples. These examples essentially represent memories of the agent getting into, then out of trouble. When the agent detects a situated grounding problem, it searches its history for similar situations to see how it resolved that problem in the past.

## 8.1 Instance-Based Learning for Recovery Selection

One goal of this thesis is to develop a method by which an agent learns to improve how it selects recovery strategies over time. The learning task is classification; given a new observation (i.e., situated grounding problem), it must select one of multiple possible recovery strategies. Many data-driven learning methods train a model using data collected beforehand (i.e., *eager* learning methods [92]). These

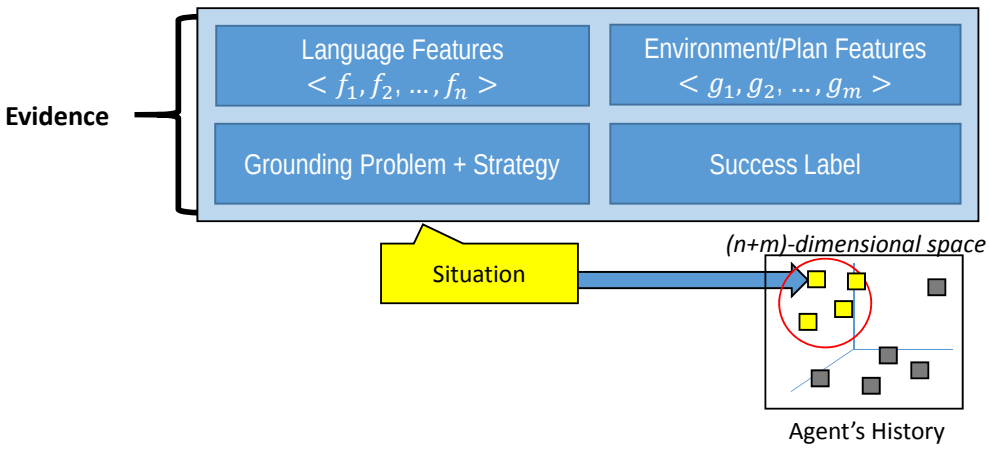



Figure 8.1: With the $k$-nearest neighbors method, we can cast a numeric representation of past dialogue recoveries as retrievable memories. The learning algorithm will continue to add new situations to the training data. The closest successfully-resolved situations to the current observation represent neighbors.

approaches demand a great deal of training data to operate successfully. Physically situated dialogue is costly to collect and difficult to parallelize, so we turned to a different approach. In this thesis, we evaluated *lazy* learning, a method that delays learning until the agent receives a new observation. When a new observation is made, a lazy learning algorithm queries its instances to retrieve a class label. We used $k$-nearest neighbors learning to select recovery strategies when the agent faced a choice [94]. In this approach, the most common recovery strategy used among the nearest $k$ instances of similar situations is selected. The general approach is illustrated in Figure 8.1.

With smaller datasets, an eager learning model would need to periodically be retrained in order to to best represent the structure of new gold standard observations. While eager learning is useful for problems where large amounts of data capture variation (e.g., thousands of interactions with commercial phone-based spoken dialogue systems), we believe it is a less desirable option for physically situated interaction. These interactions are more costly to collect than non-situated systems. On the other hand, "lazy" learning methods like instance-based learning can adapt to variations as often as they observe new examples [92]. Such methods store all past examples as training data and delay the generalization task until a new observation is made. To initiate the learning process, we used the crowdsourced

data collected in Chapter 5. This process allows new dialogues to bootstrap the crowdsourced data.

We formally state this problem as a classification task: *Given a history of past situated interactions, and the feature vector $A$ of the current situation* (features are presented in Table 8.1), *classify the situation and its grounding problem into one of a set of recovery strategies. Identify the closest matching past situations (i.e., "neighbors") that resulted in success. For those neighbors to the current situation, tabulate the recovery strategy that the agent used among a set $S$.* The agent evaluates the neighbors to the current situation to select an appropriate recovery strategy. Since this reasoning happens directly after detecting a grounding problem, it also needs to check the feasibility of the $n$ best strategies selected by the learning algorithm. This requires monitoring the current context stored in the agent's representation of the world, both dynamic and static. Selection is also tied to the concept under discussion in RavenClaw; we focus on grounding object references mentioned during the dialogue. For example, in the navigation task "go to the door," situated reasoning for the concept "door" selects a recovery strategy.

### 8.1.1 Exploring Features for Selection

In situated dialogue, if an agent is aware of its surroundings, it will be able to learn from misunderstandings associated with them. We developed a feature set that combines evidence from the agent's surroundings, the content of the agent's plans (i.e., action sequences), and the content of spoken language commands that the agent receives. By bringing together these sources of evidence, we predict that agents will monitor their situations and determine more effective strategies to recover as they make new observations.

In [82], we identified several indicators that lead to grounding problems by examining dialogue corpora that relate to navigation (the HCRC Map Task [2] and SCARE [128] dialogue corpora). We expanded on that list for this thesis. The agent uses these indicators to select recovery strategies for observed situations. The feature list can be found in Table 8.1.

### 8.1.2 Fast, Approximate Nearest Neighbors

$k$-nearest neighbors ($k$-NN) classification is an instance-based machine learning algorithm that has shown to be very effective for a variety of problem domains

where underlying densities are not known [46]. Numerically representing situations in the physical space demands tracking many features. Since our data has a high number of dimensions, the traditional $k$-NN algorithm can be too slow as the data collected scales. We selected the FLANN toolkit; a fast, approximate approach that scales and does not impact response time to real-time dialogue [94]. FLANN is also one of the most widely used nearest neighbors libraries, included as part of the OpenCV, ROS (Robot Operating System), and PCL (Point Cloud Library) libraries. We structure a situation as the set of features described in Table 8.1.

Since every physically situated interaction can be costly, the $k$-NN approach enables every new learned situation to be added to the training data as soon as the agent determines success or failure. In situations where the agent must select a recovery strategy, the user's verbal responses to its selection permit implicit labeling. Successful situations are those that ground a single referent, while unsuccessful ones are any that result in task failure or a continuing inability to ground a referent.

The Situated Reasoner, described in Section 6.4, uses the multiple randomized K-d tree[1] variant of the $k$-nearest neighbors (k-NN) algorithm to learn from past examples [94]. The $k$-nearest neighbors algorithm casts training examples as datapoints on a $y$-dimensional Euclidean space, where $y$ is the size of a feature vector $A$. The feature vector represents the characteristics of a situation we wish to use as a query. Upon execution the learning algorithm finds matches that are numerically similar to the current situation.

For a new situation $x$, its feature vector is described by

$$A =< a_1(x), a_2(x), \cdots, a_y(x) >$$

where $a_r(x)$ is the value associated with the $r$th feature in situation $x$ [92]. We define an situation as a training example that begins with an instruction from the user, contains a detected situated grounding problem, and ends with a label for task success or failure. We separated training examples based on the situated grounding problem type, either ambiguous or impossible.

[1] We used the four K-d trees, the default number provided in FLANN.

Feature types are denoted as $C = countable$, $R = real$, $B = boolean$.

Table 8.1: Features for numerically representing situations with the instance-based learning method.

| Indicator | Type | Feature Description |
|---|---|---|
| Utterance Features | | |
| word_num | C | the number of words in the utterance |
| sentence_num | C | the number of sentences in the utterance |
| ref_prop_num | C | the number of situated properties in only the referent of the utterance (e.g., *"big box by the red bed"* has only one for *"big"*) |
| situated_prop_num | C | the number of situated properties in the utterance (e.g., *"big box by the red bed"* also counts *"red"*) |
| asr_conf | R | the confidence score for the top recognition hypothesis |
| parse_coverage | C | ratio of uncovered parse fragments to covered parse content |
| num_parse_frag | C | number of parse fragments |
| Discourse Features | | |
| conflicts_exist | B | true if the agent detects an existing information conflict |
| turn_number | C | the turn number |
| num_plan_matches | C | number of possible plans that match the current utterance. executable plans have a value of 1, impossible have a value of 0, ambiguous have a value $\geq 2$. |
| num_actions | C | number of actions obtained from the current utterance |
| num_objects_instr | C | number of objects mentioned in the current utterance |
| num_impossible_steps | C | number of impossible steps in the current utterance. with 0 possible plans, the value is 1. |
| Environment Features | | |
| dist_nearest_match | R | distance in meters to the nearest matching object (if any) |
| dist_to_goal | R | distance in meters if there is only one possible goal. if there is more than one, value is 0. |
| num_matching_obj_instr | C | number of visible objects that match the referent |

| Environment Features (continued) | | |
|---|---|---|
| num_matching_obj_right | C | number of visible objects that match the referent and are on the agent's right |
| num_matching_obj_left | C | number of visible objects that match the referent and are on the agent's left |
| num_matching_obj_front | C | number of visible objects that match the referent and are in front of the agent |
| num_matching_obj_behind | C | number of visible objects that match the referent and are behind the agent |
| num_obj_right | C | number of visible objects that are on the agent's right |
| num_obj_left | C | number of visible objects that are on the agent's left |
| num_obj_front | C | number of visible objects that are in front of the agent |
| num_obj_behind | C | number of visible objects that are behind the agent |
| num_matching_blue_obj | C | number of visible blue objects that match the referent |
| num_matching_brown_obj | C | number of visible brown objects that match the referent |
| num_matching_orange_obj | C | number of visible orange objects that match the referent |
| num_matching_black_obj | C | number of visible black objects that match the referent |
| num_matching_purple_obj | C | number of visible purple objects that match the referent |
| num_matching_red_obj | C | number of visible red objects that match the referent |
| num_matching_gray_obj | C | number of visible gray objects that match the referent |
| num_matching_white_obj | C | number of visible white objects that match the referent |
| num_matching_green_obj | C | number of visible green objects that match the referent |
| num_blue_obj | C | number of visible blue objects |
| num_brown_obj | C | number of visible brown objects |
| num_orange_obj | C | number of visible orange objects |
| num_black_obj | C | number of visible black objects |
| num_purple_obj | C | number of visible purple objects |
| num_red_obj | C | number of visible red objects |
| num_gray_obj | C | number of visible gray objects |
| num_white_obj | C | number of visible white objects |
| num_green_obj | C | number of visible green objects |

| Environment Features (continued) | | |
|---|---|---|
| num_previously_mentioned_objects | C | number of objects mentioned previously. in this thesis this feature was held constant. |
| num_matching_big_obj | C | number of visible big objects that match the referent |
| num_matching_small_obj | C | number of visible small objects that match the referent |
| num_matching_long_obj | C | number of visible long objects that match the referent |
| num_matching_square_obj | C | number of visible square objects that match the referent |
| num_matching_round_obj | C | number of visible round objects that match the referent |
| num_matching_pointy_obj | C | number of visible pointy objects that match the referent |
| num_matching_straight_obj | C | number of visible straight objects that match the referent |
| num_matching_low_obj | C | number of visible low objects that match the referent |
| num_big_obj | C | number of visible big objects |
| num_small_obj | C | number of visible small objects |
| num_long_obj | C | number of visible long objects |
| num_square_obj | C | number of visible square objects |
| num_round_obj | C | number of visible round objects |
| num_pointy_obj | C | number of visible pointy objects |
| num_straight_obj | C | number of visible straight objects |
| num_low_obj | C | number of visible low objects |
| num_obj_near_obj | C | number of objects near other objects (*near* as defined using the proximity algorithm in Section 6.4.4) |
| num_obj_near_ref | C | number of objects near possible referents |
| num_nearby_doors | C | number of doors nearby the agent |

The feature vector contains information from the environment context and the user's spoken language input. In this thesis, the feature vector used the features summarized in Table 8.1 to find neighbors of the current situation. Central to this model is $E$, the current set of environment beliefs and plan information ($E = < g_1, g_2, \cdots, g_m >$ where $m$ is the number of features extracted from the agent's surroundings). $R$, features from a user's spoken language input, is included, where $R$ is composed of features from user's language input ($R = < f_1, f_2, \cdots, f_n >$ where $n$ is the number of features observed).

Countable and real-valued features had their values directly included as part of the feature vector. Boolean features were mapped to $0$ or $1$ based on whether some specific condition was true. Combined these features serve as the "coordinates" for each situation. The algorithm expands the search if no feasible strategies are found in the first $k$-NN query. Given the number of features we included in our model (over 60 features), approximate $k$-nearest neighbors is a suitable approach.

The learning algorithm uses a variant of the distance formula to compute the similarity between the current situation and past ones. Given the feature vector $A$ above of each situation (current and past), the learning algorithm determines the most common $k$ neighbors to the current one by computing the distance $d(x_i, x_j)$ between the current situation $x_i$ and past situation $x_j$, where

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

$a_r(x)$ is the value of the $r$th feature in $x$. The algorithm approximates $k$-NN by constructing several randomized K-d trees and searching them in parallel. A K-d tree is a form of binary trees where each node is a K-dimensional point [122]. The value of K is equivalent to the number of dimensions that datapoints have, or $y$, the dimension of the Euclidean space. The K-d tree approximation breaks up the search space into the dimensions with the highest variance. This allows the algorithm to converge on neighbors quickly (i.e., in real-time agent interactions, the algorithm does not noticeably increase latency).

### 8.1.3 Preparing Initial Training Data

The $k$-nearest neighbor learning algorithm takes in a set of features describing a situation, and determines the $k$ most similar situations to the current one in

| Training Subset | Egocentric Proximity | Intrinsic: Color | Intrinsic: Size | Object Proximity | Total # Examples |
|---|---|---|---|---|---|
| Ambiguous | 85 | 103 | 64 | 27 | 228 |
| Impossible | 28 | 24 | 4 | 4 | 49 |

Table 8.2: Initial training data totals for referentially ambiguous and impossible situations by situated property. A single training example could have multiple situated properties.

the training data. The initial training data set consisted of a subset of responses collected from the crowdsourced study in Chapter 5. Since the responses collected from the study were typed recovery strategies, an annotator generalized them by labeling the responses with which situated properties they contained. Given a response like *Did you mean the blue or brown box?*, the class label would be INTRINSIC:COLOR. Section 5.3.3 describes the *situated dimension* annotation. These annotations served as the class labels for each of the situation types developed in that study. The dimension types were intrinsic (color or size), object proximity, and egocentric proximity.

The training data were for two types of problems: referentially ambiguous instructions and those that were impossible-to-execute. We only included responses where crowdsourced workers typed a response appropriate for that situation (i.e., correct responses). We also removed the responses that correctly detected a problem, but did not provide any useful information (e.g., *I see more than one. Which did you mean?*) because there was no class label. This resulted in removing 22 entries from the ambiguous training set and 103 entries from the impossible-to-execute training set. This amount was larger because several situations did not allow for mentioning of situated properties, like the impossible path situations.

For each of the situations, we developed functions for the agent to automatically populate the feature vectors. Populating the feature vector consisted of running the agent through all the situations that were used in the previous study. We made the simplifying assumption that speech recognition was perfect (i.e., the feature for speech recognition confidence was 1.0, the feature for parse coverage was 1.0, and the feature for parse fragments was 0.0). This was reasonable for the crowdsourced dataset because the participants authoring strategies replayed a synthesized speech command and could not proceed with the task without understanding the language.

The crowdsourced study had fifteen participants provide recovery strategies

| $k$ value | Consistent Voting Count (Unanimous or Majority) | Inconsistent Voting Count (Plurality or Tie) |
|---|---|---|
| $k = 3$ | 16 (7 UN, 9 MAJ) | 9 (5 PLU, 4 TIE) |
| $k = 5$ | 22 (2 UN, 20 MAJ) | 3 (2 PLU, 1 TIE) |
| $k = 7$ | 21 (21 MAJ) | 4 (3 PLU, 1 TIE) |
| $k = 9$ | 19 (19 MAJ) | 6 (5 PLU, 1 TIE) |
| $k = 11$ | 20 (20 MAJ) | 5 (4 PLU, 1 TIE) |

Table 8.3: We determined a starting value for $k$ by determining voting performance. Voting was either unanimous (UN), a majority (MAJ), a plurality (PLU), or a tie (TIE). We evaluated the popularity of the class with the top votes for each example in the development set.

for each of fifty situations. As such, the training data consisted of fifteen identical feature vectors for each situation, except the class label was that of each crowdsourced worker's response, which varied. In total this resulted in 228 training data entries for the 25 ambiguous situations and 49 training data entries for 11 impossible situations (see Table 8.2). Stratified[2] subsets containing a third of each crowdsourced dataset were withheld as test sets for evaluation.

Each training data set was divided into two files: a file containing the feature vectors and a file with the corresponding annotated class labels. The feature vector file was encoded as an HDF5 (hierarchical data format) data table with one row for each training data entry. Each row in the class label file directly corresponded to the row in the feature vector file.

### 8.1.4 Selecting the Value for $k$

The value of $k$ determines how many neighbors to a new observation can be used to determine its class label. The class label for each of the $k$ neighbors is used to determine the new observation's class. This amounts to votes for what the class label should be. At a minimum, the value of $k$ must be odd to mitigate ties. We experimented with odd values of $k$ until we observed diminishing performance on a development set. Performance was measured by how consistently neighbors would agree on a class label. The development set consisted of twenty-five examples, one for each scenario in the ambiguous training set.

[2]Subsets were stratified by class label. This resulted in a similar distribution of situated properties in both the training and test sets.

We ran the $k$-NN algorithm with the development set for values of $k$ starting from $k = 3$. Table 8.3 presents these results. We directly observed the neighbors selected by the learning algorithm, and tabulated the voting performance for each value of $k$. A given development data entry with $k$ neighbors would result in votes for class labels selected by each neighbor. Multiple class labels from each neighbor were possible because recovery strategies could include multiple situated properties. Performance was good if the neighbors resulted in consistent voting, that is, a *unanimous*-determined class label or a *majority*-determined class label. Less desirable results would be inconsistent voting, that is, either a *plurality*-determined class label (i.e., the most common class was not a majority) or a *tie*. The best performing values of $k$ using this metric were $k = 5$ and $k = 7$. Both had similar voting traits that maximized majority voting and had few conflicts. Performance began to degrade with greater values of $k$; this may be due to training data sparsity causing fewer majority votes to happen. In the end, we selected $k = 7$ because it leveraged more data when making a decision.

Occasionally, $k = 7$ may not be a reasonable choice, such as when ties occur or the class label selected is not contextually appropriate. Contextually inappropriate classes are situated properties that are not distinctive in the current situation. Consider the example where the learning algorithm decides that for the incoming ambiguous input *"Move to the box"*, color is the selected label. However, in the agent's environment, color is not a distinctive property. This would result in a response like *"Do you mean the brown box or the brown box?"* and would lead to task failure. The learning algorithm expands the search when this happens by increasing the value of $k$ to the next-highest odd number and running the search again. This process repeats until a valid class is found. The same goes for ties: When the top $n$ class labels are tied with votes (which can occur since each data point can have multiple labels) the search runs again with the next-highest odd value of $k$.

## 8.2 Real-Time Experimentation with Recovery Learning

One method for evaluating the $k$-NN learning algorithm is to observe how the agent's selection of recovery strategies changes as it accumulates examples in real time. We conducted a longitudinal study with this goal in mind. Six participants

interacted over six days with TeamTalk, the human-robot dialogue system used in this thesis. The focus of this study was to better understand how dialogue systems for physically situated agents might recover from errors that occur over the course of a back-and-forth dialogue with people. We observed how the agent's performance changed over time with users. In this evaluation, we evaluated the agent's ability to recover from miscommunication. The study consisted of dialogue between a person giving directions and a virtual robot (represented on a computer screen) following those directions.

### 8.2.1 Method

#### Purpose

The purpose of the study was to understand how spoken dialogue systems for physically situated agents could better recover from communication problems during interactions with people. Dialogues between people and an agent were stored by the agent as memories. These memories represented positive or negative experiences of the agent recovering from errors as they occurred during the dialogue. The learning algorithm added positive experiences to its training data in an online fashion.

#### Setup

Participants viewed the agent on a computer screen. It was rendered as a robot in a virtual environment (see Figure 8.2). The only way for participants to interact with the agent was by speaking into a headset microphone. The experimenter provided the participant a task list of where the agent needed to go in the environment.

The user interface displayed a visual of the agent in the virtual environment (the agent was centered on the screen). In order to align with experiment goals, the agent's internal map was different from what appeared on the user interface. This difference in environment understanding resulted in miscommunication between the participant and agent. There was feedback and clarification questions from the agent using Cepstral speech synthesis.

#### Procedure

Upon arrival at the study, the task, virtual robot, and user interface were explained. The primary task was reconnaissance and exploration with a limited understanding

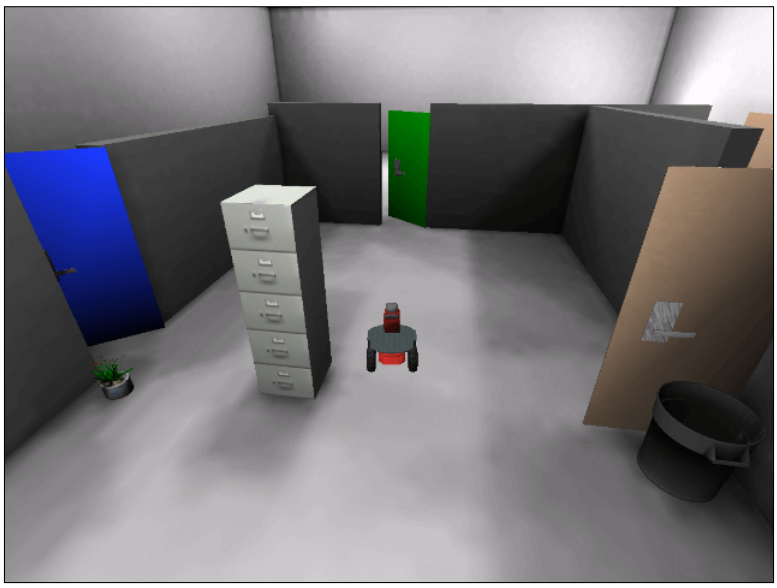


Figure 8.2: An example referential ambiguity trial where the task was to move to *"door"*. In red is the agent, rendered as a robot (*centered*). Participants would speak an ambiguous command because their task list required using the underspecified referent *"door"*.

of the environment. Participants received an overview of the area with last known information on *objects of interest*. The objects needed to be verified as existing or not present. Participants saw the agent as it moved around on-screen. An additional experimenter assisted in the setup of the system. Between trials, the participant remained with the experimenter. The script for this study can be found in the Appendix.

When participants interacted with the agent for the first time, they participated in one practice scenario with five tasks. The purpose of this was to mitigate any possible learning effects. Participants could continue the training scenario until they were satisfied with how they could interact with the agent. Following the practice session and in all future sessions, there were five scenarios; each scenario had five trials. Each trial required the participant to provide instructions that would move the agent to a goal location. Some of the situations were impossible or referentially ambiguous. This was because the participant relied on an older version of the map in order to navigate the agent through the environment. Participants viewed a rendering of the agent in this older map (described to the participant as being collected hours ago). Meanwhile, the agent had the latest map (internally, hidden

from the participant), which on occasion differed from the older map.

The two types of problems in this evaluation were referential ambiguities (e.g., *"Go to the box"* when the agent could go to multiple boxes) and impossible tasks (e.g., *"Go through the door"* when the door was closed). In trials with referential ambiguities, the agent's internal map had more objects of a relevant type than the participant's map. In trials with impossible tasks, the agent's internal map did not have objects of the relevant type, or presented obstructions that blocked a path to the goal destination. Participants could always see the agent's (outdated) surroundings during the dialogue exchange. The scenarios in this study were very similar to those in the crowdsourced study presented in Chapter 5, but were altered to improve usability with the dialogue system. Figure 8.2 presents one such trial.

In trials with referential ambiguity or impossible tasks, the agent responded with a question that included some information and an attempt to recover. We predicted that the participant's response would have more information and expand on the initial instruction based on the context of the conversation. At this moment in the interaction, the participant provided a verbal response. This created a multi-turn dialogue that either ended with the agent going to the goal location, or failing to do so. For these cases, multiple sources of knowledge could be used to initiate a recovery strategy. As such, we did not know there to be a "best" recovery strategy for any scenario beforehand. The purpose of this design was to demonstrate that the recovery strategy selection component would improve selecting recovery strategies as the agent logged more interactions.

Since scenarios were composed of multiple trials, the participant needed to successfully move the agent to the next trial, or subgoal, to proceed in the task. In the case of trials that completely broke down (i.e., dialogues that completely failed after several attempts), we noted these task failures and proceeded to the next trial in the scenario. If there were communication breakdowns (i.e., repeated task failures), the experimenter could remind participants about how to speak into the microphone or the types of commands that were possible.

Participants each repeated the scenarios in different orders according to a Latin Square. Completing all ten scenarios was completed in two phases. For a given experiment run, participants would need to complete five scenarios (i.e., the first phase). The remaining scenarios would be withheld until the next session (i.e., the second phase). This was presented to participants as "another chance" to go through the environments with the agent.

### Experiment Design

The only experiment condition for this study was how much training data that the agent used with the $k$-NN learning algorithm. Three participants interacted with an agent configured to use all data collected from all participants (*general model*), while the remaining three participants interacted with an agent configured to only store data collected with that participant (*user-specific model*).

### Measures

The performance of the agent's learning algorithm was assessed as part of the greater TeamTalk dialogue system. The key independent variable in this study was the type of learning model, *general model* or *user-specific model*. Dependent variables were the measures we describe below. We measured variation across *phases*, that is, each full pass through all ten scenarios. For example, *Phase 2* represents the second pass of each participant's execution of all trials for the study (50 trials across ten scenarios).

*Number of failed recoveries:* For an experiment trial (referential ambiguity or impossible task completion), any attempt by the agent to select a recovery strategy that did not result in successful completion of the task. Recovery strategies associated with the learning algorithm were only those that could include some feature about the surroundings to establish context (e.g., color to disambiguate a referent). The agent should learn to favor successful recoveries over time. We hypothesize that this value will decrease as the agent engages in more dialogues. Moreover we expect this to hold for both the general and user-specific models.

*Number of dialogue turns:* The number of dialogue turns tabulated between the participant and agent to complete a trial. A change in conversational floor incremented the dialogue turn count. Turns were only counted within a trial and precisely from when the participant's task-related instruction was correctly recognized by the agent. This allowed us to focus on situated dialogue recovery as opposed to speech recognition repairs. We hypothesize that the number of dialogue turns taken to complete experiment trials will also decrease as the agent engages in more dialogues.

*Time on task:* The time from when the agent first detects the participant's task-

| Learning Model Type | Phase | # Failed Recoveries | # Dialogue Turns | Time on Task (s) |
|---|---|---|---|---|
| General | Phase 1 | 29 | 506 | 2643 |
| | Phase 2 | 29 | 480 | 2363 |
| | Phase 3 | 28 | 452 | 2324 |
| User-Specific | Phase 1 | 38 | 522 | 3004 |
| | Phase 2 | 14 | 424 | 2144 |
| | Phase 3 | 10 | 408 | 2228 |

Table 8.4: Measures recorded by learning model type.

related instruction as problematic to when the task gets successfully completed or abandoned. We hypothesize that the time on task will decrease as the agent engages in more dialogues.

### 8.2.2 Participation

We recruited six volunteers for this study. They interacted with the agent for a period of approximately thirty minutes per day for six days.

### 8.2.3 Results

We compared the two model types, general and user-specific, using measures that assessed the agent's performance changes over time. While the measures were collected for every trial in the experiment, we present results with only the trials that required use of the learning algorithm. These were all of the ambiguous trials (twenty-five in total) and nine of the impossible-to-execute trials (the remaining six impossible-to-execute trials were ones where the path was not possible and did not use the learning algorithm). This resulted in 612 spoken dialogue trials across six participants with the agent. Measurements by phase are reported in Table 8.4. The user-specific models initially performed worse, as measured by higher counts for failed recoveries, dialogue turns and time on task in *Phase 1*. In the end user-specific outperformed the general model – by *Phase 3*, the user-specific models had fewer failed recoveries, fewer dialogue turns, and faster time on task than the general model.

For all measures, we analyzed them by *phase* only (a repeated measure), by *learning model type* only, and by an interaction of the two. We used a factorial

mixed-effects analysis of variance model[3] for each of these situations, with phase and learning model type counted as fixed effects (phase, learning model type, and the interaction between them). Participant, nested by learning model type, was included as a random effect in the model.

### Failed Recoveries

Table 8.4 shows that the number of failed recoveries remained about the same for the general model, while those for the user-specific model showed improvement between phases. We predicted that the number of failed recoveries by the agent would decrease between phases for both models; the results only support this hypothesis for the user-specific model. When considering only the data from from the user-specific model condition, phase had a significant main effect (F[2, 274] = 5.8, $p < 0.005$). The number of failed recoveries decreased most between the first phase of the trial and the second, but decreased again in the third phase. A Tukey HSD post-hoc test revealed that the *Phase 1* measure was significantly different with *Phase 2* ($p < 0.05$) and *Phase 3* ($p < 0.01$). The interaction *phase * learning model type* did not have a main effect (F[2, 548] = 1.8, p = 0.16). Phase also did not have a significant main effect (F[2, 548] = 2.0, p = 0.14). In addition, a Tukey post-hoc test did not reveal a significant difference.

### Dialogue Turns

The number of dialogue turns decreased between phases for both the general and user-specific models, as presented in Table 8.4. Originally we predicted that this value would decrease between phases, as the agent engaged in more dialogues for both learning model types. Results support this hypothesis. Phase had a significant main effect for results with both general and user-specific models (F[2, 548] = 5.1, $p < 0.01$). The number of dialogue turns decreased significantly between phases 1 and 3 of the study ($p < 0.01$; Tukey HSD). A Tukey HSD post-hoc test showed a significant difference between *Phase 1* and *Phase 3* with the user-specific model results ($p < 0.05$). Again analyzing only the user-specific data, there was a significant main effect for phase (F[2, 82] = 6.0, $p < 0.005$). *Phase 1* had more turns than *Phase 2* ($p < 0.05$; Tukey HSD) and *Phase 3*

[3]This approach computed standard least squares regression using reduced maximum likelihood (REML) [49].

($p < 0.005$; Tukey HSD). The interaction *phase * learning model type* was not significant (F[2, 548] = 1.0, p = 0.37).

**Time on Task**

There was a decrease in trial duration between most phases for both general and user-specific models (see Table 8.4). We predicted that time on task would decrease between phases as the agent collected more dialogues, for both learning model types. Results support this hypothesis. Phase had a significant main effect for results with both general and user-specific models (F[2, 548] = 6.9, $p < 0.005$). There was a significant decrease in time on task between *Phase 1* and *Phase 2* ($p < 0.005$; Tukey HSD test) and *Phase 1* and *Phase 3* ($p < 0.005$; Tukey HSD test). The interaction *phase * learning model type* was not significant (F[2, 548] = 1.6, $p = 0.21$). A Tukey HSD post-hoc analysis did reveal a significant difference between *phase1 * user-specific* and *phase2 * user-specific* ($p < 0.01$), and between *phase1 * user-specific* and *phase3 * user-specific* ($p < 0.05$). When analyzing only the user-specific data, phase had a significant main effect (F[2, 274] = 8.3, $p < 0.0005$), again with *Phase 1* having significantly longer time on task than *Phase 2* ($p < 0.001$; Tukey HSD) and *Phase 3* ($p < 0.005$; Tukey HSD).

### 8.2.4 Discussion

The results generally support the hypotheses, with the results for user-specific models better representing our predictions. They also demonstrate that the agent improves in performance over time after engaging in new dialogues. The general model and user-specific models update each time a new observation is made that indicates success or failure. This can happen multiple times per dialogue.

All participants experienced similar agent behavior, but participants assigned to the general model condition were interacting with an agent that was learning from all observed dialogues. The first participant in the general model condition started out interacting with a learning algorithm that was only trained with crowdsourced data. The next participant interacted with an agent whose model included the previous participant's interactions. At the start, the user-specific models all had the same training data for each participant – these models would only update with the participant using it.

User-specific models tended to result in stronger performance improvements over time compared to the general model that learned from all observations. We

| Learning Model Type | User | Phase | # Failed Recoveries | # Dialogue Turns | Time on Task (s) |
|---|---|---|---|---|---|
| General | G-1 | Phase 1 | 2 | 136 | 647 |
| | | Phase 2 | 6 | 146 | 650 |
| | | Phase 3 | 0 | 128 | 648 |
| | G-2 | Phase 1 | 9 | 182 | 940 |
| | | Phase 2 | 12 | 178 | 898 |
| | | Phase 3 | 22 | 176 | 953 |
| | G-3 | Phase 1 | 18 | 188 | 1055 |
| | | Phase 2 | 11 | 156 | 814 |
| | | Phase 3 | 6 | 148 | 723 |
| User-Specific | U-1 | Phase 1 | 14 | 180 | 1023 |
| | | Phase 2 | 1 | 128 | 670 |
| | | Phase 3 | 5 | 150 | 795 |
| | U-2 | Phase 1 | 9 | 146 | 925 |
| | | Phase 2 | 8 | 156 | 770 |
| | | Phase 3 | 1 | 128 | 693 |
| | U-3 | Phase 1 | 15 | 196 | 1056 |
| | | Phase 2 | 5 | 140 | 703 |
| | | Phase 3 | 4 | 130 | 740 |

Table 8.5: Measures recorded by learning model type and user.

believe that this is because adapting to user preferences about environment content yields more successful dialogues; preferences can vary from user to user. Moreover a specific user's preferences may differ from both the general model and the initial training data developed from crowdsourced recovery strategies of similar situations. This can impact the pace and progress of learning. We reflect on the results for each measure below.

### Failed Recoveries

Results show that the overall number of failed recoveries decreases between phases, but not in a statistically significant way. There are also no interaction effects between learning model type and phase, which suggests that the learning models are not performing much differently over time. However, a post-hoc evaluation shows that the learning algorithm makes significant improvements between the first and later phases for the user-specific condition. Moreover, results for the user-specific condition show significant performance improvements. A user-by-user in-depth look at the results, as shown in Table 8.5, shows that with the exception of one participant, the number of failed recoveries improves from the first phase to the last.

Results on the failed recovery measure differ when comparing the first phase of the general model to those of the user-specific models. In Table 8.4, there is a stark contrast in the number of failed recoveries for the general model compared to the user-specific ones; the user-specific ones have nine more failed recoveries (31% more). We attribute this in part because of user-to-user variance; the first and third users in Table 8.5 have the second and third-highest totals of failed recoveries in *Phase 1*. The models also collect observations differently. The general model accumulates observations of all participants in an online fashion; in the first phase, all users except the very first interact with an agent that has experience with real-time recovery. Alternatively, the user-specific learning models all begin with only the crowdsourced training data. In effect, the user-specific models adapt from the initial data for each user. We believe this adaptation impacts performance. The general model always has a larger training data size than the user-specific ones because it includes observations of all users.

The results also suggest that user-specific models are preferable when the nature of the interaction is longitudinal, such as with trained operators that are in long-term interactions with a particular agent. When the learning algorithm is collecting data about all users, it may negatively impact performance with people whose preferences deviate from the norm. For example, results from the crowdsourced study in Chapter 5 suggest a general preference for visual information, but that may not work well for someone who is colorblind. Someone with a high spatial ability may prefer to use egocentric spatial terms like right and left to differentiate possible referents.

### Dialogue Turns

Results support our prediction that the number of dialogue turns would decrease between phases for both learning model types. Phase had a significant main effect. The learning algorithm's recovery strategy selections make dialogues more efficient as it observes which strategies work better in different contexts. We note that there are no significant interactions, which suggests that neither learning model type was optimizing better than the other given the sample size observed.

The user-specific models have the strongest performance improvements. There is a statistically significant reduction in dialogue turns from the first to second phases. The second and third phases are not significantly different; results suggest that the learning algorithm converged on an improved strategy set for users by the

second phase. Table 8.5 shows substantial reductions in dialogue turns for two of the three participants from the first to the second phase. The other participant's model may have needed more dialogues to improve, as results end up improving by the third phase. Although one of the users had a model that degraded in performance between the second and third phases, it was still an improvement over the first phase.

#### Time on Task

Results support our prediction that time on task would decrease between phases. We found that phase again had a significant main effect. This adds to the evidence that the learning algorithm is making dialogues more efficient with its recovery strategy choices, and that it is benefiting from new observations. Similar to dialogue turn count, we did not find any significant interactions between phase and learning model type, suggesting that neither learning model type is optimizing better than the other. We echo observations made in the failed recoveries discussion when considering the initial differences in performances between the two model types (see Table 8.4). As before, we attribute the differences to user variance and the differences in training data size between the general and user models.

Post-hoc evaluations support the existing evidence that the user-specific model is performing better between phases. The results are similar to those of dialogue turn count. The user-specific model has a statistically significant reduction in dialogue turns from the first to the second phase. The second and third phases were again not significantly different; we expect that converging may have occurred. The user-by-user table (Table 8.5) shows that all three participants in the user-specific model group had improvements in time on task between the first and second phases. Combined with measures for dialogue turn count, the results suggest the learning algorithm is making improved choices over time.

### 8.2.5 Limitations

As with all repeated measures experiments, there is a possibility of a learning effect. Each phase represents one full pass through all ten scenarios, five trials per scenario. However, we note several measures we put into place to mitigate possible learning effects. First, we conducted a series of practice trials through a scenario that was separate of the experiment trials. This scenario had examples of both referentially ambiguous and impossible-to-execute instructions, so participants

knew the general problem space. Participants would only proceed to the main task after they felt comfortable with both the task and speaking with the system. Before starting any scenario, participants saw an overview of the robot's last known surroundings; due to task design these differed from the robot's internal map in some ways. We also note that for each trial, participants knew what to instruct the robot about, and the relevant terms the robot understood for the objects in the scenes. The task completion rate was consistently high. To reduce ordering effects, we counterbalanced the order of scenarios for each participant according to a Latin Square using the same approach presented in Chapter 5. We expected minimal carry-over effects because participants had at least a day before they viewed the same trial. The results clearly indicate adaptation is happening; we believe that a held-out test set evaluation, presented in Section 8.3, will support the hypothesis that the learning algorithm is improving in quality as it collects more dialogue data.

Participant-specific results between phases support our claim that learning effects about the task were minimal; occasionally the measures would degrade on a participant-by-participant basis (see Table 8.5). We expect the degradation in performance is caused by changes in how the model selects recovery strategies. The updates may or may not reflect optimal strategies for a particular participant.

## 8.3 Evaluating Recovery Learning with Crowdsourced Data

We evaluated the learning algorithm with various training data sizes using a held-out test set. This helped us determine if the feature space was appropriate for representing situations. Given the high cost of collecting physically situated interactions, the held-out test set evaluation showed whether an instance-based learning approach was feasible with a limited number of training examples. Upon completion of the study, the learning algorithm collected 605 new training examples, 464 for successfully recovered ambiguous situations, and 141 for impossible ones. We evaluated the learning algorithm's performance at two benchmarks for each problem type, 50% and 100% of this new training data for each problem type. We hypothesize that the learning algorithm's performance will be significantly better after collecting all six days' worth of data compared to half. In both cases the data in common will be the first three days.

We used the crowdsourced test set withheld in Section 8.1.3 to determine if the

learning algorithm was improving over time against human judgments. The crowdsourced data was structured the same as the training data, with rows populating all sixty-two features. We measured the agreement rate of the learning algorithm on the two configurations of training. This was accomplished by observing the number of agreements that the learning algorithm had with annotations of crowdsourced workers' recovery strategies.

### 8.3.1 Method

Given the full set of crowdsourced recovery strategies, we withheld a portion for testing (33% of the original dataset, 131 instances). The original dataset collected from Amazon Mechanical Turk had 409 correctly detected entries; 336 for ambiguous situations and 73 for impossible ones. An example situation could be: ROBOT IS POINTING FORWARD, BLUE BOX ON RIGHT, BROWN BOX ON LEFT, STIMULUS COMMAND IS *"Move to the box"*.

These data were represented the same as the real-time evaluation: a feature vector with sixty-two features that numerically represented the utterance and environment context for a situation, along with a crowdsourced worker's situated properties (i.e., class label) for that situation. For example, if a crowdsourced worker typed, *"Did you mean the box on the left or the right?"* for a situation that had *box* as an ambiguous referent, the class label was annotated as EGOPROX because it used egocentric proximity. In the crowdsourced data, this was the only way to represent such a situation – the same feature vector would have been matched to up to fourteen other crowdsourced workers for that situation. This was because participants saw the same image of the agent and heard the same synthesized command.

In the real-time evaluation in Section 8.2, the learning algorithm combined crowdsourced instances with new ones that it accumulated as the agent interacted with participants. However, since the test data does not contain spoken language features (they were typed recoveries), the learning algorithm may simply favor the crowdsourced data because it kept several language-specific features, like speech recognition confidence, constant. To mitigate the favoring of the crowdsourced data and to assess the learning algorithm's ability to accumulate novel situations, we removed the initial crowdsourced data from the training set. The remaining training data was collected exclusively from spoken dialogue interactions over the course of the real-time evaluation. We report the approximate $k$-nearest neighbors

| Learning Model Type | 50% Training (ambiguous) | 100% Training (ambiguous) | 50% Training (impossible) | 100% Training (impossible) |
|---|---|---|---|---|
| Random | 26% | 26% | 26% | 26% |
| Majority | 44% | 44% | 47% | 61% |
| General | 56% | 66% | 61% | 70% |
| User-Specific *Model 1* | 38% | 48% | 48% | 70% |
| User-Specific *Model 2* | 34% | 55% | 78% | 74% |
| User-Specific *Model 3* | 27% | 47% | 61% | 61% |

Table 8.6: Agreement results on the held-out test set evaluation for general and user-specific models. We also report random and majority baselines of agreement.

algorithm's performance on the test data at 50% and 100% benchmarks.

### 8.3.2 Results

We predicted that the learning algorithm for both the general and user-specific models would yield better agreement rates with more training data. The results confirmed this hypothesis. We made the gold standard assumption that the crowd-sourced worker's preferences for situated properties in recovery strategies were the correct ones.

#### General Model

We report on obtaining class labels for the held-out test set with the general model at 50% and 100% of the data accumulated during the course of the real-time evaluation. Results are summarized in Table 8.6. The general model trained on half of the ambiguous recoveries collected during the study resulted in a 56% agreement rate (60 / 108 situations in agreement). The model trained on all of the ambiguous recoveries resulted in a higher agreement rate of 66% (71 / 108).

Using the model trained on 50% of the impossible recoveries collected during the study, the learning algorithm had an agreement rate of 61% (14 / 23). The model trained on all of the impossible recoveries collected during the study had an agreement rate of 70% (16 / 23).

#### User-Specific Model

The results for performance at 50% and 100% benchmarks are summarized in Table 8.6. For each user model, more training data resulted in improved agreement

rate with the test set data. One exception to this was when the user-specific *Model 2* for impossible recovery already had a strong performance with half of the ambiguous data, and additional data caused it to be one fewer correct. In fact, the same user's model for impossible recovery outperformed the general model trained on all six participants. Ambiguous recovery for any user-specific model did not outperform the general model. The user-specific *Model 2* for ambiguous recovery experienced the greatest improvement. For impossible recovery, the user-specific *Model 1* improved the most. Note that the user-specific *Model 3* for impossible recovery did not improve over time.

### 8.3.3 Discussion

The results support our hypothesis that more data would result in better performance for both general and user-specific models. This suggests that as the models accumulated more situations, they tended to behave closer to crowdsourced workers' preferences. This may validate the general preferences collected from the crowdsourced study as representative of a more general population, as more data meant the models' results aligned better with test set (i.e., gold standard labels). Exceptional cases were two user model conditions where performance either did not improve or became marginally worse (i.e., mislabeling one more data entry).

An improvement in performance over time also validates the learning approach used in this thesis. Using the features we proposed, the learning models tended to align more with crowdsourced workers' preferences for recovery strategies as they collected more situations. The feature set can be used to numerically represent situations in the navigation domain. The learning algorithm used these features to match against new observations.

The general model experienced improvements in agreement rate for both ambiguous recovery and impossible recovery as more training data was added. We claim that the learning algorithm's accumulation of situations resulted in a model that aligned with the preference distribution found in the crowdsourced test data. For ambiguous situations, the general model outperformed all of the user-specific models.

The user-specific models generally improved in performance in time. The user-specific *Model 2* for impossible recovery outperformed its general model counterpart with a fraction of the training data. We speculate that this user could have results that strongly aligned with the general mean of preferences. This

suggests that the learning algorithm's observations for this user most represented the general crowdsourced population in the test data, more so than other participants. There were two instances where performance between the half-set and full-set models did not improve; the models for these situations likely already converged based on the observations from those users. For instance, the third participant's model for impossible recovery always selected egocentric proximity using the half set, and the results did not change in the final set. Preferences may have already been so heavily biased towards egocentric proximity that there was no reason for the model to deviate.

#### Limitations

We assumed that crowdsourced workers' preferences were the "correct" ones in this evaluation. This is not necessarily true; every trial permitted multiple valid responses for recovery using properties of the robot's surroundings. "Correct" in this work at best indicates a general preference of some cross-section of the general population that preferred the situated properties that we observed.

## 8.4 Concluding Remarks

In this chapter, we described and evaluated a method to improve recovery strategy selection using instance-based learning. The agent accumulated situations from dialogue with six human users using $k$-nearest neighbor learning. Each user interacted with the agent over six separate sessions. We evaluated the performance of two different models, (1) a general model that accumulated situations from all six users, and (2) three user-specific models that only accumulated situations from each participant.

Our results show that recovery strategy learning for physically situated dialogue is possible, despite having access to small scales of training data. Training data in physically situated tasks is costly to collect because it requires real-time interactions with an agent that is interpreting many streaming sources of information. The agent relies on features from sources like its spatial orientation, position, and knowledge about its environment in order to to make decisions. The $k$-nearest neighbor method proved to be an effective way to affect the agent's dialogue choices. These choices improved time on task, reduced the number of dialogue turns, and reduced the number of failed attempts that the agent took to recover.

The learning method offers several advantages over heuristics. The $k$-nearest neighbor method is able to immediately add new observations of successful recoveries to its training data in an online fashion. Rapid adaptation was important, as results with user-specific models consistently outperformed the general model in the real-time evaluation. Moreover, the user-specific models observed improvements in performance after only a single pass through all dialogue scenarios. We observed statistically significant improvements between the first two phases (i.e., measurable time periods of interaction) for all performance measures. Additionally, results from the held-out evaluation suggest that individual users may occasionally deviate from general preferences obtained from a larger population. Most user-specific models had poorer results than a general model on held-out test data obtained using crowdsourcing techniques.

A real-time evaluation of a physically situated dialogue agent with users demonstrated that the agent could learn through experience. The $k$-nearest neighbor learning method operates in real time and did not appear to influence dialogue processing latency. While the user-specific model may take slightly more time to begin to show performance improvements, it quickly outpaced the general model. In the first phase of real-time experimentation, performance measures were worse with the user-specific model. Within a full phase of observations, the user-specific models showed performance improvements not seen with the general model. This speaks to the claim that the agent's situations can vary greatly, and that the user's preferences play a role in the success of the strategies that the agent selects. These preferences are important, because accommodating an individual user improves performance measures more than considering all users' situations. While the general model is feasible, we recommend a user-specific model. This is particularly relevant in longitudinal interactions where an agent will interact with a user over an extended period of time (e.g., a smart home agent or domestic robotic assistant).

User-specific models suffer from the *cold start* problem – that is, they may not be able to make decisions without first collecting some data about a user [116]. One solution we explored in this thesis was to collect a crowdsourced dataset of recovery strategies paired with situations. This bootstrapped dataset allowed a general model and user-specific models to function while collecting more training data in an online fashion. In future application settings, a bootstrap dataset could be removed, leaving only training data collected in real time with an agent. If bootstrapping is not possible, the agent could select from one of several strategies at random. The agent could learn from which interactions yielded success and

which caused problems. With this approach, the models would eventually converge to preferences at the cost of more failed recoveries.

If the goal is to have functional user-specific models from the start of interactions with a new user, more work is necessary to address the *cold start* problem. Assigning the user to a dialogue-specific psychographic profile is a possible solution. A psychographic profile could estimate the user's personal preferences on how agents should resolve miscommunication. This could be accomplished by pooling together data obtained from users that had similar preferences. Determining some of these preferences can be collected ahead of a new user's first interactions with an agent. Consider a psychographic profile for someone that self-reports high spatial ability. That user's model could realign preferences such that it prefers to disambiguate referents using egocentric proximity. Similar approaches exist for building psychographic profiles of students [38], mobile phone users [33], and social proxemics with a robot [143]. If parallelizing data collection is possible, multiple agents could share data to build user-specific models [69]. In future work, we hope the research community explores adaptation approaches that consider a mix of both general and user-specific modeling. We believe the approach merits further investigation as a general technique for improving dialogue system performance.

Results show that the learning method can scale, given its performance on a held-out test set of crowdsourced data. We found that more data improves performance in a general model; results vary between user-specific models. The held-out test set represents a *consensus* of recovery strategy preferences that may not necessarily align well with individual users. An individual user could certainly have similar preferences to a crowd of users, and their model would perform well on a held-out test set – but many do not. We also found the feature space to be appropriate for representing situations to the agent. The results showed that the numerical representation harnesses variation between situations. Moreover, the limited number of datapoints (on the order of tens to a few hundred) could make a positive impact on performance. Finally, the user-specific results show that individual users have preferences that can deviate from the crowd. The highest correctness rate for referentially ambiguous recovery was 66% (general model); for impossible recovery this was 74% (a user-specific model).

There is some room for improvement; or at the very least opportunity to better understand how to combine general preferences with that of an individual. In future work, we hope to better understand possible convergence points that can occur. Excessive training data could cause overfitting, which likely caused the

held-out performance of user-specific models to suffer. This could be especially impactful considering the complexity of features in physically situated interactions. We believe there to be a tradeoff between more data and the preferences obtained from a crowd. Better understanding of this could help us determine the true ceiling of recovery strategy selection performance. Selecting the type of model to use requires consideration of the nature of interactions. Some necessary considerations are if interactions will happen frequently with specific users, or if interactions happen infrequently with many different users.

## 8.5 Chapter Summary

In this chapter, we proposed and evaluated an instance-based learning approach to error recovery in physically situated dialogue. We conducted two rounds of evaluation. In one study, we conducted a real-time evaluation where six volunteers spoke with an agent over the course of six sessions. The other study evaluated the learning algorithm on a held-out test set using only the data an agent had collected from a real-time evaluation.

The real-time evaluation resulted in a general preference for user-specific modeling on measures of dialogue turn count, time on task, and the number of failed recoveries selected by the learning algorithm. A held-out test set evaluation with crowdsourced data resulted in better performance with the general model for ambiguous situations and a user-specific model for impossible ones. The agent's performance measures from the real-time evaluation resulted in significant improvements. User-specific models may not have had as high an agreement with the held-out test set as the general model because they represented a single user's dialogues. This includes all of that user's implicit preferences over the course of multiple interactions. Individual user preferences are less likely to align with a general population's preferences than if the model collected dialogues from all users.

The context of the application in physically situated interactions may best determine which model type to use. We recommend user-specific models for longitudinal use cases, such as a long-term interaction that will occur over the course of tens or hundreds of dialogues with a situated agent. We suggest a general model that leverages all users when short-term interactions across a greater user population are expected.

Chapter 9

# Conclusion and Future Work

The work described in this thesis examines miscommunication detection and recovery in physically situated dialogue. This work allows agents with spoken dialogue systems to be more resilient to miscommunication and operate in a wider range of scenarios than in previous work (i.e., this approach is *novel*). We demonstrated that learning in as structurally complex a domain as physically situated dialogue is possible – and with long-standing machine learning techniques (i.e., this approach is *feasible*). The integrated framework we presented enables an agent to engage in physically situated dialogue, detect problems with user instructions, and learn to initiate better recovery strategies from experience (i.e., this approach is *useful*).

This type of dialogue opens up a dynamic and rapidly expanding set of tasks in human-computer interaction. Dialogue systems in physically situated contexts rely on streaming sources of environment information, coupled with language, to determine what (if any) tasks are to be accomplished. At the same time, task success depends on being able to not only interpret the speech and language information being exchanged, but also how that information can be grounded to the current environment context. An agent's immediate surroundings could change at a moment's notice, especially if the agent is mobile. As such, the agent must be able to reason and determine if there are grounding problems with actions under discussion. If there is indeed a problem, an agent can use dialogue recovery strategies to present its current understanding of the situation to a dialogue partner, and offer ways to get the conversation back on track.

## 9.1 Summary of Contributions

In this thesis, we identified two core problems that occur in physically situated dialogue: referential ambiguity and impossible-to-execute instructions (Chapter 1). Chapter 2 presented existing research on miscommunication detection and recovery in dialogue. In Chapter 3, we studied the nature of *route instructions*, a primarily linguistic method for delivering tasks to physically situated agents. Next we presented a representation that connects plans to grounding, called the *plan grounding model* (Chapter 4). We described how people recover from this type of miscommunication, and how people expect agents to recover from similar problems using a crowdsourced data collection approach (Chapter 5). We studied these *situated grounding problems* in the virtual agent navigation domain (Chapter 6). We represented virtual agents as robots that needed to accomplish tasks in a variety of situations. These observations led to the development of a software infrastructure that enables detection of situated grounding problems by combining spoken language input with a representation of the agent's surroundings and plan information. We developed a series of recovery strategies to resolve situated grounding problems, motivated by observations from human-human dialogue corpora and crowdsourced from people taking the perspective of an agent (Chapter 7). Strategies are a combination of conveying the agent's internal understanding of the situation with, if appropriate, a question that could repair the problem. A dialogue agent learns over time which recovery strategies to choose in new situations based on successful resolutions of previous problems (Chapter 8). These techniques can make the agent more effective at completing physically situated tasks with people that are willing to help.

The goal of this thesis was to develop a contextually aware, adaptive approach to recovery from situated grounding problems. This was made possible by developing a model for grounding at the *joint projects* level of human-computer communication. Activities are conducted in a joint fashion by the person formulating a plan and by the agent executing it.

We have shown that a *plan grounding model* allows a dialogue agent to reason about plans in the context of navigation and ask for clarification when appropriate. We developed a method for detecting situated grounding problems that combines multiple sources of information: that of the spoken language input, the environment context, and automated planning components responsible for executing tasks. A software component called the Situated Reasoner encapsulates this representation

and enables a spoken dialogue system for human-robot interaction, TeamTalk, to detect and recover from referential ambiguity and impossible-to-execute tasks. Recovery is made possible by selecting recovery strategies that are a combination of presenting the problem and asking for information that can allow the conversation to proceed. We have shown that recovery strategies such as presenting lists with distinctive object properties and refashioning over-specified instructions enable dialogue agents to complete tasks in physically situated dialogue. Often, the agent has to choose a recovery strategy; we have shown how an instance-based learning method improves dialogue recovery and performance as it collects new observations and checks them against successful past situations.

The plan grounding model merits further investigation for other kinds of grounding beyond physically situated dialogue. The learning approach was effective at resolving problems in the navigation domain. We expect that other kinds of grounding could benefit, especially in dialogue domains where collecting interactions is time-intensive and costly. Language-specific discussions about plans, that don't consider the agent's surroundings, could also use the plan grounding model. Referent resolution could instead refer to concepts previously mentioned in discourse. We consider the general recovery strategies initiated by plan grounding to be task-independent skills for a dialogue agent; for the navigation domain they rely on physically situated information like location, spatial reasoning, and intrinsic object properties. Strategies like disambiguating a list and proposing alternative referents were implemented as a domain-independent `Library Agency`. They do not rely on environment information to operate – any domain that requires reference resolution can use them. The approach we proposed has promise to be effective in other tasks. We expect that the most immediate domain to benefit from this work is object manipulation, which can couple the domain-independent strategies with our implementation of reasoning about spatial information and object properties.

## 9.2 Concluding Remarks and Future Directions

This thesis contributes a framework for handling miscommunication in physically situated dialogue. We presented ways that problems naturally occur in this type of dialogue, and provided strategies agents can use to recover from them. In this section, we provide concluding remarks on this topic and outline opportunities for future work.

### 9.2.1 Recovery Strategies for Physically Situated Dialogue

Recovery strategies in physically situated dialogue boil down to a fairly limited set of strategy types. For referential ambiguity, useful strategies were to confirm one referent, or present a list of choices to the dialogue partner. We believe that listing is most preferable for agents: listing grounds awareness of surroundings, presents a fixed set of options to the user, and constrains the range of linguistic responses. For impossible recovery, the strategies we identified were to simply declare the problem, propose an alternative plan, or by extension, propose a list of alternatives. We made these observations in existing human-human dialogue corpora, and methodically validated them using crowdsourced data collection techniques.

We found that people tend to traverse a *salience hierarchy* [51] when they consider describing the properties of a scene. When teasing apart candidate referents, they usually select the most salient feature. The Situated Reasoner's learning component was one way to replicate this hierarchy for agents. The *Givenness Hierarchy* [44], a framework for determining the linguistic focus of referring expressions in discourse, can also be useful for resolving referents. In the Givenness hierarchy, referents specified as pronouns like *it* indicate higher focus than referents specified with determiners like *the* (e.g., the box). A higher focus in the hierarchy means a greater attention towards – and more immediate memory associated with – a referent. Within an utterance, the Givenness hierarchy offers a computationally feasible way to track referents as they are mentioned. Williams et al. implemented the entirety of the Giveness hierarchy, and demonstrated its usefulness in the reference resolution task [147].

The salience hierarchy makes use of multimodal information, and resolves an ambiguous referent by determining distinctive properties. In contrast, the Givenness hierarchy relies on linguistic distinctions in how referents are specified; this enables handling of long route instructions that mention a referent multiple times. However, both hierarchies have their limitations. The salience hierarchy depends on certainty regarding the properties of objects observed by an agent, and is only functional with nouns specified with the determiner *the*. The Givenness hierarchy relies on pronouns and determiners that can become ambiguous across multiple dialogue partners. A combined approach using both hierarchies may be best, especially in cases where the agent lost track of a referent and must rely on the discourse history to determine the most likely referent the user mentioned

previously.

Physically situated dialogue follows Clark's *least collaborative effort* principle [27] – people tend to use environment properties that require less cognitive effort to describe, like visual information, as opposed to those that must be computed, like spatial processing and history. As discussed in Chapter 5, people were tasked with taking an agent's perspective and selecting environment properties to include in recovery strategies. They selected properties that would require less work on their part, but we also interpret this as collaborative – it is intended to also be less work for the listener to process. We also found that good situation awareness is difficult to acquire in virtual environments. This may impact dialogue interactions with virtual agents differently than those with robots in real world spaces.

Recovery strategies for physically situated dialogue make use of some skills borrowed from traditional dialogue systems. Some task independent dialogue skills are disambiguation, computing distinctiveness relations between choices in a list, and constraint relaxation methods to refashion overspecified instructions. Disambiguation is a necessary technique for referential ambiguity recovery, while constraint relaxation methods are a way for an agent to propose an alternative plan to one detected as impossible. We identified physically situated skills like spatial reasoning, proximity reasoning, and processing intrinsic object properties like color and size. Not surprisingly these same skills were also necessary for detecting problems.

### 9.2.2 Recovery Strategy Learning

Learning to improve recovery strategy decision-making is possible, and can scale in a domain as rich and complex as physically situated interaction. We found that instance-based learning, a well-known machine learning approach, to be effective at improving recovery performance with experience. Moreover, learning was effective despite the limited access to training data germane to physically situated dialogue.

Our results stress the importance of user modeling; the user-specific models performed best in the real-time evaluation found in Chapter 8. We expect that the user's perception of a robot's intelligence and capabilities can strongly impact the success of recovery strategies. In this research, we attempted to clearly indicate the agent's capabilities through its dialogue actions and task instructions. Modeling

the user's belief of the agent's intelligence over course of successful and failed interactions could impact whether the agent takes ambitious strategies (e.g., *Blue box?*) or conservative ones (e.g., *I don't see the blue box. Did you mean the red box?*). Paek and Horvitz [53] investigated a model that would ask fewer clarification questions as the user became more frustrated.

Cognitive modeling provides another method for the agent to monitor the user. A cognitive architecture like ACT-R [3] can permit the agent to behave and react with human-like mental processes. A cognitive model of the agent's perception of the environment could learn relations between environment salience and recovery strategy selection. Simulating confidence about environment observations could reorder which properties to include in a strategy. For instance, a model that has poor color performance could rely on egocentric proximity instead. In addition to spatial reasoning (as in [134]), a cognitive model could simulate mental load. This could impact not only the content of a recovery strategy (e.g., preferring visual information) but also the delivery of the strategy. Listing options may be more work for the agent to do when its cognitive model is incurring a high load, so it could resort to a shorter yes-no confirmation. Taking the user's perspective with a cognitive model would also be useful. This kind of model could monitor the agent's mutual beliefs with the user. For example, when the model determines that the user is no longer visible in the current scene (and therefore no longer co-present), a more verbose explanation of the agent's surroundings could be issued.

Preferences on how to interact with agents in this space can vary from person to person. Exact situations between people occur less frequently in this problem space, so we encourage user-specific modeling when possible. An agent that engages in longitudinal interactions like a domestic robotic assistant would greatly benefit from such a learning technique. Finally, we believe there to be a tradeoff between more data and the preferences obtained from a user population. Better understanding of this could help us determine the true ceiling of recovery strategy selection performance.

### 9.2.3 Future Work

In this thesis, we only looked at exchanges of dialogue where plans were conveyed in at most one action at a time. In future work, we hope to extend the current understanding capabilities to process sequences of actions. We anticipate that this will require a thorough understanding of human recovery, coupled with real-

time evaluations with agents in several tasks relevant from the *learning from demonstration* community, such as robot navigation and manipulation. We could also develop recovery strategies that pinpoint specific parts of a multi-step dialogue that were problematic, and attempt to repair the inconsistencies in them. The strategies that we developed for this thesis serve as a starting point; conveying a dialogue agent's understanding of a sequence of context-dependent actions may require careful consideration of the partner's understanding of the situation. At least initially, being able to craft elements of a plan with codewords (e.g., step ALPHA) would permit users and agents to discuss and amend steps to a plan.

The recovery strategies in this thesis used templates to generate natural language. When presented to the user, the language had consistent form. Over time, the language form could become repetitive and impact how users perceive the intelligence of the agent. Improving the naturalness of the agent's language generation is an important topic for future work. Approaches from the growing body of work on referring expression generation (e.g., [40, 137]) could be combined with the recovery strategy selection algorithms we developed for this thesis. We expect that a combined approach would improve the agent's delivery of recovery strategies.

Borrowing the brevity to strategies obtained from human-human dialogue (see Section 7.1) could also improve generation, especially for agents that produce spoken language. However, reducing the length of strategies runs the risk that the agent fails to provide sufficient grounding or understanding of a situation. This could lead to miscommunication later in the dialogue. Moratz et al. found that people quickly change how they issue commands to a robot from high-level goals to turn-by-turn instructions if an instruction is unsuccessful [93]. Revealing the agent's internal state, and monitoring *when* the agent does this, will remain crucial to effective dialogue. The agent could also model the user's preferences for generation. This model could determine whether leveraging the user's memory recognition (e.g., *Did you mean the blue or red box?*) or memory recall (e.g., *Which color box?*) is best for initiating a strategy. Improving the flexibility of generation with these methods should permit more natural grounding.

While the contributions in this thesis enable recovery from situated grounding problems, the agent does not acquire new knowledge about the situation beyond whether or not the selected strategy was successful. A desirable extension would be to amend the agent's situation awareness using automated techniques to update its representation of the environment. These could be as simple as learning that an object being discussed has changed state; such as a door opening or closing.

We make the simplifying assumption in this thesis that the agent has *a priori* knowledge about its environment surroundings. There is no notion of uncertainty about the properties and locations of objects that it has seen. The system does have a notion of visibility, or line of sight, which is absolute in nature. When joining the environment, the virtual agent can only see the objects in front of it, and as it moves, others become visible. The color, shape, and other properties of these objects become known without a measure of uncertainty. In real-world systems, a physically situated agent will often need to operate with uncertain sensory perception. As a result, miscommunication detection and recovery, along with the dialogue overall, can be affected. We encourage others to continue to push forward in the growing body of work on decision making under uncertainty.

If the rate of data collection in this space were to become inexpensive, we still expect user-specific models to outperform a general one. However, there may be a way to determine a limited number of user profiles that classify general preferred ways to interact with an agent (especially how to refer to the agent's surroundings). We expect this to vary based on whether the user is co-present or not.

The learning approach explored in this work, instance-based learning, can be effective in other domains where tasks can be long and costly to collect. While we expect such a learning approach to be effective in other domains of human-computer dialogue, they still remain to be explored. In addition, parallel collections of data in the same domain, which may in serial processing be costly, could also prove effective as a way to improve recovery strategy selection over time. We believe that instance-based learning holds promise in dialogue system research as a way to obtain value from limited amounts of dialogue data, collect without simulation.

The agent's vocabulary restricts the range of words and phrases users can say to refer to the surroundings. We expect out-of-vocabulary and common-sense reasoning techniques to supplement the agent's knowledge in these situations in future research. OOV techniques can allow an agent to learn new words, or at the very least supplement its own vocabulary of relevance with existing words, using building blocks like phonemes and subword units. Agents must learn not only the words that refer to objects, but relationships between objects (e.g., the presence of a desk and a computer monitor may imply that the space is an office). These higher-order relations are also referable in physically situated dialogue and will require greater reasoning capabilities.

Lastly, there are several opportunities for future research on physically situated

dialogue. Multimodal grounding requires a much greater understanding of human-human communication than what is available in today's corpora. We must explore what modes of communication (e.g., gaze, gesture, speech) people use in a variety of physically situated interactions, both with other people and with agents. Within multimodal interactions, an agent must reason about the uncertainties associated with these sensory perceptions. We must explore ways for an agent to assess confidence with various sources of information as it talks about tasks relevant to its environment with users. Miscommunication recovery with multimodal input opens up a greater number of strategies. For example, being able to disambiguate a referent using gaze and speech simultaneously could be more effective than a verbal description alone; we observed this strategy to be common in the SCARE corpus. Head pose, arm, and body gestures offer additional ways to express information about the agent's surroundings. An integrative approach to building intelligent agents remains a promising area for future research.

# Bibliography


[1] Patricia L. Alfano and George F. Michel. Restricting the field of view: Perceptual and performance effects. *Perceptual and motor skills*, 70(1): 35–45, 1990. [5.3.2]

[2] Anne H. Anderson, Miles Bader, Ellen Gurman Bard, Elizabeth Boyle, Gwyneth Doherty, Simon Garrod, Stephen Isard, Jacqueline Kowtko, Jan McAllister, Jim Miller, Catherine Sotillo, Henry S. Thompson, and Regina Weinert. The HCRC Map Task Corpus. *Language and Speech*, 34(4): 351–366, 1991. [1.1, 3.2.1, 5.2, 6.1, 6.4.2, 7.1.1, 8.1.1]

[3] John R. Anderson. ACT: A simple theory of complex cognition. *American Psychologist*, 51(4):355, 1996. [9.2.2]

[4] Kevin Wayne Arthur. *Effects of field of view on performance with head-mounted displays*. PhD thesis, University of North Carolina at Chapel Hill, 2000. [5.3.2]

[5] S. Balakirsky, C. Scrapper, and E. Messina. Mobility open architecture simulation and tools environment. In *Proc. of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 2005. [6.2.2]

[6] Adrian Bangerter and Herbert H. Clark. Navigating joint projects with dialogue. *Cognitive Science*, 27(2):195–225, 2003. [5.1, 5.5]

[7] Dhananjay Bansal and Mosur K. Ravishankar. New features for confidence annotation. In *Proc. of ICSLP*, 1998. [2.2]

[8] Samuel Bayer, Christine Doran, and Bryan George. Exploring speech-enabled dialogue with the Galaxy Communicator infrastructure. In *Proc. of HLT*, 2001. [6.2.3]

[9] Christina Bennett and Alexander I. Rudnicky. The Carnegie Mellon Communicator Corpus. In *Proc. of ICSLP*, pages 341–344, 2002. [3.1.1, 3.2.3]

[10] Dan Bohus. *Error Awareness and Recovery in Conversational Spoken Language Interfaces*. PhD thesis, Carnegie Mellon University, 2007. [1, 2.3, 4.1.5, 5.1, 6.3.1, 6.6]

[11] Dan Bohus and Eric Horvitz. On the Challenges and Opportunities of

Physically Situated Dialog. In *Proc. of the AAAI Fall Symposium on Dialog with Robots*, 2010. [1, 1.1, 6.1]

[12] Dan Bohus and Alexander I. Rudnicky. Modeling the Cost of Misunderstanding Errors in the CMU Communicator Dialog System. In *Proc. of ASRU*, 2001. [4.1.5]

[13] Dan Bohus and Alexander I. Rudnicky. Integrating Multiple Knowledge Sources for Utterance-Level Confidence Annotation in the CMU Communicator Spoken Dialog System. *Technical Report CS-190*, 2002. [1, 2.2]

[14] Dan Bohus and Alexander I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23 (3):332–361, 2009. [6.2.3]

[15] Dan Bohus, Antoine Raux, Thomas K. Harris, Maxine Eskenazi, and Alexander I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *Proc. of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, 2007. [6.2.3]

[16] Gies Bouwman, Janienke Sturm, and Louis Boves. Incorporating Confidence Measures in the Dutch Train Timetable Information System Developed In the ARISE Project. In *Proc. of ICASSP*, pages 493–496, 1999. [1, 2.3]

[17] Jack E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems journal*, 4(1):25–30, 1965. [6.10, 6.4.2]

[18] Guido Bugmann, Ewan Klein, Stanislao Lauria, and Theocharis Kyriacou. Corpus-Based Robotics: A Route Instruction Example. In *Proc. of IAS-8*, pages 96–103, 2004. [1.1]

[19] Jennifer L. Burke, Robin R. Murphy, Michael D. Coovert, and Dawn L. Riddle. Moonlight in Miami: Field study of human-robot interaction in the context of an urban search and rescue disaster response training exercise. *Human–Computer Interaction*, 19(1-2):85–116, 2004. [5.3.2, 5.5]

[20] Janet E. Cahn and Susan E. Brennan. A Psychological Model of Grounding and Repair in Dialog. In *Proc. of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999. [4.1.1, 4.1.1]

[21] Jean Carletta. Planning to fail, not failing to plan: Risk-taking and recovery in task-oriented dialogue. In *Proc. of the 14th Conference on Computational Linguistics, Volume 3*, pages 896–900, 1992. [5.2, 5.3.2]

[22] Jean Carletta, Stephen Isard, Gwyneth Doherty-Sneddon, Amy Isard, Jacqueline C. Kowtko, and Anne H. Anderson. The reliability of a dialogue structure coding scheme. *Computational Linguistics*, 23, 1997. [7.1.1]

[23] Laura A. Carlson and Patrick L. Hill. Formulating Spatial Descriptions

across Various Dialogue Contexts. In Kenny R. Coventry, Thora Tenbrink, and John Bateman, editors, *Spatial Language and Dialogue*. Oxford University Press, 2009. [5.2, 5.5]

[24] Stefano Carpin, Mike Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. USARSim: a robot simulator for research and education. In *Proc. of ICRA '07*, pages 1400 –1405, 2007. [3.2.2, 6.2.2]

[25] Jennifer Casper and Robin R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(3):367–385, 2003. [5.3.2, 5.5]

[26] Lin Chase. *Error-Responsive Feedback Mechanisms for Speech Recognizers*. PhD thesis, Carnegie Mellon University, 1997. [2.2]

[27] Herbert H. Clark. *Using Language*. Cambridge University Press, May 1996. [1, 2.1.1, 4.1.1, 4.1.1, 4.3, 5.3.2, 6.3.1, 6.6, 6.3.2, 9.2.1]

[28] Herbert H. Clark and Susan E. Brennan. Grounding in communication. *Perspectives on socially shared cognition*, 1991. [1, 4.1.1]

[29] Herbert H. Clark and Edward F. Schaefer. Contributing to Discourse. *Cognitive Science*, 13, 1989. [4.1.1]

[30] Ron Cowan. *The Teacher's Grammar of English with Answers: A Course Book and Reference Guide*. Cambridge University Press, 2008. [5.3.3]

[31] Stephen Cox and Richard Rose. Confidence measures for the SWITCHBOARD database. In *Proc. of ICASSP*, 1996. [2.2]

[32] Marie-Paule Daniel and Michel Denis. The production of route directions: Investigating conditions that favour conciseness in spatial discourse. *Applied cognitive psychology*, 18(1):57–75, 2003. [3.1]

[33] Rodrigo de Oliveira, Alexandros Karatzoglou, Pedro Concejero Cerezo, Ana Armenta Lopez de Vicuña, and Nuria Oliver. Towards a psychographic user model from mobile phone usage. In *CHI Extended Abstracts on Human Factors in Computing Systems*, 2011. [8.4]

[34] Robin Deits, Stefanie Tellex, Pratiksha Thaker, Dimitar Simeonov, Thomas Kollar, and Nicholas Roy. Clarifying commands with information-theoretic human-robot dialog. *Journal of Human-Robot Interaction*, 2(2):58–79, 2013. [6.3.1]

[35] Robert Desimone and John Duncan. Neural mechanisms of selective visual attention. *Annual review of neuroscience*, 18(1):193–222, 1995. [5.3.2, 5.5]

[36] Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to do and how to do it: Translating natural language directives into temporal

and dynamic logic representation for goal management and action execution. In *Proc. of ICRA*, 2009. [1.1]

[37] Kathleen M. Eberhard, Hannele Nicholson, Sandra Kübler, Susan Gundersen, and Matthias Scheutz. The Indiana "Cooperative Remote Search Task" (CReST) Corpus. In *Proc. of LREC*, 2010. [3.2.1]

[38] Floriana Esposito, Oriana Licchelli, and Giovanni Semeraro. Discovering student models in e-learning systems. *Journal of Universal Computer Science*, 10(1):47–57, 2004. [8.4]

[39] Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Kristina Striegnitz. The GIVE-2 Corpus of Giving Instructions in Virtual Environments. In *Proc. of LREC*, 2010. [3.2.1]

[40] Konstantina Garoufi and Alexander Koller. Generation of effective referring expressions in situated context. *Language, Cognition and Neuroscience*, 29 (8):986–1001, 2014. [5.2, 9.2.3]

[41] Genevieve Gorrell, Ian Lewin, and Manny Rayner. Adding intelligent help to mixed-initiative spoken dialogue systems. In *Proc. of Interspeech*, 2002. [2.3]

[42] Arthur C. Graesser, Kurt VanLehn, Carolyn P. Rosé, Pamela W. Jordan, and Derek Harter. Intelligent tutoring systems with conversational dialogue. *AI Magazine*, 22(4):39, 2001. [1]

[43] Markus Guhe and Ellen Gurman Bard. Adapting referring expressions to the task environment. In *Proc. of the 30th Annual Conference of the Cognitive Science Society*, pages 2404–2409, 2008. [5.2, 5.5]

[44] Jeanette K. Gundel, Nancy Hedberg, and Ron Zacharski. Cognitive status and the form of referring expressions in discourse. *Language*, pages 274–307, 1993. [9.2.1]

[45] Edward T. Hall. *The Hidden Dimension*. Doubleday & Co., 1966. [5]

[46] Eui-Hong Han, George Karypis, and Vipin Kumar. Text Categorization Using Weight Adjusted-Nearest Neighbor Classification. In David Cheung, Graham Williams, and Qing Li, editors, *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 53–65. Springer Berlin - Heidelberg, 2001. [8.1.2]

[47] Stevan Harnad. The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1):335–346, 1990. [1, 4.1.3]

[48] T.K. Harris and A.I. Rudnicky. TeamTalk: A platform for multi-human-robot dialog research in coherent real and virtual spaces. In *Proc. of AAAI*, 2007. [6.2.1]

[49] David A. Harville. Maximum likelihood approaches to variance component estimation and to related problems. *Journal of the American Statistical Association*, 72(358):320–338, 1977. [3, 3]

[50] Sachithra Hemachandra, Felix Duvallet, Thomas M Howard, Nicholas Roy, Anthony Stentz, and Matthew R. Walter. Learning Models for Following Natural Language Directions in Unknown Environments. *arXiv preprint arXiv:1503.05079*, 2015. [1.1]

[51] Graeme Hirst, Susan McRoy, Peter Heeman, Philip Edmonds, and Diane Horton. Repairing conversational misunderstandings and non-understandings. *Speech Communication*, 15(3-4):213 – 229, 1994. [2.1, 2.1.1, 5.5, 9.2.1]

[52] Eric Horvitz and Tim Paek. A computational architecture for conversation. In *Proc. of International Conference on User Modeling*, pages 201–210, 1999. [4.1.2]

[53] Eric Horvitz and Tim Paek. Harnessing models of users' goals to mediate clarification dialog in spoken language systems. In *Proc. of International Conference on User Modeling*, pages 3–13. Springer, 2001. [9.2.2]

[54] Eric Horvitz and Tim Paek. Computer-based representations and reasoning methods for engaging users in goal-oriented conversations, July 16 2002. US Patent 6,421,655. [4.1.2]

[55] Eric Horvitz and Tim Paek. Multi-level decision-analytic approach to failure and repair in human-computer interactions, December 3 2002. US Patent 6,490,698. [4.1, 4.2]

[56] Ting-Hao K. Huang, Walter S. Lasecki, Alan L. Ritter, and Jeffrey P. Bigham. Combining Non-Expert and Expert Crowd Work to Convert Web APIs to Dialog Systems. In *Proc. of AAAI Conference on Human Computation and Crowdsourcing*, 2014. [5.2]

[57] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W Black, Mosur Ravishankar, and Alexander I. Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Proc. of ICASSP*, 2006. [6.2.3]

[58] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, 4: 237–285, 1996. [4.1.5]

[59] Alexander Klippel and Stephan Winter. Structural salience of landmarks for route directions. *Spatial information theory*, pages 347–362, 2005. [3.1]

[60] Holger Knublauch, Ray W. Fergerson, Natalya F. Noy, and Mark A. Musen. The Protege OWL plugin: An open development environment for semantic

web applications. In *Proc. of IWSC*, 2004. [6.1, 6.2.1]

[61] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In *Proc. of HRI*, 2010. [1.1]

[62] Kazunori Komatani and Tatsuya Kawahara. Generating effective confirmation and guidance using two-level confidence measures for dialogue systems. In *Proc. of Interspeech*, pages 648–648, 2000. [4.1.5]

[63] Theodora Koulouri and Stanislao Lauria. Exploring Miscommunication and Collaborative Behaviour in Human-Robot Interaction. In *Proc. of SIGdial*, pages 111–119, 2009. [1.1, 5.2]

[64] Theodora Koulouri and Stanislao Lauria. A corpus-based analysis of route instructions in human-robot interaction. Technical report, University of Ulster, 2009. [1.1]

[65] Emiel Krahmer, Marc Swerts, Mariet Theune, and Mieke Weegels. Error detection in spoken human-machine interaction. *International Journal of Speech Technology*, 4(1):19–29, 2001. [2.3, 4.1.5]

[66] Geert-Jan Kruijff, Hendrik Zender, Patric Jensfelt, and Henrik I. Christensen. Situated dialogue and understanding spatial organization: Knowing what is where and what you can do there. In *Proc. of RO-MAN*, 2006. [5.2]

[67] Geert-Jan Kruijff, Miroslav Janicek, and Hendrik Zender. Situated Communication for Joint Activity in Human-Robot Teams. *IEEE Intelligent Systems*, 27(2):27–35, 2012. [1.1]

[68] Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. Chorus: a crowd-powered conversational assistant. In *Proc. of UIST*, 2013. [5.2]

[69] Yezdi Lashkari, Max Metral, and Pattie Maes. Collaborative interface agents. *Readings in agents*, page 111, 1998. [8.4]

[70] Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein. Training Personal Robots Using Natural Language Instruction. *IEEE Intelligent Systems*, 16:38–45, 2001. [3.2.1]

[71] Stanislao Lauria, Guido Bugmann, Theocharis Kyriacou, Johan Bos, and Ewan Klein. Converting natural language route instructions into robot executable procedures. In *Proc. of RO-MAN*, pages 223–228. IEEE, 2002. [1.1]

[72] Patrick Lester. A* pathfinding for beginners. 2005. URL `http://www.policyalmanac.org`. [6.4.2]

[73] Esther Levin and Roberto Pieraccini. A Stochastic Model of Human-Machine Interaction for Learning Dialog Strategies. In *IEEE Transactions*

*on Speech and Audio Processing*, pages 11–23, 2000. [4.1.5]

[74] Gina-Anne Levow. Characterizing and recognizing spoken corrections in human-computer dialogue. In *Proc. of ACL*, 1998. [1, 2.2]

[75] Diane J. Litman, Julia B. Hirschberg, and Marc Swerts. Predicting Automatic Speech Recognition Performance Using Prosodic Cues. In *Proc. of NAACL*, 2000. [1, 2.2]

[76] Changsong Liu, Jacob Walker, and Joyce Chai. Ambiguities in Spatial Language Understanding in Situated Human Robot Dialogue. In *Proc. of the AAAI Fall Symposium on Dialog with Robots*, 2010. [2.1.2]

[77] Kristin L. Lovelace, Mary Hegarty, and Daniel R. Montello. Elements of good route directions in familiar and unfamiliar environments. *Spatial information theory*, 1999. [3.1]

[78] Matt MacMahon. Walk the Talk: Connecting language, knowledge, and action in route instructions. In *Proc. of AAAI*, 2006. [1.1]

[79] Matt MacMahon. *Following Natural Language Route Instructions*. PhD thesis, University of Texas at Austin, 2007. [3.1, 3.2.1]

[80] Ramesh Manuvinakurike and David DeVault. Pair Me Up: A Web Framework for Crowd-Sourced Spoken Dialogue Collection. In *Proc. of IWSDS*, 2015. [5.2]

[81] Matthew Marge and Alexander I. Rudnicky. Comparing Spoken Language Route Instructions for Robots across Environment Representations. In *Proc. of SIGdial*, 2010. [1.1, 3, 3.1, 3.2.2, 3.2.3, 3.2.4, 3.2.4]

[82] Matthew Marge and Alexander I. Rudnicky. Towards Overcoming Miscommunication in Situated Dialogue by Asking Questions. In *Proc. of the AAAI Fall Symposium on Building Representations of Common Ground with Intelligent Agents*, 2011. [5.2, 7.1, 7.1.1, 7.2, 8.1.1]

[83] Matthew Marge and Alexander I. Rudnicky. The TeamTalk Corpus: Route Instructions in Open Spaces. In *Proc. of RSS Workshop on Grounding Human-Robot Dialog for Spatial Tasks*, 2011. [3, 6.1, 6.4.2]

[84] Matthew Marge and Alexander I. Rudnicky. Towards evaluating recovery strategies for situated grounding problems in human-robot dialogue. In *Proc. of RO-MAN*, 2013. [5.1]

[85] Matthew Marge and Alexander I. Rudnicky. Miscommunication Recovery in Physically Situated Dialogue. In *Proc. of SIGdial*, 2015. [3, 5]

[86] Matthew Marge, Aasish Pappu, Benjamin Frisch, Thomas K. Harris, and Alexander I. Rudnicky. Exploring Spoken Dialog Interaction in Human-Robot Teams. In *Proc. of Robots, Games, and Research: Success stories in*

*USARSim IROS Workshop*, 2009. [6.2.1, 6.2.3, 6.3, 6.5]

[87] Matthew Marge, Satanjeev Banerjee, and Alexander I. Rudnicky. Using the Amazon Mechanical Turk for transcription of spoken language. In *Proc. of ICASSP*, 2010. [3.1.1, 3.2.3]

[88] Susan W. McRoy and Graeme Hirst. The repair of speech act misunderstandings by abductive inference. *Computational Linguistics*, 21(4):435–478, 1995. [4.1.1]

[89] Raveesh Meena, Gabriel Skantze, and Joakim Gustafson. Data-driven models for timing feedback responses in a Map Task dialogue system. *Computer Speech & Language*, 28(4):903–922, 2014. [6.1]

[90] Pierre-Emmanuel Michon and Michel Denis. When and Why Are Visual Landmarks Used in Giving Directions? In Daniel Montello, editor, *Spatial Information Theory*, Lecture Notes in Computer Science, pages 292–305. Springer Berlin / Heidelberg, 2001. [3.1]

[91] Margaret Mitchell, Dan Bohus, and Ece Kamar. Crowdsourcing language generation templates for dialogue systems. *Proc. of INLG and SIGdial*, 2014. [5.2]

[92] Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., 1997. [8, 8.1, 8.1, 8.1.2]

[93] Reinhard Moratz, Thora Tenbrink, John Bateman, and Kerstin Fischer. Spatial knowledge representation for human-robot interaction. *Spatial Cognition III*, 2003. [5.5, 9.2.3]

[94] Marius Muja and David G. Lowe. Scalable Nearest Neighbor Algorithms for High Dimensional Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014. [8.1, 8.1.2]

[95] Yukiko I. Nakano, Gabe Reinstein, Tom Stocky, and Justine Cassell. Towards a model of face-to-face grounding. In *Proc. of ACL*, 2003. [4.1.1]

[96] Curtis W. Nielsen, Michael A. Goodrich, and Robert W. Ricks. Ecological interfaces for improving mobile robot teleoperation. *IEEE Transactions on Robotics*, 23(5):927–941, 2007. [5.3.2]

[97] Yasuhisa Niimi and Yutaka Kobayashi. A dialogue control strategy based on the reliability of speech recognition. In *Proc. of ICSLP*, 1996. [4.1.5]

[98] Tim Paek. Toward a taxonomy of communication errors. In *Proc. of ISCA Workshop on Error Handling in Spoken Dialogue Systems*, 2003. [6.3.1, 6.6]

[99] Tim Paek and Eric Horvitz. Uncertainty, utility, and misunderstanding: A decision-theoretic perspective on grounding in conversational systems. In

*Proc. of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999. [4.1.2]

[100] Tim Paek and Eric Horvitz. Conversation as action under uncertainty. In *Proc. of UAI*, 2000. [4.1.2]

[101] Gabriele Paolacci, Jesse Chandler, and Panagiotis G. Ipeirotis. Running experiments on Amazon Mechanical Turk. *Judgment and Decision Making*, 5(5):411–419, 2010. [5.3.1]

[102] Aasish Pappu and Alexander I. Rudnicky. The Structure and Generality of Spoken Route Instructions. In *Proc. of SIGdial*, 2012. [1.1]

[103] Aasish Pappu and Alexander I. Rudnicky. Predicting tasks in goal-oriented spoken dialog systems using semantic knowledge bases. In *Proc. of SIGdial*, 2013. [6.2.1]

[104] Aasish Pappu and Alexander I. Rudnicky. Knowledge acquisition strategies for goal-oriented dialog systems. In *Proc. of SIGdial*, 2014. [6.2.1]

[105] Matthew Purver. CLARIE: Handling Clarification Requests in a Dialogue System. *Research on Language and Computation*, 4(2-3):259–288, 2006. [4.1.4]

[106] Matthew Purver, Jonathan Ginzburg, and Patrick Healey. On the means for clarification in dialogue. In *Proc. of SIGdial*, 2001. [1.1]

[107] Antoine Raux and Maxine Eskenazi. Non-Native Users in the Let's Go!! Spoken Dialogue System: Dealing with Linguistic Mismatch. In *Proc. of HLT-NAACL*, 2004. [2.3]

[108] Antoine Raux and Maxine Eskenazi. Optimizing endpointing thresholds using dialogue features in a spoken dialogue system. In *Proc. of SIGdial*, 2008. [6.3.1]

[109] Antoine Raux, Brian Langner, Dan Bohus, Alan W. Black, and Maxine Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Proc. of Interspeech*, 2005. [1]

[110] Verena Rieser and Johanna D. Moore. Implications for generating clarification requests in task-oriented dialogues. In *Proc. of ACL*, 2005. [5.3.2, 5.5]

[111] Kepa Joseba Rodríguez and David Schlangen. Form, Intonation and Function of Clarification Requests in German Task-Oriented Spoken Dialogues. In *Proc. of SemDial*, 2004. [1.1]

[112] Alexander I. Rudnicky, Aasish Pappu, Peng Li, Matthew Marge, and Benjamin Frisch. Instruction Taking in the TeamTalk System. In *Proc. of the AAAI Fall Symposium on Dialog with Robots*, 2010. [1.1, 6.2.1]

[113] Rubén San-Segundo, Bryan Pellom, Wayne Ward, and José M Pardo. Confidence measures for dialogue management in the CU Communicator system. *Proc. of ICASSP*, 2000. [1, 2.2]

[114] Gregory A. Sanders, Audrey N. Le, and John S. Garofolo. Effects of word error rate in the DARPA communicator data during 2000 and 2001. In *Proc. of Interspeech*, 2002. [2.1.2]

[115] Konrad Scheffler and Steve Young. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of HLT*, 2002. [4.1.5]

[116] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *Proc. of SIGIR*, 2002. [8.4]

[117] Stephanie Schuldes, Michael Roth, Anette Frank, and Michael Strube. Creating an Annotated Corpus for Generating Walking Directions. In *Proc. of the Workshop on Language Generation and Summarisation*, 2009. [1.1]

[118] John R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1969. [4.1.1]

[119] Hui Shi and Bernd Krieg-Brückner. Modelling Route Instructions for Robust Human-Robot Interaction on Navigation Tasks. *International Journal of Software and Informatics*, 2(1):33–60, 2008. [1.1]

[120] Hui Shi and Thora Tenbrink. Telling Rolland where to go: HRI dialogues on route navigation. In Kenny R. Coventry, Thora Tenbrink, and John Bateman, editors, *Spatial Language and Dialogue*, pages 177–190. Oxford University Press, 2009. [1.1]

[121] Nobuyuki Shimizu and Andrew Haas. Learning to follow navigational route instructions. In *Proc. of IJCAI*, 2009. [1.1]

[122] Chanop Silpa-Anan and Richard Hartley. Optimised KD-trees for fast image descriptor matching. In *Proc. of CVPR*, 2008. [8.1.2]

[123] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. *Journal of Artificial Intelligence Research*, 16: 105–133, 2002. [4.1.5]

[124] Gabriel Skantze. Exploring human error recovery strategies: Implications for spoken dialogue systems. *Speech Communication*, 45(3):325–341, 2005. [1, 2.3, 5.1, 5.3.2, 5.5]

[125] Gabriel Skantze. Making grounding decisions: Data-driven estimation of dialogue costs and confidence thresholds. In *Proc. of SIGdial*, 2007. [4.2.2]

[126] Gabriel Skantze. *Error Handling in Spoken Dialogue Systems: Managing Uncertainty, Grounding and Miscommunication*. PhD thesis, KTH Royal Institute of Technology, 2007. [2.1, 2.3, 5.1, 6.3.1]

[127] Michael J. Spivey, Michael K. Tanenhaus, Kathleen M. Eberhard, and Julie C. Sedivy. Eye movements and spoken language comprehension: Effects of visual context on syntactic ambiguity resolution. *Cognitive Psychology*, 45(4):447–481, 2002. [5.3.2]

[128] Laura Stoia, Darla Magdalena Shockley, Donna K. Byron, and Eric Fosler-Lussier. SCARE: A Situated Corpus with Annotated Referring Expressions. In *Proc. of LREC*, 2008. [1.1, 3.2.1, 5.2, 6.1, 6.4.2, 7.1.1, 8.1.1]

[129] William Swartout, Jonathan Gratch, Randall W. Hill, Eduard Hovy, Stacy Marsella, Jeff Rickel, and David Traum. Toward virtual humans. *AI Magazine*, 27(2):96–108, 2006. [1]

[130] Marc Swerts, Diane Litman, and Julia Hirschberg. Corrections in Spoken Dialogue Systems. In *Proc. of Interspeech*, 2000. [1, 2.2]

[131] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *Proc. of AAAI*, 2011. [1.1]

[132] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI Magazine*, 32(4):64–76, 2011. [4.1.3]

[133] Thora Tenbrink, Robert J. Ross, Kavita E. Thomas, Nina Dethlefs, and Elena Andonova. Route instructions in map-based human-human and human-computer dialogue: A comparative analysis. *Journal of Visual Languages & Computing*, 21(5):292–309, 2010. [1.1, 4.1.4, 5.1]

[134] J. Gregory Trafton, Nicholas L. Cassimatis, Magdalena D. Bugajska, Derek P. Brock, Farilee E. Mintz, and Alan C. Schultz. Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 35 (4):460–470, 2005. [9.2.2]

[135] David R. Traum. On Clark and Schaefer's Contribution Model and its Applicability to Human-Computer Collaboration. In *Proc. of the Workshop on Use of Clark's Models of Language for the design of Cooperative Systems*, 1998. [4.1.1]

[136] Ielka Francisca Van Der Sluis. *Multimodal Reference, Studies in Automatic Generation of Multimodal Referring Expressions*. PhD thesis, University of

Tilburg, 2005. [5.5]

[137] Jette Viethen and Robert Dale. Algorithms for generating referring expressions: Do they do what people do? In *Proc. of INLG*, 2006. [5.1, 5.2, 5.3.2, 5.5, 9.2.3]

[138] Adam Vogel and Dan Jurafsky. Learning to follow navigational directions. In *Proc. of ACL*, 2010. [1.1]

[139] Marilyn Walker, Donald Hindle, Jeanne Fromer, and Craig Mestel. Evaluating Competing Agent Strategies for a Voice Email Agent. In *Proc. of EUROSPEECH*, 1997. [2.1.2]

[140] Marilyn Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. Evaluating spoken dialogue agents with PARADISE: Two case studies. *Computer Speech & Language*, 12(4):317–347, 1998. [2.1.2]

[141] Marilyn Walker, Jerry Wright, and Irene Langkilde. Using Natural Language Processing and Discourse Features to Identify Understanding Errors in a Spoken Dialogue System. In *Proc. of ICML*, 2000. [1, 2.2]

[142] Marilyn Walker, Rececca Passonneau, and Julie E. Boland. Quantitative and Qualitative Evaluation of DARPA Communicator Spoken Dialogue Systems. In *Proc. of ACL*, 2001. [2.1.2]

[143] Michael L. Walters, Kerstin Dautenhahn, René Te Boekhorst, Kheng Lee Koay, Christina Kaouri, Sarah Woods, Chrystopher Nehaniv, David Lee, and Iain Werry. The influence of subjects' personality traits on personal spatial zones in a human-robot interaction experiment. In *Proc. of RO-MAN*, 2005. [8.4]

[144] William Yang Wang, Dan Bohus, Ece Kamar, and Eric Horvitz. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *Proc. of SLT*, pages 73–78, 2012. [5.2]

[145] Yu-Fang H. Wang, Stefan W. Hamerich, and Volker Schless. Multi-Modal and Modality Specific Error Handling in the GEMINI Project. In *Proc. of ISCA Workshop on Error Handling in Spoken Dialogue Systems*, 2003. [4.1.5]

[146] Wayne Ward and Sunil Issar. Recent improvements in the CMU spoken language understanding system. In *Proc. of HLT*, 1994. [6.2.3]

[147] Tom Williams, Stephanie Schreitter, Saurav Acharya, and Matthias Scheutz. Towards Situated Open World Reference Resolution. In *Proc. of the AAAI Fall Symposium on Artificial Intelligence and Human-Robot Interaction*, 2015. [9.2.1]

[148] Hendrik Zender, Patric Jensfelt, Óscar Martíinez Mozos, Geert-Jan M. Kruijff, and Wolfram Burgard. An integrated robotic system for spatial under-

standing and situated interaction in indoor environments. In *Proc. of AAAI*, 2007. [1.1]

[149] Teresa Zollo. A Study of Human Dialogue Strategies in the Presence of Speech Recognition Errors. In *Proc. of the AAAI Fall Symposium on Psychological Models of Communication in Collaborative Systems*, 1999. [2.3]

# Appendix A: Thesis Experiment Scenarios

Beginning on the following page, we present all fifty scenarios that were part of the final evaluation of the learning component that was described in Chapter 8.

**Scene labels**[1]: Each scene has a label that indicates one or more of the following properties:

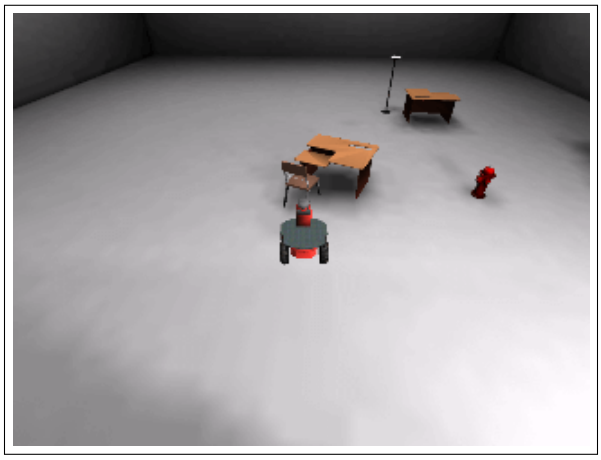
- (I $\vee$ H $\vee$ P $\vee$ S) – ($\emptyset$ $\vee$ C $\vee$ G) - a referential ambiguity scene containing one or more of the following situated features: **i**ntrinsic property, recency in **h**istory, **p**roximity to an object, **s**patial region. For intrinsic properties, they break down into **c**olor and **g**eometric property (size and shape).
- IMP - an impossible-to-execute scene.
- CTRL - an experiment control scene.

[1]For the evaluation described in Chapter 8, we removed some landmarks to enable direction-giving participants to provide problematic instructions to the agent. The agent's actual representation of the environments were those in the following scenes.

### Building 1: 5 Scenes

- "Go to the chair"
  **Scene 1:** Referential ambiguity that can be resolved by intrinsic property (the chairs are different colors), proximity (one of the chairs is near a desk), or spatial region (one chair is in front of the agent, and one to its right). **(IPS-C)**

- "Move to the hydrant"
  **Scene 2:** No situated grounding problem (control for participant). Only one hydrant is in the building. **(CTRL)**



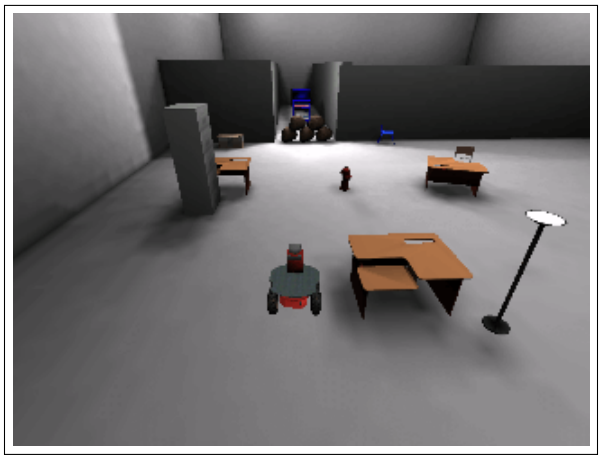
- "Move to the vendor"
  **Scene 3:** Impossible-to-execute instruction: No object of that type exists (no vendor in the building). Vendor is the code word for soda machine in this work. **(IMP)**

- "Move to the desk"

  **Scene 4:** Referential ambiguity that can be resolved by referring to an object in its recent history (the agent encountered a desk in the instruction, "Go to the chair"), proximity to a nearby object (one desk is by a lamp, another by a cabinet, and another by a chair), or spatial region (one desk is in front of the agent, one to its left, and one behind it). **(HPS)**

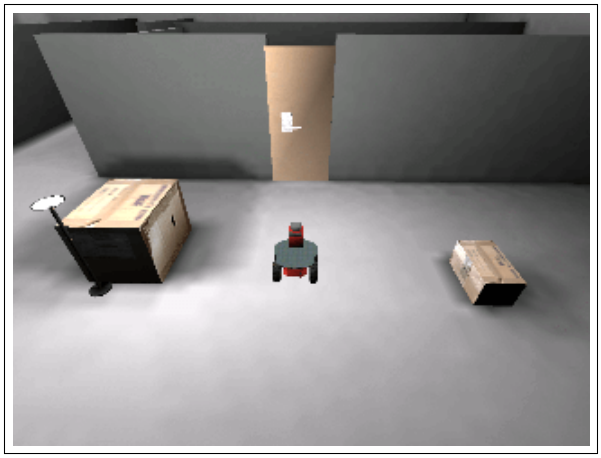
- "Go to the blue mailbox"

  **Scene 5:** Impossible-to-execute instruction: path not possible by obstruction (the path to the mailbox is blocked by a rock). **(IMP)**

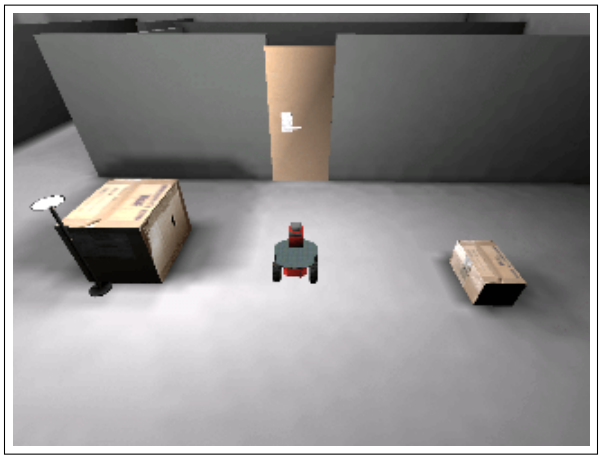


### Building 2: 5 Scenes

- "Go through the door"
  **Scene 6:** Impossible-to-execute instruction: path not possible by object state (the only doorway that could match is closed). **(IMP)**

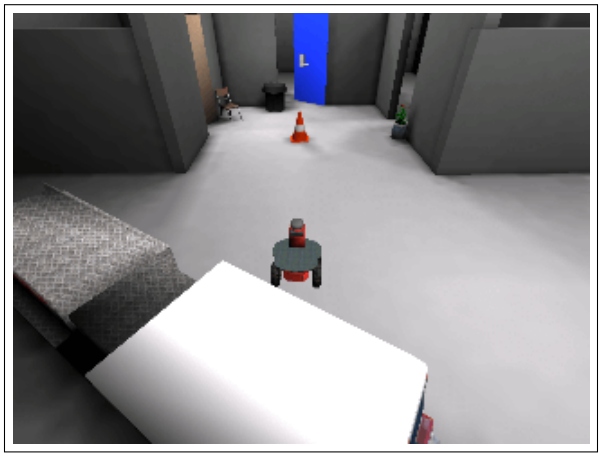


- "Go to the box"
  **Scene 7:** Referential ambiguity that can be resolved by intrinsic property (one box is much larger than the other), proximity to a nearby object (one box is by a lamp), and spatial region (one box is to the left of the agent and one to the right of it). **(IPS-G)**



- "Go to the truck"
  **Scene 8:** No situated grounding problem (control for participant). Only one truck exists in the building. **(CTRL)**

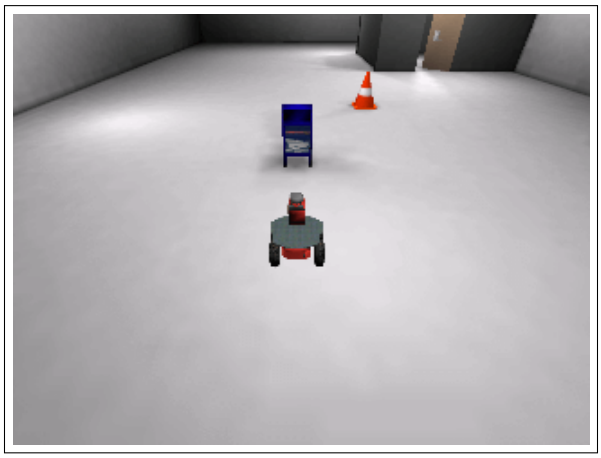
- "Move to the cone by the desk"
  **Scene 9:** Impossible-to-execute instruction: proximity restriction (while there is a cone in the building, it is not near a desk). **(IMP)**



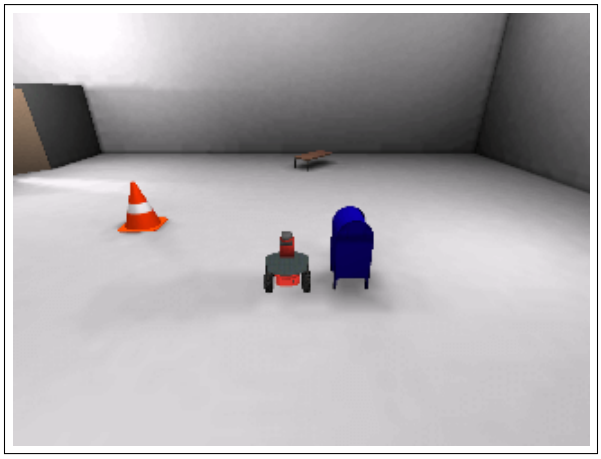
- "Move to the door"
  **Scene 10:** Referential ambiguity that can be resolved by intrinsic property (one door is blue, one brown, and one purple), proximity to a nearby object (one door is by a chair, another by a bin, another by a computer tower), and spatial region (one door is right of the agent, one to its left, and one in front of it). **(IPS-C)**

### Building 3: 5 Scenes

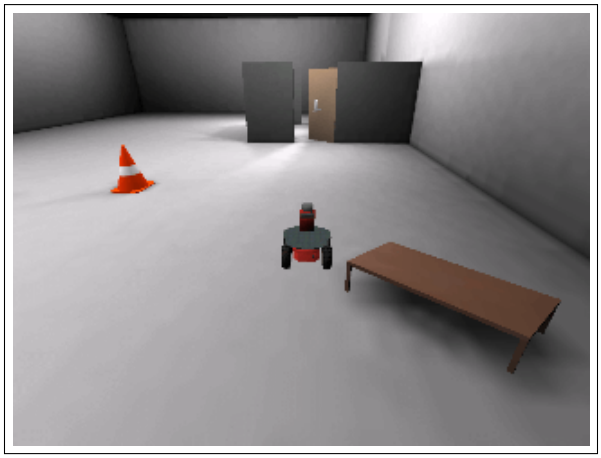
- "Move to the mailbox"
  **Scene 11:** No situated grounding problem (control for participant). Only one mailbox exists in the building. **(CTRL)**



- "Move through the table"
  **Scene 12:** Impossible-to-execute instruction: action-ontology mismatch (the action is not possible because the agent cannot move through the table). A table cannot be moved through. **(IMP)**



- "Move to the door by the vendor"
  **Scene 13:** No situated grounding problem (control for participant). Only one door is viewable, and it is open. **(CTRL)**

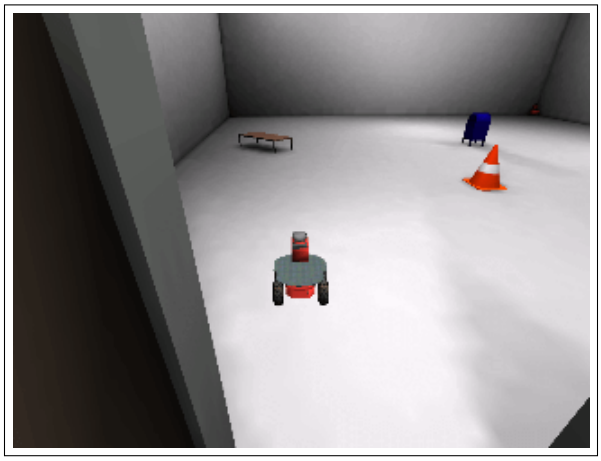


- "Go through the door"

  **Scene 14:** Referential ambiguity that can be resolved by referring to an object in its recent history (the agent just went through a door), proximity to a nearby object (one door is by a vendor, the other by a lamp), and spatial region (one doorway is in front of the agent, and one behind it). **(HPS)**

- "Go to the blue cone"

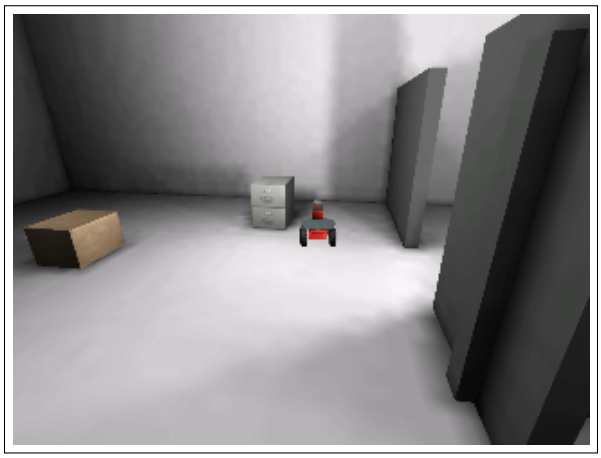
  **Scene 15:** Impossible-to-execute instruction: intrinsic property restriction (although there is a cone in the building, there are no blue ones). **(IMP)**



### Building 4: 5 Scenes

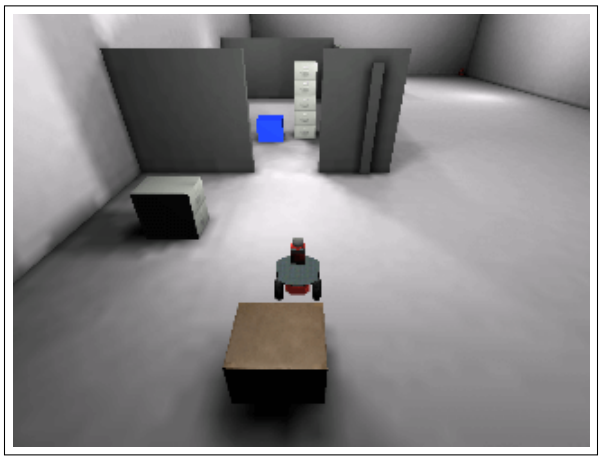
- "Go to the cabinet on the left"
  **Scene 16:** No situated grounding problem (control for participant). Only one cabinet is referrable. **(CTRL)**

- "Move to the box in front of you"
  **Scene 17:** Impossible-to-execute instruction: spatial restriction (there is no box in front of the agent). **(IMP)**



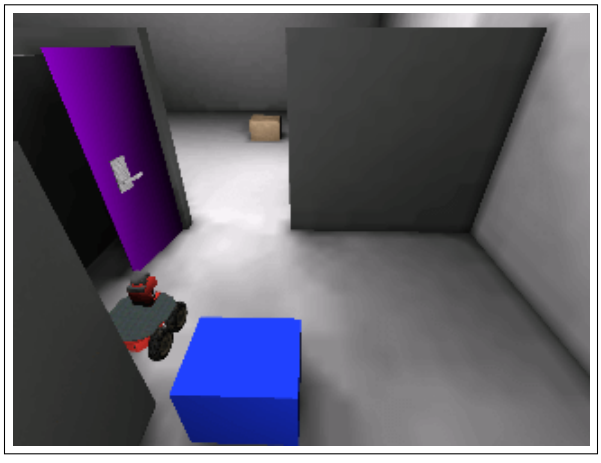
- "Move to the cabinet"

  **Scene 18:** Referential ambiguity that can be resolved by intrinsic property (size; one cabinet is much larger than the other), referring to an object in its recent history (the agent went to a cabinet already), and spatial region (one cabinet is to the left of the agent, one in front of it). **(IHS-G)**

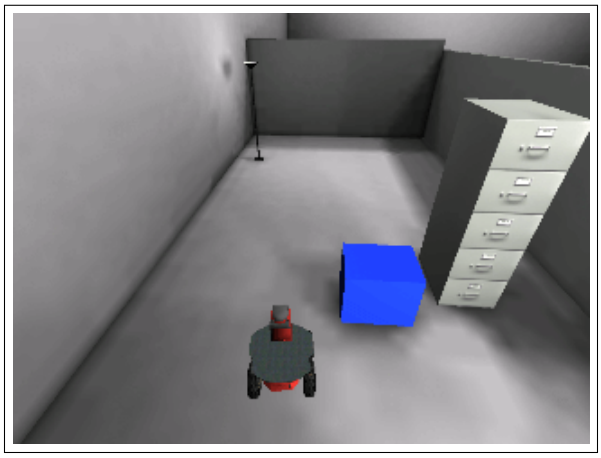


- "Move to the box"

  **Scene 19:** Referential ambiguity that can be resolved by intrinsic property (color; one box is brown, the other blue), referring to an object in its recent history (the agent went to one box already), proximity to a nearby object (one of the boxes is by a cabinet), and spatial region (one box is in front of the agent, one on its right). **(IHPS-C)**



- "Move to the black lamp"
  **Scene 20:** No situated grounding problem (control for participant). Only one lamp is in the building. **(CTRL)**

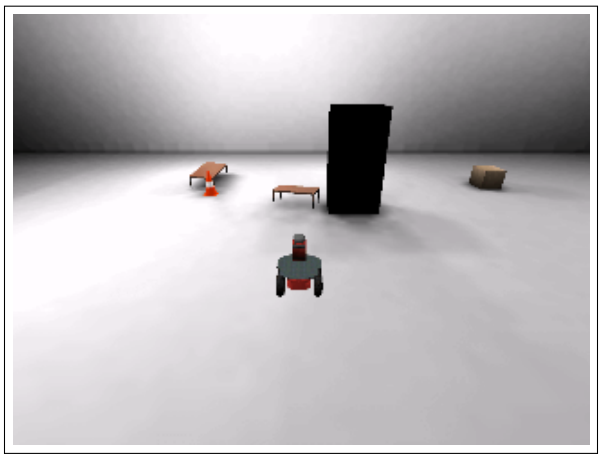


### Building 5: 5 Scenes

- "Move to the table"
  **Scene 21:** Referential ambiguity that can be resolved by intrinsic property (size; one table is larger than the other), proximity to a nearby object (one of the tables is by a vendor, the other by a cone), and spatial region (one table is in front of the agent, one is to its right). **(IPS-G)**

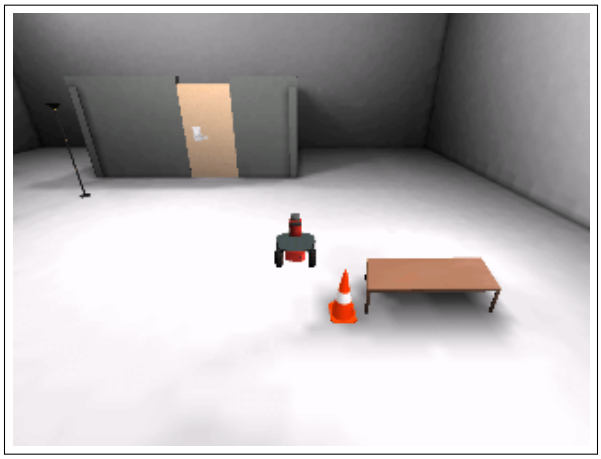
- "Move to the table"

  **Scene 22:** Referential ambiguity that can be resolved by intrinsic property (size; one table is much larger than the other), referring to an object in the agent's recent history (the agent just went to one table as a subgoal), and proximity to a nearby object (one of the tables is by a snack machine, the other by a cone). **(IHP-G)**



- "Move through the door"

  **Scene 23:** Impossible-to-execute instruction: path not possible by object state (the door is closed). **(IMP)**



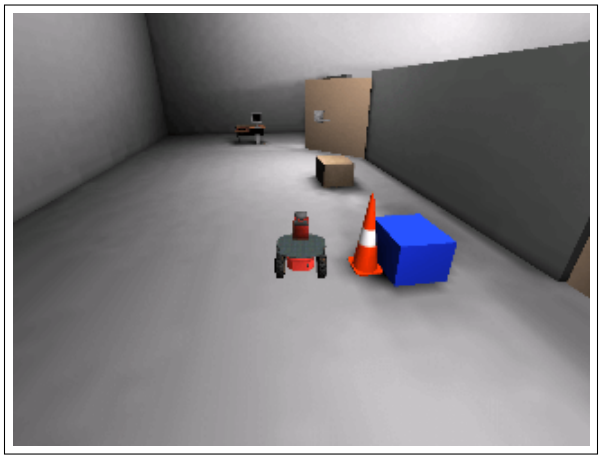
- "Go to the cabinet by the lamp"
  **Scene 24:** Impossible-to-execute instruction: proximity restriction (there are no boxes by lamps). **(IMP)**

- "Go to the black lamp"
  **Scene 25:** No situated grounding problem (control for participant). There is only one lamp in the building. **(CTRL)**

### Building 6: 5 Scenes

- "Move to the box"
  **Scene 26:** Referential ambiguity that can be resolved by intrinsic property (the boxes are different colors) and proximity to a nearby object (one of the boxes is near a cone). **(IP-C)**

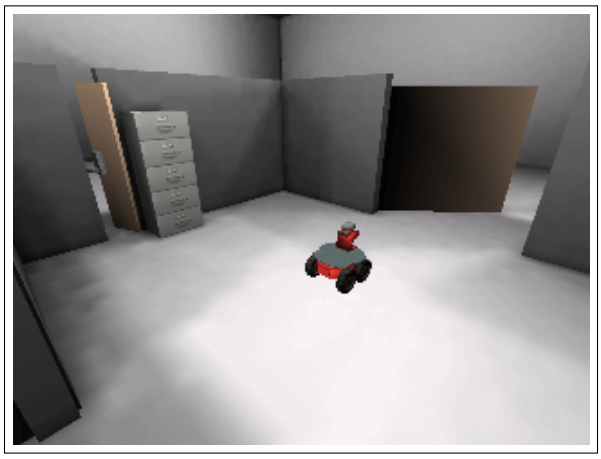
- "Go to the computer"
  **Scene 27:** No situated grounding problem (control for participant). Only one computer is in the building. **(CTRL)**



- "Go through the door on the left"
  **Scene 28:** Impossible-to-execute instruction: spatial restriction (no door is on the agent's left). **(IMP)**

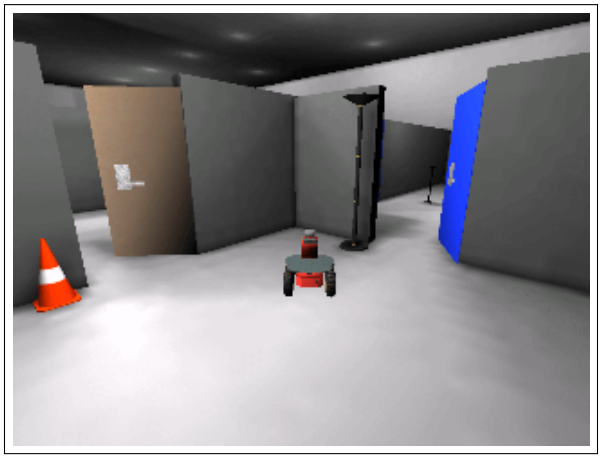
- "Move to the table"
  **Scene 29:** Impossible-to-execute instruction: path not possible by obstruction (there is a roadblock preventing the agent from reaching the table). **(IMP)**

- "Go to the door"
  **Scene 30:** Referential ambiguity that can be resolved by intrinsic property (size; one door is narrow, the other wide), referring to an object in the agent's recent history (the agent passed through one door already), and proximity to a nearby object (one door is by a cabinet). **(IHP-G)**



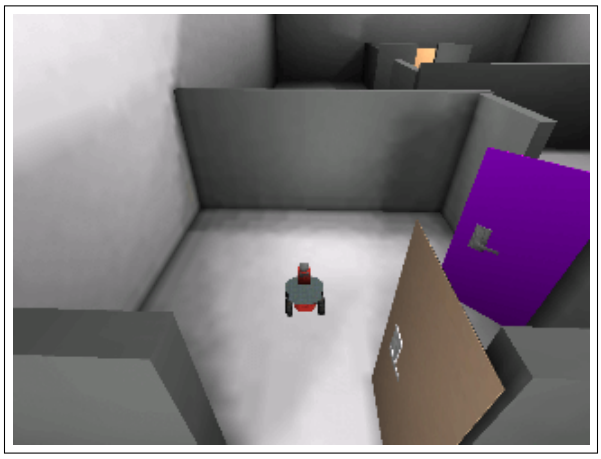
### Building 7: 5 Scenes

- "Move through the door"
  **Scene 31:** Referential ambiguity that can be resolved by intrinsic property (color; the doors are different colors) and proximity to a nearby object (one door is by a cone, and one is by a lamp). **(IP-C)**



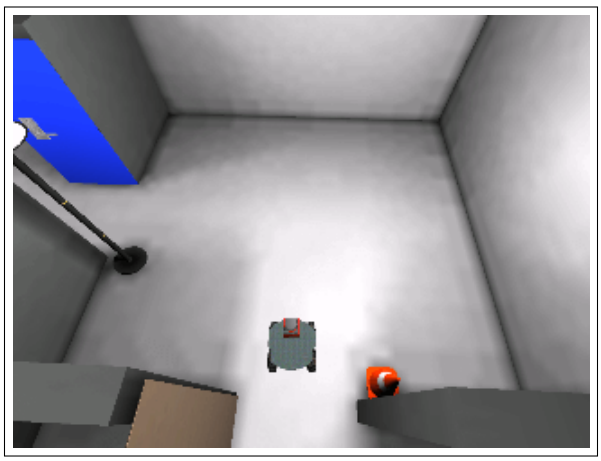
- "Move through the door"

  **Scene 32:** Referential ambiguity that can be resolved by intrinsic property (color; the doors are different colors), referring to an object in its recent history (the agent just passed through one door), and spatial region (one door is to the right of the agent and one behind it). **(IHS-C)**



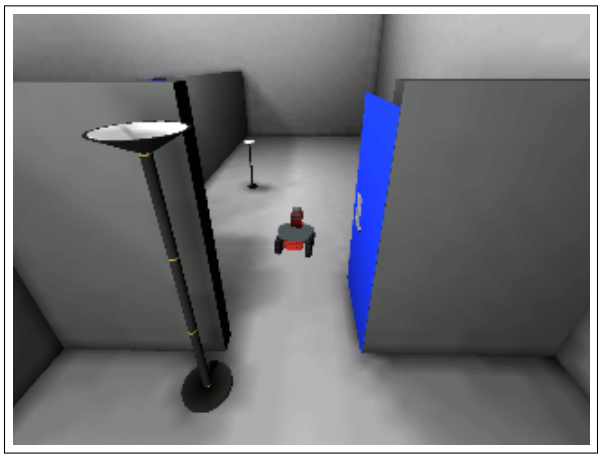
- "Move through the door"

  **Scene 33:** Referential ambiguity that can be resolved by intrinsic property (color; the doors are different colors), referring to an object in the agent's recent history (one door was just passed through), proximity to a nearby object (one door is near a lamp, the other near a cone), and spatial region (one door is to the left of the agent, and one behind it). **(IHPS-C)**



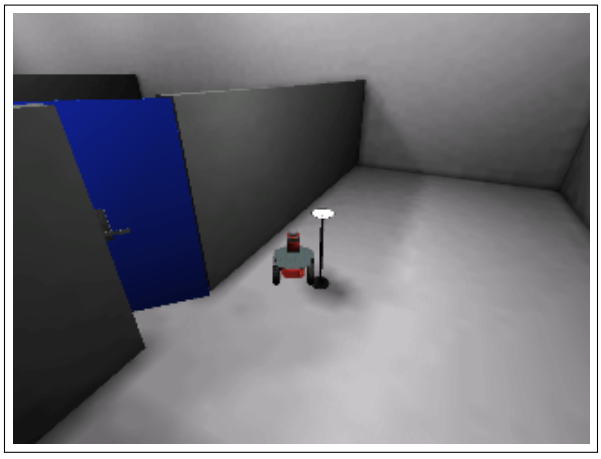
- "Move to the black lamp"

  **Scene 34:** Referential ambiguity that can be resolved by intrinsic property (size; the lamps are different sizes), referring to an object in the agent's recent history (the agent passed one of the lamps already), and spatial region (one lamp is in front of the agent, one behind it). **(IHS-G)**



- "Move to the mailbox"

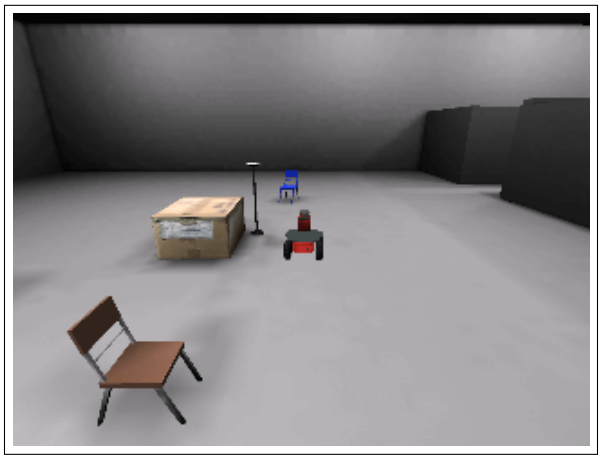
  **Scene 35:** Impossible-to-execute instruction: No object of that type exists (there is no mailbox in the building). **(IMP)**



### Building 8: 5 Scenes

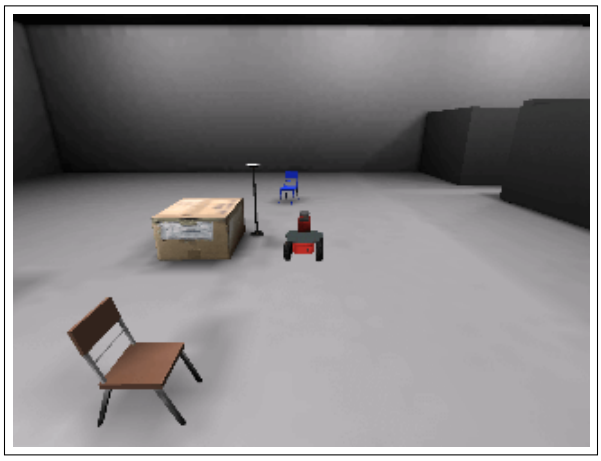
- "Go to the box"
  **Scene 36:** Referential ambiguity that can be resolved by intrinsic property (size; one box is larger than the other), proximity to nearby objects (one box is by a lamp, the other is by a cabinet), and spatial region (one box is to the left of the agent and one in front of it). **(IPS-G)**

- "Go to the lamp behind you"
  **Scene 37:** Impossible-to-execute instruction: spatial restriction (there is no lamp behind the agent in the current context). **(IMP)**



- "Move to the chair"

  **Scene 38:** Referential ambiguity that can be resolved by intrinsic property (the chairs are different colors), referring to an object in the agent's recent history (the agent started at one of the chairs), and spatial region (one chair is behind the agent, one in front of it). **(IHS-C)**



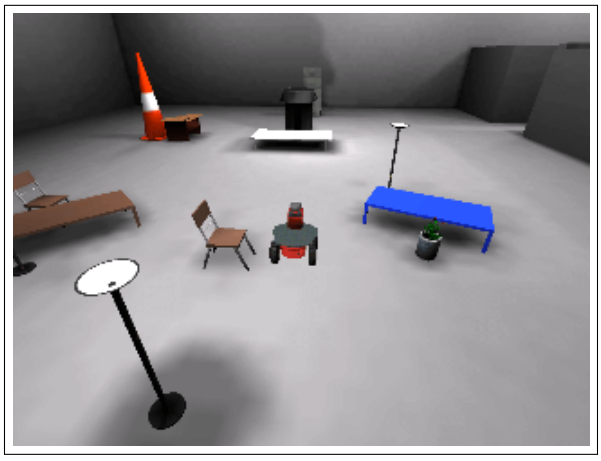
- "Move through the door"

  **Scene 39:** Impossible-to-execute instruction: path not possible by object state (the only possible doorway is closed). **(IMP)**

- "Move to the hydrant"
  **Scene 40:** No situated grounding problem (control for participant). Only one hydrant is in the building. **(CTRL)**

### Building 9: 5 Scenes

- "Move to the table"
  **Scene 41:** Referential ambiguity that can be resolved by intrinsic property (color; the tables are different colors), proximity to a nearby object (the tables are by different objects), and spatial region (one table is to the right of the agent, one to its left, and one in front of it). **(IPS-C)**



- "Move to the chair"

  **Scene 42:** Referential ambiguity that can be resolved by referring to an object in the agent's recent history (the agent started near a chair), proximity to a nearby object (one of the chairs is by a table), and spatial region (one in front of the agent, and one is behind it). **(HPS)**

- "Move to the black lamp"

  **Scene 43:** Referential ambiguity that can be resolved by referring to an object in the agent's recent history (the agent encountered one of the lamps already), proximity to a nearby object (one lamp is by itself and the others are near different objects), and spatial region (one lamp is behind the agent, one in front, and one to its right). **(HPS)**

- "Move to the bin"

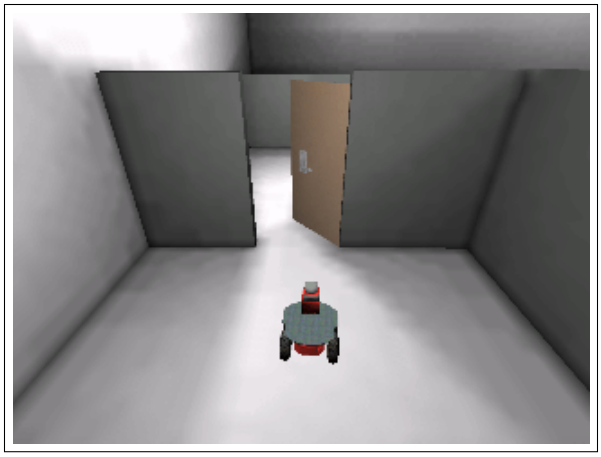
  **Scene 44:** Referential ambiguity that can be resolved by intrinsic property (size; one bin is larger than the other), referring to an object in its recent history (the agent encountered one bin already), and proximity to a nearby object (the bins are near different objects). **(IHP-G)**

- "Go to the cone"

  **Scene 45:** Referential ambiguity that can be resolved by intrinsic property (size; one of the cones is smaller than the other), proximity to an object (the cones are by different objects), and spatial region (one cone is to the left of the agent and one to its right). **(IPS-G)**

### Building 10: 5 Scenes

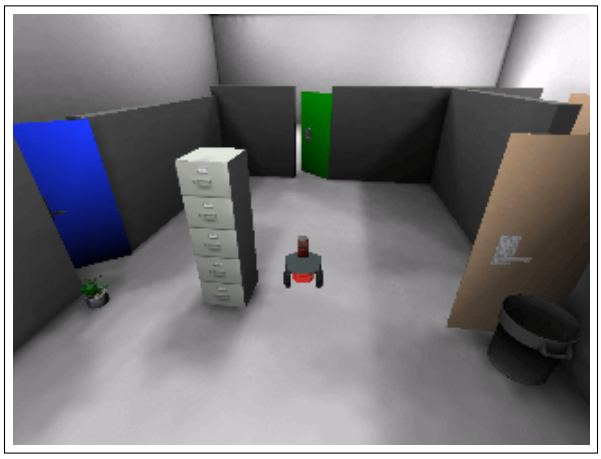
- "Go through the door on your left"
  **Scene 46:** Impossible-to-execute instruction: spatial restriction. There is no door to the left of the agent in the current context. **(IMP)**



- "Move to the cabinet"
  **Scene 47:** No situated grounding problem (control for participant). Only one cabinet is in the building. **(CTRL)**

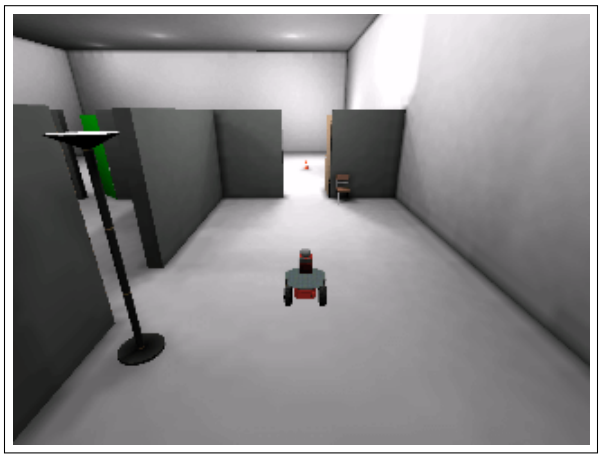
- "Move through the door"

  **Scene 48:** Referential ambiguity that can be resolved by intrinsic property (the doors are different colors), proximity to a nearby object (the doors are near different objects), and spatial region (one of the doors is to the left of the agent, one to its right, and one in front of it). **(IPS-C)**



- "Move through the door"

  **Scene 49:** Referential ambiguity that can be resolved by referring to an object in the agent's recent history (the doorway that was just passed through), proximity to a nearby object (the doorways are by different objects), and spatial region (one doorway is in front of the agent, one is to its left). **(HPS)**



- "Move to the cone"

  **Scene 50:** Referential ambiguity that can be resolved by intrinsic property (one of the cones is larger than the other), proximity to a nearby object (one of the cones is by a mailbox), and spatial region (one of the cones is in front of the agent, and one to its left). **(IPS-G)**

# Appendix B: Thesis Experiment Script

Before the participant arrives, the primary experimenter will prepare the necessary forms for the experiment. These forms include the 2D maps of the environments, guides on what to say, and scratch paper. Upon arrival at the experiment each participant will be greeted as follows:

> *Welcome to the Talking with Robots study. Please have a seat.*

Next, the primary experimenter will describe his role.

> *Alright, let's get started. I'm going to be your experimenter. I'll be your point of contact here. You can ask me any questions you want, though I may or may not have an answer depending on whether it will impact what we're testing in this study.*

**Introductions**

The primary experimenter will describe the scenario and introduce the task.

> *In this experiment you will be providing verbal instructions intended for a robot. Here is the robot Alphie you will be working with (shows the Pioneer virtual robot).*
>
> *The robot can understand verbal instructions and requests such as moving to objects and turning. For the robot to understand you properly, please speak clearly at a normal pace and do not exaggerate your speech, even if the robot did not understand what you said.*
>
> *Before the start of the task, you should say just the word "Alphie" to get the robot's attention.*

**Training exercise (Practice map)**

*We are going to go through a training exercise with the robot, so you can see the kinds of things it can understand. Right now you'll see a 3D map with some objects and the robot. For this training exercise, your job will be to get the robot to move to all the different places shown on this map (hands 2D map to participant).*

*In this scenario, you need to determine whether certain objects of interest are still present in the robot's environment. Your primary objective is to verify whether the objects in the environment are there or have moved as per the robot's map. You'll be using the robot to explore the space and locate objects of interest.*

*The robot may have the same map as you or a different one, or the same map. The robot may see what you don't see and fail to see what you can see. Some service disruption has caused the speech channel to be noisy, so the robot will not always understand you correctly. But it will say as much as it understood.*

*The robot will report to you about objects of interest as you navigate. The map has keywords you should use to refer to objects in the robot's environment.*

*As you have the robot traverse the map, we will notate whether or not objects are present or missing.*

*Also, when choosing from a list, you don't need to give the whole command again, just a choice from the list. You can say "Cancel" if none of the choices are valid.*

Participant is now ready to begin training.

*Any questions before we begin?*

**Landmark navigation task (Maps 1-10)**

The primary experimenter will then introduce the main scenario:

*Your task is to visit these objects in order. Use the exact object names we provide. The robot knows these names and where some (but not all) of the objects are located. Those names can be found on this sheet (hands Labeled Object List to participant). The robot moves at a constant speed.*

*Note that the robot may not always be "at" the object of interest, but it will try to get itself as close as possible. If the robot is nearer to that location than any other, that's its closest spot.*

*Also, we discourage use of specific distances, use commands that mention the objects themselves.*

*After you complete verifying all of the objects of interest in one location, let me know. I'll place the robot on a different map.*

*Again, when choosing from a list, you don't need to give the whole command again, just a choice from the list. You can say "Cancel" if none of the choices are valid.*

Participant is now set to begin the study.

*Any questions before we begin?*