SPATIAL LANGUAGE DRIVEN ROBOT

A Dissertation

Presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

ZHIYU HUO

Dr. Marjorie Skubic, Dissertation Supervisor

JULY 2017

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

SPATIAL LANGUAGE DRIVEN ROBOT

presented by Zhiyu Huo, a candidate for the degree of doctor of philosophy, and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Marjorie Skubic

Professor James Keller

Professor Zhihai He

Professor Mihail Popescu

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation to my adviser, Dr. Marjorie Skubic, for her valuable guidance and advice and for her vast reserve of patience and knowledge.

I would like to also express my sincere thankfulness to my colleague Tatiana Alexenko, and corporators Dr. Laura Carlson, Dr. Xiaoou Li, Dr. Jared Miller who helped me with my research.

Finally, I would like to acknowledge my family and friends for their encouragement and devotion.

Table of Contents

List of Tables

List of Figures

ABSTRACT

This dissertation investigates the methods to enable a robot to interact with human using spatial language. A prototype system of human-robot interaction using spatial language running on an autonomous robot is proposed in the dissertation. The system includes two complementary works. One is to control the robot by human natural spatial language to find the target object to fetch it. Another work is to generate a natural spatial language description to describe a target object in the robot working environment. The first task is called spatial language grounding and the second work is named as spatial language generation. The spatial language grounding and generation are both end-to-end process which means the system will determine the output only by the natural language command from a human during the interaction and the raw perception data collected from the environment. Furniture recognizers are designed for the robot to detect the environment during the tasks. A hierarchy system is designed to translate the human spatial language to the symbolic grounding model and then to the robot actions. To reduce the ambiguity in the interaction, a human demonstration system is designed to collect the spatial concept of the human user for building the robot behavior policies under different grounding models. A language generation system trained by real human spatial language corpus is proposed to automatically edit spatial descriptions of the location of a target object. All the modules in the system are evaluated in the physical environment, and a 3D robot simulator developed on ROS and GAZEBO.

Chapter 1. Introduction

1.1 Motivations

For a social robot designed as an assistant for the daily life of its human user, a fundamental skill is to interact with humans by spatial language. Spatial language is a kind of human language that contains spatial information ([1]). The usages of spatial language include describing locations or using spatial relationships to identify targets. For example, in the scenario of a cocktail party, a guest is asking the host to identify a VIP. The host can describe the VIP by the gender, the dress or the age, or just use spatial language to tell the guest that the VIP is the man standing by the round bar table beside the door. Here, spatial language can help the guest to lock in on the VIP quickly even though there may be several potential VIPs in the room.

Human use of spatial language is a complex activity which relies on both linguistic and non-linguistic representations and processes. This creates a great challenge in the interaction between humans and robots. Consider two scenarios of using spatial description on the same target object, i.e., a cellphone. The instructions are: (a) "*Go down the hallway, then, turn right. Walk forward and then turn left. You will find the cellphone.*" (b) "*The cellphone is on the table to the left of the bed in the bedroom.*" For sentence (a), the addressee was given a sequence of actions from the spatial command to find the cellphone, while in (b) the addresser gave a description of the cellphone position by using references based on advanced knowledge (the addressee knowing where the bedroom is). Both (a) and (b) are spatial descriptions based on human memory or learned associations, which can make it difficult for a robot when it takes on a human position to complete a task and

process the command language. There are four main factors as described below, which have proven to be problematic for human-robot spatial language interaction.

(1) **Quantitative Expression vs. Qualitative Expression**: In both descriptions, the predicates used qualitative spatial references (such as “down, right, forward” with verbs) rather than quantitative terms to describe an action or a position. Although it is natural to people to use qualitative expressions, a robot can only “think” in terms of mathematical expressions and numbers which creates a gap between human and machine [2, 3].

(2) **Ambiguous in Perspective**: The directional description (“down, right, forward”) depends upon the reference frame that defines the terms, which could be based on the perspective of the speaker or addressee, the orientation of the room, or the axes of the reference objects (e.g., couch and table) [4, 5]. Apart from the diversity in reference selections, ambiguity is another challenge. Selecting a reference frame must also allow for reinterpretation if the initial selected frame is incorrect [4, 5].

(3) **Environment Modeling**: There must be a prioritization of the many possible features that define a good reference object or beacon, requiring integration of conceptual, functional, perceptual, and spatial information. Interpretations cannot be based strictly on geometric information as assumed from a traditional viewpoint [6]. However, using traditional approaches, i.e., laser, sonar or stereo camera, for environment reconstruction can only produce a simple and rough map exclusively for 2D navigation. The information needed for spatial language interaction by a robot should include the class, dimension, pose and shape of possible reference objects and beacons, which is difficult to register by traditional methods.

(4) **Common Ground and the Cognitive Burden**: A human speaker and an addressed robot can hardly agree on the complexity of the context and environment. A human user may give a spatial description, which is beyond the robot's capability of processing [7].

A key motivation of this research was to create a more natural interface for human-robot interaction in spatial language. The goal was to provide robots with the ability to produce and comprehend these spatial descriptions through integration with more general cognitive characterizations that include reference frames, reference object features, complexity, and speaker/addressee assumptions. These ideas were tested within several home scenes to illustrate their real-world applicability. Human subject experiments were run with both young and elder participants to ensure generalizability. It is anticipated that the products of this research will be applicable to many assistive settings.

The methods presented in this dissertation build a closed-loop workflow of human-robot interaction using spatial language. The loop can be divided into two paths that represent two complimentary robot tasks in a home-like environment. One is a natural language directive for the fetch task. In this task, a human user orders a robot to fetch a target object by giving a spatial command. Another is spatial description generation, which lets a robot answer to a human user with the location of a target object by using natural spatial language. To support the first task, a robot needs to have the capability to ground the human addressed natural spatial command to robot behavior and then execute actions in its working environment to find it. For the second task, a robot needs to have the capability of natural spatial description generation by the information it collects from the working environment. The functions for the two tasks were developed and integrated into

a Kinect-based differential-drive autonomous robot, which represents a good prototype of an eldercare and household agent.

1.2 Primary Goal

The principle objectives of the research that formed this dissertation include the following:

(1) To build a generalized natural language processing (NLP) model, which uses spatial language in an in-room environment.

(2) To design a prototype autonomous social robot with the capacity of navigation and perception in an in-room environment. The perception system should contain an environment model of a home environment which can match the human's home with an accurate understanding of the spatial relations in his or her environment.

(3) To develop a natural language grounding system that allows the robot to follow the spatial description given by a human user to perform a fetch task in an indoor environment.

(4) To develop a natural language generation system that enables the robot to give a human-like description on the position of a target object for a missed object as part of the searching task in an indoor environment.

1.3 Research Overview

A prototype autonomous robot system was demonstrated and assessed in this proposal. The capabilities of the system include: recognizing and processing a natural spatial language command given by the human user, programing robot behavior to follow the command, environment modeling for navigation in a home environment, and the generation of a natural spatial description answer for the human inquirer.

The work started from a human experiment conducted by our collaborators, Dr. Laura Carlson, Dr. Xiaoou Li, Dr. Jared Miller from Notre Dame to discover the usage of spatial language in an indoor environment [8, 9]. The scenario studied was a home setting in which the elderly resident had misplaced an object, such as eyeglasses, and the robot helped the resident find the object. A corpus of spatial descriptions, the Carlson-Skubic Indoor Spatial Language (***CSISL)*** corpus, was collected and reorganized from both young subjects and elder subjects. To explore the spatial language used in the context of the fetch task and investigate generational understanding of language, a set of human subject experiments was planned, studying both young and elder subjects. Experiments were conducted in a virtual environment (VE), which provided a controlled setting that made it easier to capture potentially subtle metrics between test conditions. In each test, a subject explored the virtual house with the assistance of an experimenter and found a designated target. Then they were asked to provide a spatial description. This experiment resulted in the CSISL having 1024 spatial commands, which were then summarized and transferred to a 149-command corpus. The key features of the spatial descriptions were summarized, including their dynamic versus static nature and the perspective adopted by the speaker. The critical cognitive and perceptual processing capabilities necessary for the robot were also investigated from the collected language samples [7, 10, 11].

A system which enables spatial language interaction between a human and robot was assessed on an autonomous robot platform. The robot was designed to match the requirements based on capability of perception and locomotion, and the required interaction for each service task. These tasks included but were not limited to human-machine dialog (voice recognition and speech generation) as well as fetching and searching

for missed objects. The P3DX robot was chosen as the chassis part to give the robot an adequate load capability as well as maneuverability. The sonar array which surround the upper edge of the chassis, provide a close obstacle detection to prevent collision. The laser range finder on the top of the chassis provides a more accurate estimation of the obstacle objects. A Kinect camera worked as a main sensor placed 1.05 m from the floor. The Kinect camera was sustained by a metal frame structure on top of the chassis. The robot software was developed on the robotic operating system (ROS) and ran on an HP laptop.

Using a robot in a home environment illustrates challenges in object recognition and environment modeling. The work was done to assure the success of an autonomous human-robot interaction using spatial language. To allow the robot to follow spatial language commands and execute tasks such as fetching objects, the recognition goal was not only to recognize objects by labeling their names but to find detailed geometry features that could help the robot obtain the spatial relationships between objects. I built a fuzzy classifier using RGB and depth data on a restricted number of furniture items in the M.S. study [12]. However, this classifier had a low accuracy when the robot was in some viewing locations and had an unsatisfying performance for another furniture database. In addition, the processing speed of the fuzzy classifier was too slow to work on real-time spatial relations problems. During my PhD study, my team and I developed an innovative method to recognize furniture items and their dimensions, locations and orientations by using depth data only. In this classification model, the depth data were structured to generate point cloud normal vectors in real time, which were then used as features for recognition. The real-time modeling approach provided a higher accuracy on furniture recognition. Also, it allowed the robot to work in a room with rearranged furniture placement.

The robot platform was integrated with the perception system to process robot fetch tasks driven by natural language. An innovative natural spatial language grounding system was proposed to process natural spatial language and interpret it to a robot's understandable navigation instructions to initiate actions [13, 14]. The system's first process used spatial referencing language according to the part-of-speech (POS) and extracted a tree structure of language chunks, which was conducted by my collaborator, Tatiana Alexenko [15]. This step employs a semantic spatial language grammar and a novel chunking method that allows nested structures to be encoded as a single label. The semantic grammar is based on an interdisciplinary analysis of a corpus of human-generated indoor spatial language. A "deep" chunking method facilitates encoding deep grammatical structures into a single-level label. After obtaining a tree structured spatial command, the spatial language description was then grounded to a robot navigation instruction in the form of a sequence of actions which referred to as the reference-direction-target (RDT) model. This model was based on spatial references to furniture and room structure. Furthermore, the best navigation instruction was selected by scoring in a probabilistic model. The initial form of the RDT node was proposed in the M.S. research and was further developed when we finished the probabilistic model for grounding. To control the robot for the fetch task, a behavior model was designed based on the RDT model. The policy model of robot behavior was built by robot learning, which implements programming by demonstration (PbD). An interface was designed to provide a mapping between the robot state, which includes the spatial command, and the environment state, as a means to robot action. The natural language processing and grounding both work together to enable a robot to follow spatial language commands in a physical indoor environment.

The last task in completing the dissertation research was natural spatial language generation, which is the opposite of spatial language following tasks such as fetching [16, 17] The language generation task described the location of a target object found by a robot. The system generates a spatial description in three steps. First, the robot searches and finds the target object indicated by the human user. Second, an RDT grounding model is inferred to best match that location. Finally, the grounding model is converted to a natural spatial description. Structure prediction is employed to train the mapping from robot state domain to a grounding model domain and ultimately to a natural language domain.

Several contributions were proposed during the Ph.D. study, which include:

- A framework of an end-to-end system for human-robot interaction using spatial language. The framework includes:
    - A furniture detection system which can detect the category and pose of the furniture items in the indoor environment.
    - An environment model which extracts the information of the spatial relationships between objects in the robot's working space.
    - A spatial language model which can parse a natural spatial description to: 1) a tree structure using part-of-speech tagging; 2) a reference-direction-target grounding model.
    - A robot behavior model which can navigate a robot to move to the target place by the grounding model and the environment model.
    - A spatial language generation system which can let the robot to edit the language to describe the position of a target object.

- A 3D robot simulator which provides a virtual environment to evaluate the system by end-to-end tasks.

The components of the human-robot spatial language interaction system are shown in Figure 1.1

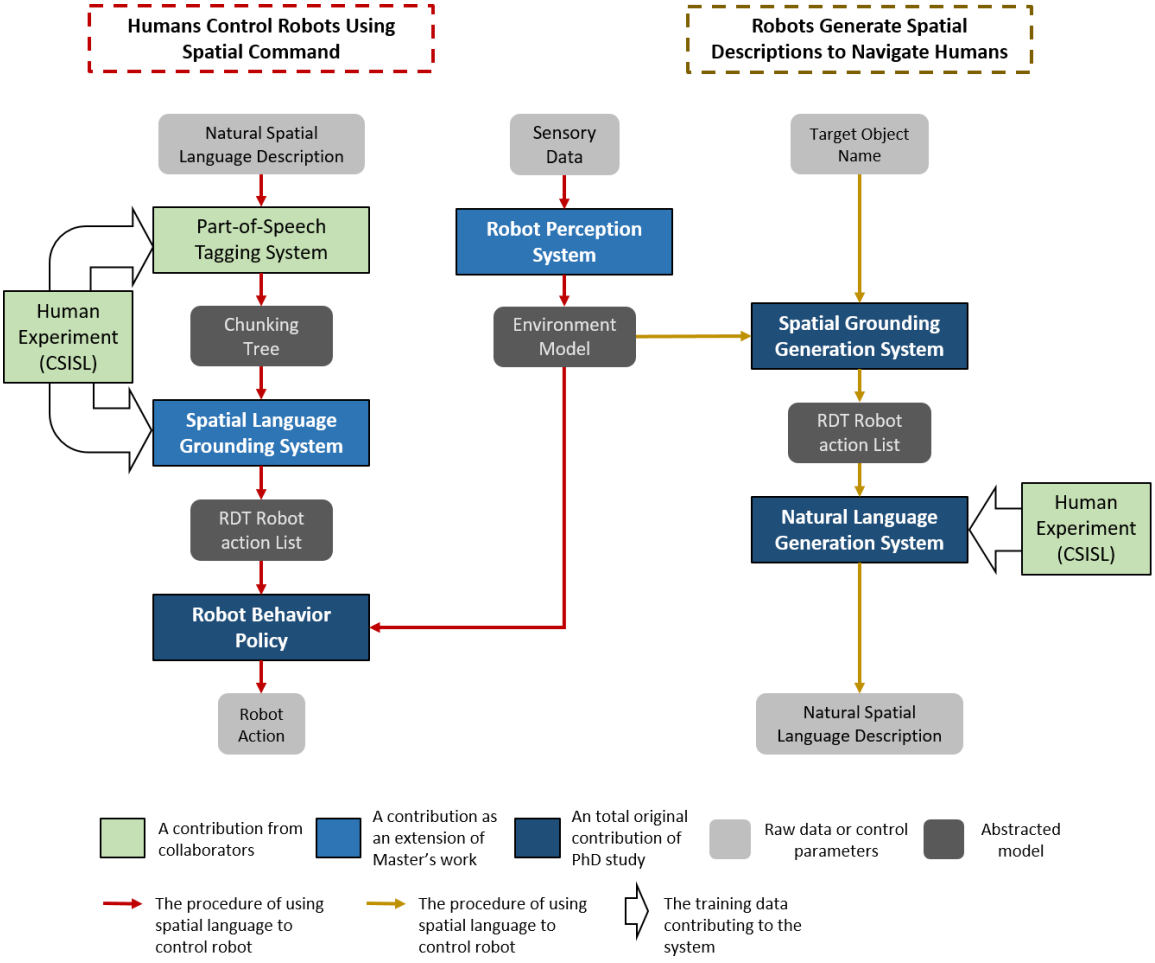


Figure 1.1 The diagram of two complementary works for the research of human-robot interaction using spatial language. The left side supports human control of robots using spatial commands; the right side provides robot-generated spatial descriptions to navigate humans.

1.4 Organization

Chapter 2 is a review of the results of the human spatial language experiment which was conducted by the author and collaborators. This chapter introduces the approach used

to edit a template of spatial descriptions for an indoor environment through a human subject experiment.

Chapter 3 introduces the robotic platform used to research a spatial language-driven robot. The hardware components and software platform are listed. It also presents an entity model to encode the home environment, which provides sufficient information for the robot to achieve the spatial language interaction task. The sensory information was collected by Kinect. A new furniture recognition system is proposed which is faster and more accurate than the one used in the M.S. research. The objectives of the system include furniture classification, dimension modeling, and furniture orientation regression. Both SVM and deep network object detection models are tested and compared on different databases in this chapter.

Chapter 4 demonstrates a method to ground a human spatial language command to a robot navigation instruction model referred to as the reference-direction-target (RDT) model and then delineates robot control factors for a robot fetch task. It builds a bridge between natural spatial language commands and robot behavior. This chapter includes a published paper and an add-on section of a grounding model built from a recurrent neural network (RNN).

Chapter 5 proposes an approach that builds the policy model for a mobile robot to follow a human's spatial commands. The system implements programming by demonstration (PbD) to develop policies for different robot spatial commands. This chapter is also a draft paper prepared for the Journal of Human-Robot Interaction.

Chapter 6 proposes a natural spatial language generation system. It is an inverse of the language grounding work. The system was validated by both statistical metrics and a human scorer. The chapter consists of a conference paper and a section of additional results.

Chapter 7 is the conclusion and contributions.

Since the content of each chapter is mostly independent from the others, I include an introduction of the related literature into the respective chapters but have not composed them into a single chapter.

Chapter 2. Background on the Human Spatial Language Model

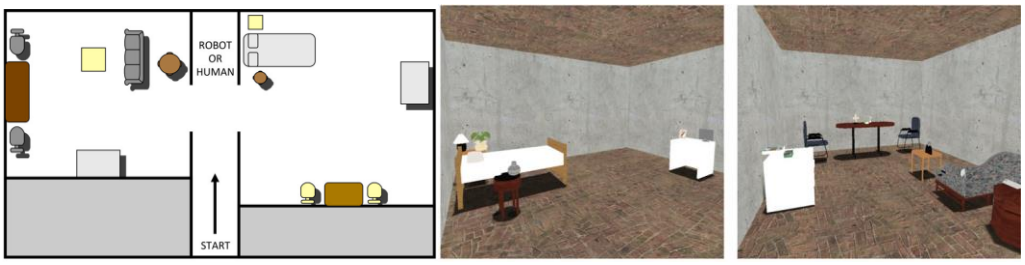
The research in this section is conducted by a multi-discipline research group from the University of Missouri and the University of Notre Dame. The contributions of this work include: (1) Collect a human spatial language corpus; (2) Build a spatial language template; (3) Find the difference between elder adults and younger adults on using spatial language.

2.1 Introduction

Human comprehension of spatial language is a complex activity. Consider an utterance: "*Your eyeglasses are behind the radio on the table in the bedroom.*" Although human interpretation of such instructions normally proceeds naturally and fluently, the comprehension of spatial descriptions is particularly problematic for robots. While it is possible to train a speaker to restrict robot directives to a set of constrained commands (i.e., make the user adapt to the robot), the intent in this project instead is to explore how the robot can adapt to the human user, with all of the ambiguities and complexities inherent in natural language. In this section, these complexities will be addressed by collecting a corpus of spatial descriptions elicited within a 3D virtual setting in the context of a fetch task.

2.2 The spatial description corpus

The corpus of spatial descriptions was collected within an eldercare scenario in which a participant navigates through a virtual 3D house environment (Figure 2.1) to find a target, and then provides spatial descriptions that specify the target's location to an avatar in the context of a fetch task.



(a) (b)

Figure 2.1 Figure (a) shows a survey perspective of the house environment that contains a living room on the left, a hallway in the middle, and a bedroom on the right. A trial within the virtual environment started at the location in the hallway indicated by "start" with the perspective indicated by the arrow, facing the robot or human avatar addressee. Figure (b) shows part of the living room and bedroom at the eye-level height adopted in the experiment.

Three critical aspects of the scenario were the focus of this research. First, research has shown an increased reliance on landmarks during wayfinding by older adults [18]. Thus, the robot strategies had to recognize the furniture objects and also note their orientations, which were then used during ambiguity resolution. Second, within the virtual environment, participants offered spatial descriptions to either a human avatar named Brian or a robot avatar modeled after a real-life robot. Third, the instructions were manipulated by using these two prompts:

Where prompt: Tell (Brian/the robot) where the <target> is (e.g., "*The cell phone is on the table by the bed in the bedroom*")

How prompt: Tell (Brian/the robot) how to find the <target> (e.g., "*Go forward, turn right, look to the left...*")

2.2.1 Methods

Subjects: Sixty-four older adults participated in the experiment. For each trial, participants were shown a picture of the to-be-found target on a gray background. The experiment monitor named the target to ensure full identification. Participants started in the hall and

told the monitor how to navigate to find the target. They were allowed to search as long as necessary to discover the target; the procedure was not timed. No trials were excluded due to an inability to locate the target. Participants were then returned to the starting location, received their assigned "where" or "how" prompt, and then described the location of the target to the monitor; these descriptions were recorded. Finally, after completing all trials, participants drew a map of the house that was coded to verify that participants had an accurate representation of the environment in terms of the layout of the rooms.

2.2.2 Summary of Behavior Results

Generally, descriptions contained a combination of spatial terms and house and furniture landmarks but very few object landmarks. Moreover, there were key differences as a function of addressee and instruction. When talking to the robot, participants preferred to use fewer words and to adopt a speaker's perspective, whereas when talking to Brian, participants used more words and preferred an addressee perspective. Future work could examine the extent to which these differences as a function of addressee are based on differences in appearance between Brian and the robot or differences in the inferences that speakers make about the capabilities of the addressee. When describing how to find the target, participants consistently used dynamic descriptions that contained more spatial terms and fewer house units and hedges, regardless of addressee. However, when describing where the target was, participants used fewer spatial terms and more house units. They were also more likely to use dynamic descriptions for Brian but static descriptions for the robot and to avoid the use of hedges.

2.3 Templates Derived from Spatial Language Corpus

The work above resulted in a corpus named CSISL (Carlson-Skubic indoor spatial language) corpus containing 1024 spatial language descriptions. This resulting spatial language was analyzed, and templates were created to capture language structure that was common to the spatial language descriptions logged for each test manipulation. In all, across all categories, there were 149 unique templates. There was some repetition across categories because of a lack of a meaningful differences between word counts. Because of this repetition, the templates were examined subsequently simply as a function of how/where and as a function of landmark type (none, goal, path). The other differences that emerged from the older vs. younger and robot vs. human addressee manipulations in the original study and were not expected to be reflected in the path metrics used for comparing robot and human performance, due to the very subtle differences.

2.4 Result Discussion

The research resulted in a corpus of spatial descriptions offered by older adults for finding a target within a virtual house environment in the context of a fetch task. The work uncovered systematic differences in the word choice, selection of particular landmarks such as furniture items, perspectives adopted, and structure, as a function of the addressee and instruction. More generally, the key features of this approach were informed by the corpus including sensitivity to the addressee and the differential assumptions speakers made about its capabilities; differential willingness to accommodate to the addressee; the task context within which the descriptions were offered (specifying how to find an object vs. specifying where the object is); the likely perspective and the likely structure of the

description as a function of addressee and instruction; and the reliance on certain objects in the house (house units and furniture units but not object units) as landmarks.

The goal of this research is to establish a common ground with the elderly user by making the robot adapt to the user's needs as much as possible. However, the results of our study show that seniors may want a more streamlined communication with a task-oriented robot and do not necessarily want to speak to robots in the same way they speak to other people. In conclusion, the work presented here is part of a larger project that has the goal of developing an intelligent system that uses natural spatial descriptions to direct a robot to a target object's location in a fetch task. To accomplish this task, important capabilities were required for cognitive, linguistic, and perceptual processing by the robot.

Chapter 3. The Robotic Research Platform for a Spatial Language Driven Robot: Physical and Simulated Autonomous Mobile Robot System, Perception and Environment Modeling.

3.1 Introduction

In our design, the home robot should have the capability to follow a natural spatial language command to move to a target place. Correspondingly, the world model for the home robot should be able to represent the grounding information given in spatial language. In other literature related to this work [14, 2014 #81], a reference-direction-target (RDT) grounding model was proposed. This model works well as the representation of spatial information contained in commands. In the investigation on human use of spatial language, it has been found that when addressing the position of a target object, the human addresser prefers to use a reference in the description. This requires the robot to not only have the capability of recognizing and labeling the semantic name on objects in the working environment but to extract the spatial relationship between the objects in the working environment so that it can accurately match the world model to spatial descriptions. For the robot, spatial relation extraction relies on a deep understanding of the object's spatial information in the working environment, which calls for a robot perception process providing more details about the target.

The problem of robot perception was initially investigated during the author's M.S. study; the PhD work has extended the work to be more rigorous and robust. The robot perception system in the author's M.S. thesis had an adequate performance on the furniture of the training data. However, it had weaknesses in accuracy and speed. The old system had a big blind range for furniture detection which means for some viewing angles and

distances, the results were very bad and unreliable. Also, the old system was not able to provide real-time recognition for the robot. The robot perception developed in this PhD study is based on the normal features from the depth information collected by the Kinect, and it provides a more accurate and faster furniture detection function to the robot which enabled real-time perception.

3.2 Related Work

The furniture recognition system in the PhD study is inspired by several works on object recognition using the Kinect. The approaches of object recognition using the Kinect can be divided into two groups. One is using depth images and another uses point cloud data, which is converted from the depth image by the Kinect camera. An early stage research conducted by Bo et. al. [19] was motivated by local descriptors on images, in particular, kernel descriptors that developed a set of kernel features on depth images that included model size, 3D shape, and depth edges in a single framework. Their kernel features significantly outperformed traditional 3D features (e.g., spin images). Blum et. al. [20] considered this problem with a bag of features. Their method detects interest points and combines color and depth information into one, concise representation. For the domain of using a point cloud for object recognition, Willow Garage has published a point cloud library (PCL) [21] which presents an advanced and extensive approach to the subject of 3D perceptions. PCL provides support for all the common 3D building blocks that applications need. The library contains state-of-the art algorithms for: filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. Rusu. developed a method using point cloud data to rebuild an indoor environment and generate an object map. PCL provides a powerful tool to compute Normal features [22] and other

point cloud features like point feature histograms (PFH) [23], which made them popular in object detection using the point cloud.

3.3 Kinect-based Autonomous Robot

The robot system is the same one that I used for my M.S. study. It was designed to assist elderly people with household tasks and has a typical framework of an autonomous mobile robot. It is built using a P3DX differential drive mobile robot as the central focus, with sonar array along the side part. Its main sensor is a Kinect camera mounted on top, which is about 1 m from the ground. The Kinect can capture both RGB and depth images simultaneously which can be used to build the environment model during its working time. To command the robot, we first used an android phone with speech recognition to get the spatial description and send it to the robot through WLAN. The robot then grounds the command to obtain understandable navigation instructions. Finally, the robot was navigated to the move-to target. The system has been tested and has proven to be a good platform in experiments [13]. The design of the robot system is shown in Figure 3.1.

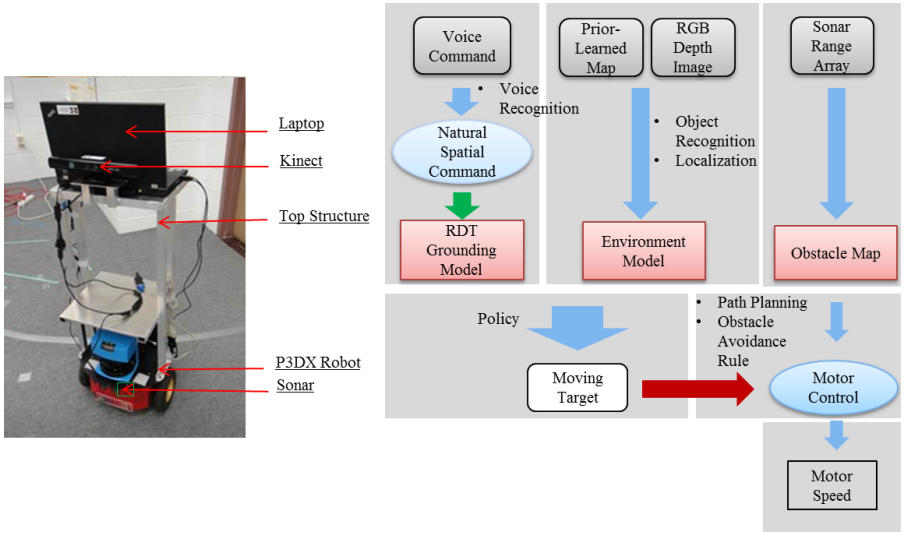


Figure 3.1 (LEFT) Robot design; (RIGHT) A diagram of the system workflow.

3.4 3D Robotic Simulator for the Research of Spatial Language Driven Robot

A side work in this research was to develop a 3D simulator to efficiently evaluate the performance of the spatial language driven robot system designed as the focal point of this dissertation work. The primary contribution of a robot simulator to a physical platform is that it is much more convenient to set up the test scene and record the ground truth. However, most robot simulation systems are unable to absolutely match the real-world scene the robot will work in and it always requires extra work to migrate a target system from the virtual platform to a physical platform. The Gazebo3D robotic simulation platform [24] was chosen as our simulation engine. This simulator has been proven robust and reliable for real-time robotic simulation. The simulator offers wide options on sensors and actuators which makes it easy for users to build up their own robot system. Our robot simulation system includes a robot having all the functions (sensing and acting) of the same physical platform which is introduced in section 3.3 and the several virtual robot working environments used in the experiments of Chapter 5 and Chapter 6. There are four virtual worlds programmed in our robotic simulators. They are the apartment world, the hk-studio world, the one-bedroom-house world and two-bedrooms-house world. The apartment world is identical to the environment we used in the human spatial language experiment in Chapter 2 and the robot experiment in the IROS2014 paper [13], including the building structure, furniture appearance and daily objects in the data collection scenes. The other three worlds use the same models of furniture and daily objectives but are different in the building structure, the placement of furniture items and the positions of daily objects. Figure 3.2 shows the bird's-eye view of the four virtual worlds. The list of furniture items and daily objects are listed in Table 3-1 and their images are shown in Figure 3.3.

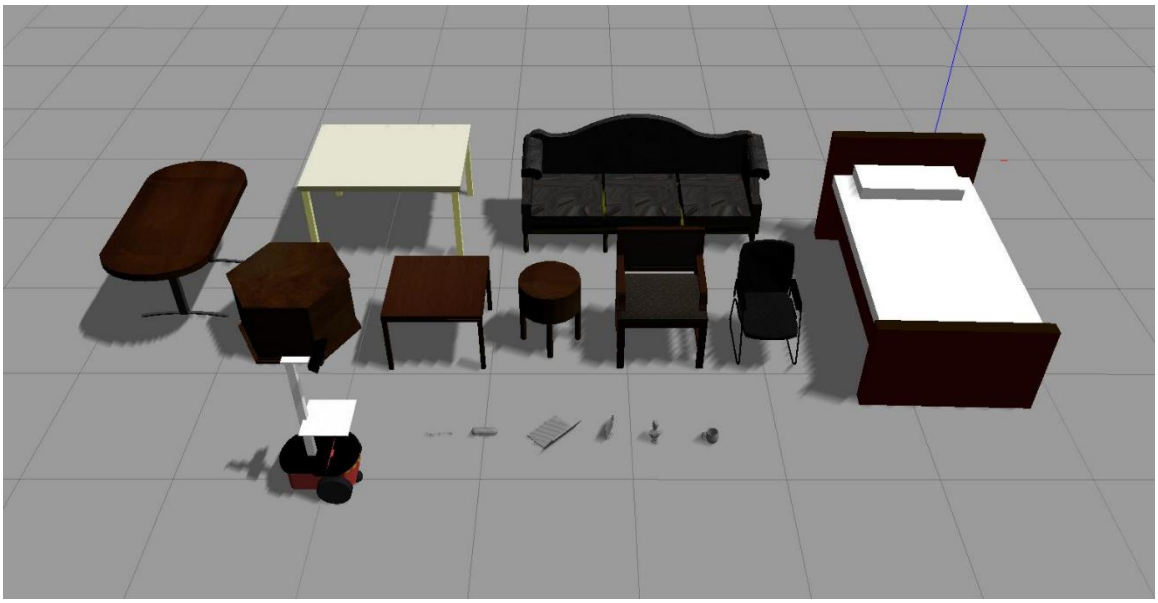
Table 3-1 The types of rooms, furniture and objects in the four virtual worlds.

Room Structures	Furniture Items	Objects
apartment	round table	fork
hk-studio	coffee table	glasses case
one-bedroom-house	hexagon table	laptop
two-bedrooms-house	wood chair	statue
	blue chair	monitor
	dinner table	mug
	desk	
	couch	
	bed	

The ROS-based spatial language grounding system and the natural language generation system will also work from the simulator in the same way as on our physical robot platform. The simulator creates embedded applications for a mobile robot without depending physically on the actual machine, thus saving cost and time for other researchers who are working on or planning to work on the same topic.

Using the words in the template introduced in Chapter 2, we edited another corpus of object fetching tasks in our four virtual worlds. The corpus includes 77 spatial language commands for 24 fetch tasks (six per world). We tested our spatial descriptions system, which follow as well as the language generation on these commands and tasks. The goal was to provide other researchers working on their own spatial language robot a simulator and corpus as a benchmark challenge.

Figure 3.2 The bird's-eye views of the four worlds in the Gazebo3D simulator (top-left: apartment; top-right: hk-studio; bottom-left: one-bedroom-house; bottom-right: two-bedrooms-house)



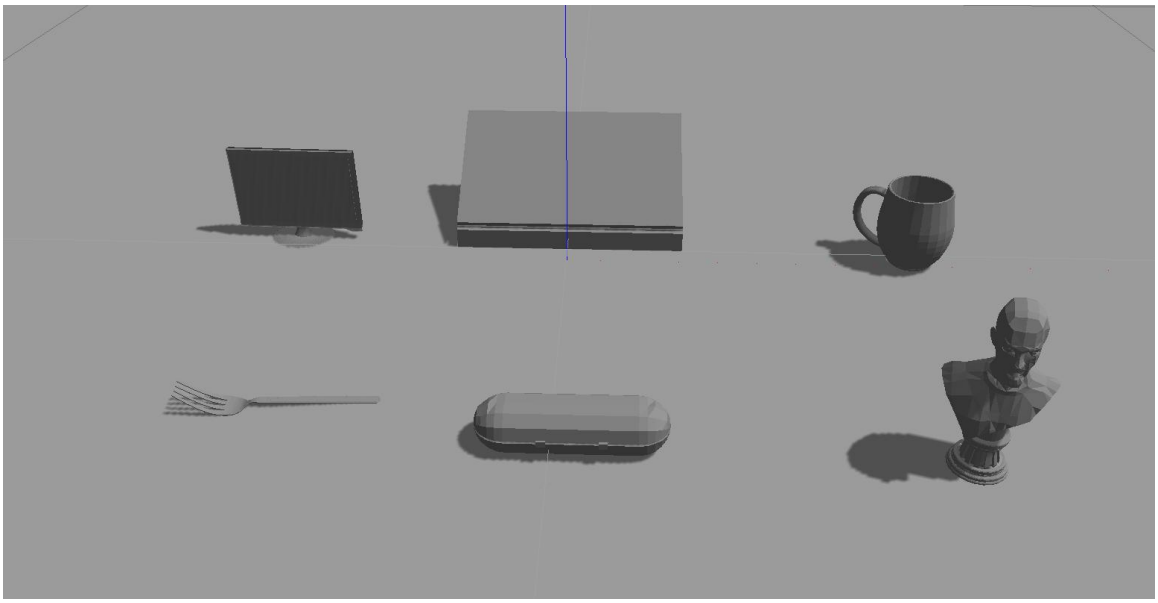


Figure 3.3 The design of the robot, the furniture items, and the daily objects which appear in the four virtual worlds. The upper figure is the joint photo of all the furniture items, the robot and the objects and the bottom figure is a snapshot of all the six target objects.

3.5 Environment Model

Our robot is designed to work in an in-home environment, which includes both private homes and public residences (assisted living apartments, nursing homes, etc.). The robot was designed to conduct assistive tasks such as fetching daily objects. Each fetching task is under the navigation of spatial language given by human users. Those tasks and environments create the following challenges to the robot: (1) The scale of the robot working space is not large, but the space is cluttered with walls, furniture items and daily objects of various sizes and shapes. Those objects are all related to the natural language, which means they should be all registered in the environment. (2) The human spatial commands contain the information of spatial relations between objects, which should be understood by the robot for navigation. Thus, the robot must obtain not only the name label but also the spatial information of each object. (3) The furniture items, which are

considered good landmarks for localization are always moved by users without notice to the robot. In other words, the robot may work in a partially unknown environment. Our solution to these difficulties is to build a more elaborate environment model to include all the information needed.

In the environment model for the robot, the objects are described by an entity model which is discussed in Section 5.4.1. In our model, we use “entity” to represent semantic objects handled in human spatial language. An entity has: (1) an ID, (2) a name, (3) a 2D point set, and (4) an orientation. The ID is the unique identification of an object in a robot task. The ID number of an entity is given by the sequence of detection. The name is a word representing the objects obtained from the spatial language corpus. The 2D point set describe the positions of the cells in a 2D grid map which represent the object’s projection on the floor. To reduce the computation and noise we downsampled the raw point cloud to a voxel grid point cloud. The orientation of an entity is defined as the direction value of its functional front side in its ego-centric reference. For example, a chair has its functional front as the direction that a person faced when last sitting on it. Here, we do not define the orientation by linguistic variable but set it at a more precise numerical angle value in world coordinates. It should be noted that some kinds of furniture items such as night stand or round table are rarely used by human as reference since it is difficult to define a functional front side. These curved items not considered in our orientation estimation algorithm.

3.6 Furniture Recognition

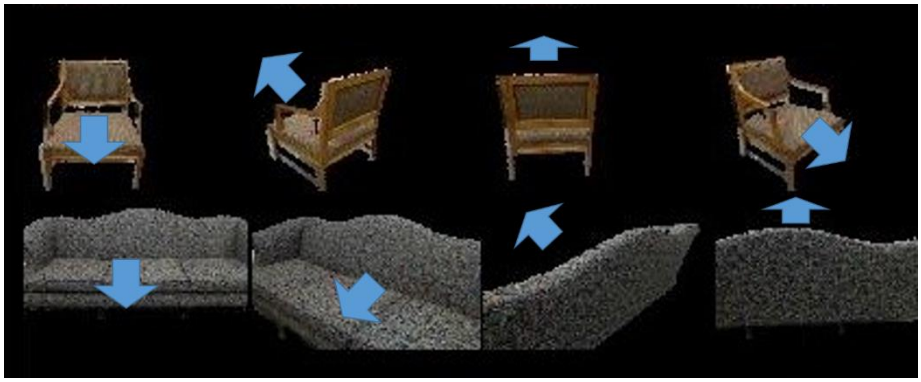


Figure 3.4 The orientation of furniture item (shown by blue arrow).

The furniture recognition system designed in this study has the same goal as in my previous M.S. study. It needs to detect and recognize furniture pieces in an indoor environment, and also needs the capability of detecting each item's position and orientation. As part of the entity world discussed in Section 3.5, a furniture item is represented by a 2D point set of the cells on the grid map, and its orientation is defined as the front side of that furniture item and the direction it faces toward (See Figure 3.4) in world coordinates. In summary, the furniture perception task for the robot includes two aspects: (1) furniture classification and (2) furniture pose detection. Compared with the work in my M.S. thesis [12], this method has not made further development on instance-level classification, but it has focused on category-level classification because the furniture recognition performance is mainly decided by category-level classification; thus, to recognize a furniture item on the instance level is not necessary for a robot to follow a spatial language command.

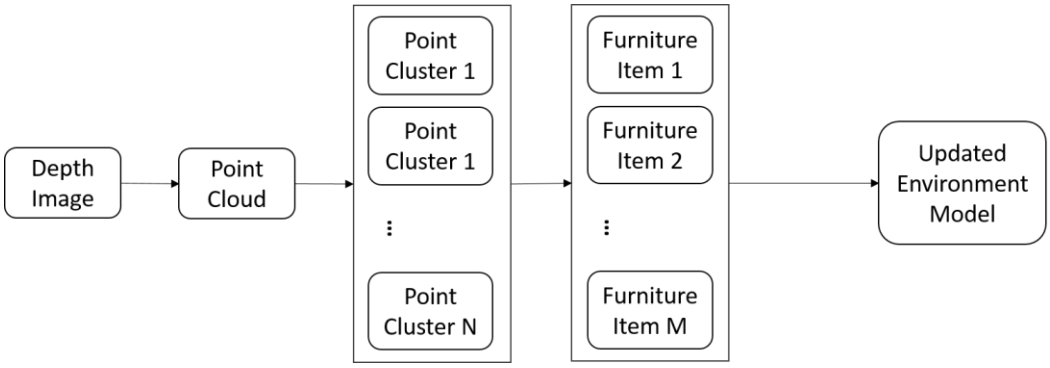


Figure 3.5 The procedure of furniture perception

For a home robot, the prerequisites of the furniture recognition algorithm include: (1) high recognition accuracy and (2) real-time processing. The recognition algorithm in this work is novel and specific for a home environment, and it takes full advantage of the Microsoft Kinect camera, which is used as the main sensor of the robot. The furniture classification is a multi-stage process. Given an input depth image, the first step is to transfer the data from an image to a point cloud and then to preprocess the point cloud data by segmenting the scene and extracting point clusters as potential furniture samples from that scene. Then, furniture classifiers are run on those samples to get their categories. Finally, the pose of each recognized furniture item is estimated and the environment model is built or updated. The workflow of the furniture perception is shown in Figure 3.5.

3.6.1 Extract Furniture Sample

3D point cloud segmentation and furniture sample extraction is preprocessed on the scene by using the Kinect to get furniture samples for recognition. In this stage, we first generate a point cloud with N points which is $P=\{p_1,...p_N\}$ from the depth image by the method described in [25]. After that, the coordinate of each point in the point cloud is resampled from the Kinect camera coordinate frame to the robot coordinate frame.

Floor Detection

Since the camera has a tilt angle to the ground, the floor plane in the camera frame is not a horizontal plane, which requires the robot coordinate origin to be placed on the ground and the x-y plane to be horizontal. The transformation needs the floor plane in the camera frame, which calls for using the random sample consensus (RANSAC [26]) algorithm on the point cloud *P*. To speed up the convergence of the RANSAC plane extraction, only the points that are highly probable and close to the pre-estimated floor are used for processing. A subset point $P'=\{p_f: d(p, f_{\theta,h})<0.1, ||p||<3\}$, where $d(p,f)$ represents the distance of point p to the floor plane f, and $f_{\theta,h}$ is the floor plane parameter roughly estimated by the Kinect tilt angle θ and Kinect camera height h. The formula of the floor plane $f_{\theta,h}$ is $z\cos\theta - x\sin\theta = h$. This restriction can filter the clutter on the floor and only allows the points closest to the robot and the floor to be used to compute the floor plane parameter.

Figure 3.6 Map of 3-D point cloud (left) as it corresponds to a 2-D grid map (right). The white regions of the map have points projected onto the floor and the black regions are labeled as background.

Figure 3.7 An example of an extracted furniture sample.

Extract Furniture Sample

After getting the robot coordinate point cloud, we removed all the points where z-axis values are smaller than 0.1 m so that the points above the floor could be retained. Then the points which were more than 4 m were removed because the measurement precision of Kinect declines with increasing distance and this scale is large enough for perception in a home environment. To segment the point cloud to furniture samples, we first plotted a 2D grid map by projecting all the points to an X-Y coordinate. The range of the coordinates were: $-3.0 \leqslant x \leqslant 3.0$m; $0\text{m} \leqslant y \leqslant 4.0$ m. The 2D grid map has a resolution of 0.1m×0.1m. If a point in the point cloud falls into the range of a grid cell, the cell will be set as occupied. Unoccupied cells are labeled as background (See Figure 3.6). In the following step the connected components in the grid map image were detected by using the method in [27], and were labeled with indices above zero (1, 2, 3…). The background cells which did not contain any points were labeled *0*. The next step was to gather the points that belong to the same component to be a point cloud of sampling. Figure 3.7 shows the result of sampling. Finally, we ignored the samples if the height of the highest point was larger than two meters or the component had fewer than 10 cells because it was part of a piece of wall or a piece of clutter on the ground. We used a voxel grid approach to downsample the point cloud to create a 3D voxel grid over the sample point cloud. Then, in each voxel, all the points presented were approximated (i.e., downsampled) with their centroid. This reduces the points in a sample to increase the speed in classification and improve accuracy. The voxel sample of Figure 3.7 is shown in Figure 3.8.

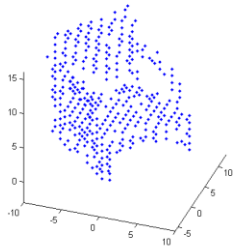


Figure 3.8 The voxel sample of **Figure 3.7**.

3.6.2 *Normal Feature Histogram*

In object recognition, a feature is an individual measurable heuristic property of a sample being observed. Choosing discriminating and independent features is the key to any successful algorithm in classification. For the furniture classifier, the features of a furniture sample are built from the normal vectors of the points in the voxel point cloud. The estimation of the normal value of a point uses its neighboring points. It is a local feature representation that captures the geometry of the underlying sampled surface around the query point.

The normal vectors for each point are coded in the robot coordinate system, and the distance threshold to decide neighbor points is 0.025 m. Using the Point Cloud Library (PCL) [28], the normal vector of each point can be easily generated in a point cloud sample in real time. With the PCL normal estimator, the normal of a point is a 3-D vector which is generated by the methods presented in [23]. Figure 3.9 shows the normal vectors of a chair sample.

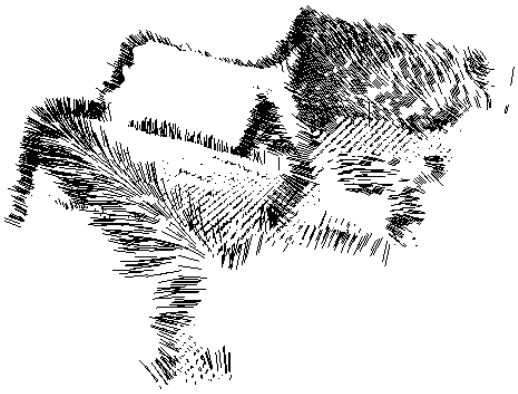


Figure 3.9 The normal vectors of a chair point cloud sample.

A normal feature histogram that describes the entire furniture item as an integration of the normal values of all the voxel units is a 2D histogram map M_h, which is a map of normal vectors by radius (r) and height (z). The radius is defined as the distance from a point to the centroid of the furniture sample in the x-y plane. The height is the z value of the point in the robot coordinate. Considering the largest possible size of furniture, the range of radius distances is set at [0, 2 m] and for the range of height, the range is set at [0, 1 m]. The map grid is defined by "binning" on the ranges of these two variables, which is to divide the entire range of values into small intervals. The intervals of radius and height are both set at 0.05 m in the two 2D grid maps. The normal information is then coded to generate M_h. Because the robot can view a furniture sample from different directions and distances, it is necessary to make the furniture scale features to be invariant in rotation and scale. Thus, a normalized horizontal degree v_p is delivered to each point p from its normal vector $n_p = \{nx_p, ny_p, nz_p\}$ in the feature M_n, where

$$v_p = \frac{\tan^{-1}\left(\frac{nz_p}{\sqrt{nx_p{}^2 + ny_p{}^2}}\right)}{\pi/2} \qquad \textbf{3-1}$$

The range of a v_p value is in [0, 1]. The value of units in M_h which is defined by a pair of radius and height is defined by the means of v_p values of the points adapted to that unit. For those units which have no points, their value is set to be −1. Finally, we have the features of a furniture sample. The orientation for the example of the chair shown in Figure 3.7 is 312 degree.

3.6.3 *Support Vector Machine Classifier*

One of the challenges for perception of the robot system is that a furniture item may look different when viewed by the robot from a different perspective. Therefore, it is better not to train a single classifier for each kind of furniture. Thus, for furniture classification, a furniture class is defined by a furniture category combined with a direction. Table 3-1 shows all the class names. For example, the *couch* is divided into three patterns which are *couch-front*, *couch-side* and *couch-back*. Those subclasses are captured by using K-means clustering on the training samples.

Table 3-2 Furniture categories and subclasses

Category	Chair	Small Table	Dinner Table	Couch	Bed
subclass	chair+front chair+back	small table	dinner table	couch+front couch+back couch+side	bed+side bed+front

Using the feature model mentioned above, both linear kernel-SVM and Gaussian kernel-SVM is used for the classification. For our robot, the outputs of classification are the pattern names in the form of "*furniture+direction."* The two classifiers scored a higher than 90% accuracy on our furniture database. The Gaussian kernel-SVM was finally chosen in the system because it has high accuracy and is fast enough for real-time recognition. The comparison between our current furniture detector and other furniture detection approaches will discussed later.

3.6.4 Deep Neural Network Classifier

The deep neural network (DNN) has become the leading model for all kinds of research fields on machine learning. It is also replacing the other traditional machine learning models, such as Naïve Bayes, SVM and KNN, for the task of object recognition [29]. DNN grabbed the attention of researchers after [30] revealed that it could beat all other methods on the benchmark test. Thousands of papers have been published since then with hundreds of tools and new software kits developed. This milestone research ushered in the age of using a general and universal framework to solve all kinds of machine learning tasks such as classification, regression and optimization.

The problem of furniture recognition can be solved as a classification problem. The DNN-based furniture recognition system will extract the point-cloud based feature samples from the scene by following the same process introduced in Section 3.6.1 and Section 2. Then it will recognize the furniture from a pre-defined set of types. Many different types of DNNs have been implemented for object recognition purposes. The networks can be grouped into three main categories, which are autoencoders (AE) (Vincent 2010, p. 122), the restricted Boltzmann machine (RBM) (Hinton 2010, p. 123), and the convolutional neural network (CNN) [31]. The first two approaches are very general models used to construct a multi-layer neural network and are very popular for many machine learning problems. The third one, the CNN, is more popular in processing 2D image data, but is now becoming the model of choice for mainstream image processing tasks such as human/object detection [32] and character recognition [33]. CNN has performed well on image data, i.e., it does not outperform other data types like our 3-D point samples or the histogram extracted from them. Sometimes CNN, cannot do as well other DNNs [34].

Finally, we built a seven-layer autoencoder based DNN for our furniture recognition system with a trade-off on speed and performance.

The autoencoder DNN furniture recognition system contains five layers which include the input layer of the 200-length feature and the softmax output layer. The other three-layer structure contains the encoder layers where the node numbers are 512, 128, and 32 from the input side to the output side. The activation function for each layer is the *purelin* function (y = x). The network used to train the autoencoders is shown in Figure 3.10, and the structure of the object-recognition network is shown in Figure 3.11. The training of the autoencoder neural network included two steps. The first step used the network shown in Figure 3.10 to train the initial weight values between layers. The weights were updated by minimizing the objective function in equation below.

$$\boldsymbol{J(\theta) = \|\tilde{x}(x, \theta) - x\|_{L1}} \qquad \textbf{3-2}$$

where θ is the weights and bias of the multilayer network shown in Figure 3.10. The value x is the input vector on the left side and $\tilde{x}$ is the output vector on the right side of the network. We want to minimize $J(\theta)$, which is the difference between the two vectors.

In this step, the weights of each layer in the encoder were adjusted so that the end layer of the network (the one with 32 nodes) can represent the input data. This reduced the dimension and improved the performance of the neural network. The second step was to train the network shown in Figure 3.11, which added the softmax output to the encoder. The first step is an unsupervised learning process, while the second step is a supervised learning process. We used cross-entropy as the loss function in the second step training.

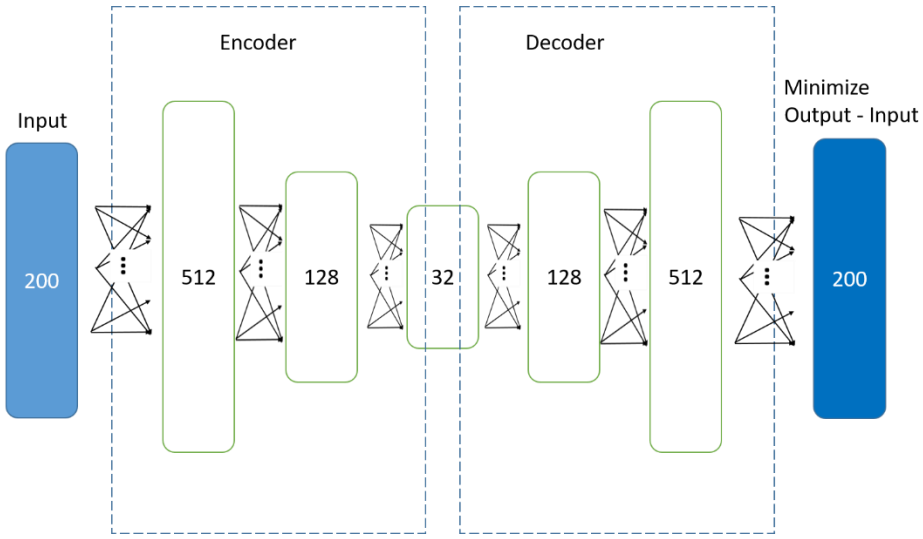


Figure 3.10 The encoder-decoder network used for training the encoder.

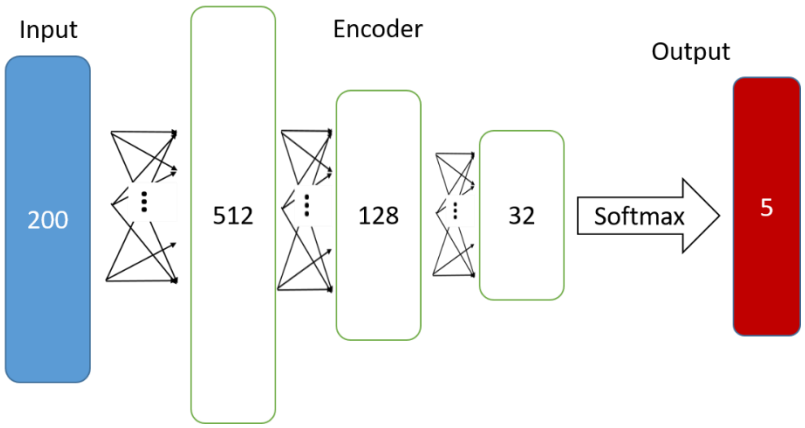


Figure 3.11 The final DNN for object recognition.

3.6.5 Furniture Pose Estimation

Empirically, to simplify the definition of an object position, the object is typically computed as the geometry centroid of its projection on the ground. However, because the size of the furniture is relatively large, the pose relationship between parts of the furniture items may not be the same, which means their relative position cannot be defined by a single direction. Hence, a furniture sample cannot be considered as a mass point. In this system, the position of a sample is represented by its corresponding connected region on

the grid map. The distance between two objects (including the robot) is then defined as the distance between their nearest points.

Human subject experiments have shown that users sometimes reference the intrinsic frame of some furniture items (e.g., in front of the couch) [8, 10, 11]. Even without an intrinsic frame, references such as front and back may depend on the orientation of the furniture item (e.g., rectangular tables) [10]. Thus, it is important for a robot to precisely detect the orientation of a furniture item. Figure 3.12 shows some instances of orientations for some furniture items.

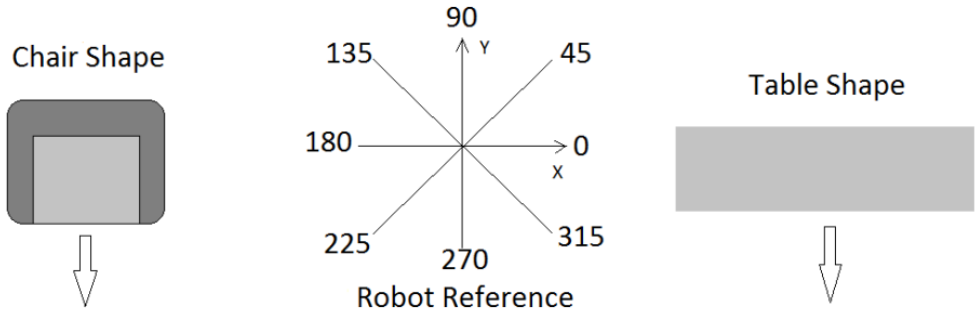


Figure 3.12 The orientations of different kinds of furniture in robot coordinate.

In my M.S. thesis [12], the furniture orientation was considered related to the normal of some functional planes on furniture items. This method was extended by developing a linear model to estimate furniture orientation based on normal directions of the sample point cloud. The goal of this system was to make its output get closer to the furniture orientation. For the basic model of a linear system:

$$\boldsymbol{y} = \boldsymbol{w} \circ (\boldsymbol{x} + \mathbf{1}) \qquad \textbf{3-3}$$

where x is the feature vector and w is the weights. The dimension of w is one more than x by adding a bias input element. The feature vector x is generated from normal vectors of the point cloud sample—similar to the computation of M_n, but here it is changed to the

horizontal component q_p of the normal vector $n_p=\{nx_p,ny_p,nz_p\}$ for any point p in the point cloud. q_p is also the orientation of point p in the robot coordinate frame:

$$q_p = atan2(ny_p, nx_p) \qquad \textbf{3-4}$$

The feature is drawn by the mean of q_p for all the points on radius and height ranges. For each radius and height range, another feature with the value $q_p+\pi$ was added. The training process is the adjustment of the weight vector w on the q of different radii and heights. The policy goal was to decrease the region that is not related to furniture orientation. The training procedure is shown in Table 3-3.

Table 3-3 The procedure of training an orientation estimator.

Assuming x is L=R*H vector, length(w)=L+1=R*H+1. S is the number of training samples
Init: w (0) = {1/L,…,1/L}
for t=1 to S
$w_l(t)=w_l(t-1)$*normalize($x_l(t)$-o(t))
w=w/max(w)
end

The normalized function sets the angle difference in the range of [0,π]. The updating process increases the weight of the grids of which value follows the orientation direction and decreases the other unrelated values. The orientation estimator had a good performance on the furniture items which have an obvious direction. This information enables the determination of the spatial relations between furniture samples.

3.7 Experiment and Results

This section discusses the experiments designed for testing the algorithms introduced in this chapter. Two experiments were run to verify the method performance. The first one was a static experiment of furniture recognition performance in category and instance. The second one was the detection of furniture pose including position and orientation.

3.7.1 Furniture Classification

To reduce the effect of any other factors that may disturb experimental results, a static experiment was run for furniture recognition. The robot was used to take 2-D photos on furniture items from different distances and directions in our experiment environment. The database for the recognition experiment included five categories of furniture items which were all different in size and shape. These furniture items were used to build up the indoor environment for the human-robot interaction experiments. The furniture recognition system had been tested by both our apartment furniture database and the RGB-D scenes database [35].

3.7.1.1 Database

As discussed in this chapter, the furniture category model was built based on a point feature cube. Five categories were defined for the environment model--*small table, large table, chair, couch and bed*. Two databases were used for testing which were the apartment furniture database (built with locally collected samples) and the RGB-D scenes database.

The apartment furniture database used for the recognition experiments included 228 RGB-Depth images taken with the Kinect for eight furniture items. It was also used in my preparatory M.S. work. The number of samples for each instance are shown in Table 3-4.

Table 3-4 The apartment furniture database

Instance Name	Category	Total Number	Training Samples Number	Testing Samples Number
Round Table	Small Table	32	8	32
Blue Chair	Chair	24	8	24
Hexagon Table	Small Table	36	8	36
Wood Chair	Chair	24	8	24
Coffee Table	Small Table	32	8	32
Dinner Table	Large Table	32	8	32
Couch	Couch	24	8	24
Bed	Bed	24	8	24

The RGB-D scenes database is from [35], and includes four categories of furniture samples and 25 instance types, which are shown in Table 3-5. The total number of instances is 226.

Table 3-5 The RGB-D scenes database

Category	Instance Number	Sample Number
Small Table	6	54
Chair	6	54
Couch	6	54
Large Table	7	64

3.7.1.2 Results

Two experiments were run to evaluate our furniture recognition method. The first experiment was run by the apartment furniture database. Eight samples were used for training, and all the samples (including training samples) in that database were used for testing. The results of category recognition are shown in Table 3-6. The second experiment used the whole apartment furniture database as training samples and tested the trained classifiers on the RGB-D scenes database. The results are shown in Table 3-7.

Table 3-6 Furniture classification results on the apartment furniture database.

Category	Sample #	Acc. (LSVM)	Acc. (GSVM)	Deep Network
Small table	100	99%	98%	93.5%
Chair	48	77%	95.8%	93.8%
Large table	32	87.5%	93.5%	93.5%
Couch	24	50%	80%	100%
Bed	24	62.5%	87.5%	90%
All	228	83%	93%	93%

Table 3-7 Furniture classification results on the RGB-D scenes database.(No bed class in RGB-D dataset)

Category	Sample #	Acc. (LSVM)	Acc. (GSVM)	Deep Network
Small table	54	100%	100%	100%
Chair	54	74%	100%	100%

Large table	63	71%	95.2%	100%
Couch	54	0%	90.9%	100%
All	225	61.6%	96.4%	100%

3.7.2 Furniture Orientation Estimation

3.7.2.1 Database

Two databases were used to evaluate the furniture orientation evaluation. One was a specially designed database which is the subset of the apartment furniture database that had 192 RGB-depth images; it was used for the orientation detection experiment. This subset apartment furniture database included eight furniture instances and 24 samples for each of them. These samples can be grouped into five categories. The 24 images included eight directions (0^o–315^o) and three distances (1 m–3.5 m). Furniture orientation test results were shown by measuring the difference of the value to the ground truth. The orientation of the samples in RGB-D scene database were also estimated.

3.7.2.2 Results

The results are shown from Table 3-8 as the absolute difference between the ground truth orientation and the estimated orientation.

Table 3-8 Results of furniture orientation experiment (in degrees).

Category	**Subclass**	**Error mean/std (Apt. Database)**	**Error mean/std (RGB-D Scene Database)**
Small table	-	-	-
Chair	+front	4.5/2.3	2.5/1.7
	+back	7.9/5.6	2.9/2.1
Large table	-	4.7/2.5	-
Couch	+front	7.7/4.5	-
	+side	19.2/14.6	-
	+back	9.5/4.1	-
Bed	+front	4.5/3.1	-
	+side	6.5/4.9	-

3.7.3 Environment Modeling for In-room Scene

3.7.3.1 Dataset

Testing the furniture detectors and furniture pose estimators on the datasets is not enough to evaluate their performance on a real robot task. Thus, an additional experiment of modeling an in-room scene was designed to validate the robot perception system in the spatial language driven task. To make it convenient to compare the perception result with the ground truth, the system was tested in the 3D simulator introduced in Section 3.4. Sixteen scenes (four from each world) were extracted and the robot explored each to detect the furniture items in the scene. In this test, a scene is a furniture cluster with two or three pieces of furniture. The robot tried to detect the type, position and orientation of each furniture item in the scene and the results were compared against the ground truth.

3.7.3.2 Results

Table 3-9 describes the furniture detection results, including the error metrics on position, the orientation to the ground truth where the furniture items were placed in the scenes, and the observation on whether the spatial relationships detected matched the ground truth. Fourteen out of the sixteen scenes were detected with the correct spatial relationships.

Table 3-9 Results of environment modeling as shown by PE (position error), OE (orientation error), and SR (spatial relation) between the detected furniture items matching the ground truth. The third column lists the ground truth of the positions (x,y coordinate by meter) and the orientations (by rad) of the furniture items in the scenes. The fourth column lists the detected positions and orientations. The fifth column is the error between the ground truth and the detection. The last column lists the result that if the detected spatial relations could match to the ground truth.

#	Room	Scene (Ground Truth) x(m), y(m), orientation(rad)	Scene (Detected) x(m), y(m), orientation(rad)	Errors distance(m) /angle(rad)	If SR Match
1	Apartment	couch: 4.5, 3.0, 1.57 table: 4.0, 2.0, -1 table: 4.5, 4.0, -1	couch: 3.95, 2.71, 1.45 table: 3.83, 2.01, -1 table: 4.31, 3.92, -1	0.62/0.12 0.17/- 0.2/-	Y
2		table: 4.5, 7.0, -1 chair: 5.75, 7.0, 3.14	table: 4.89, 6.77, -1 chair: 5.59, 6.87, 2.60	0.45/- 0.20/0.54	Y
3		table: -1.25, -5.0, -1 chair: -1.25, -6.0, 0 chair: -1.25, -4.0, 0	table: -1.06, -5.00, -1 chair: -1.07, -5.98, 0.10 chair: -1.08, -4.04, 0.24	0.19/- 0.18/0.10 0.17/0.24	Y
4		bed: 5.00, -3.00, 4.71 table: 4.0, -2.0, -1	bed: 5.00, -3.00, 4.71 table: 3.82, -2.05, -1	0/- 0.18/-	Y
5	HK Studio	chair: -4.0, -1.7, 4.71 table: -4.0, -3.0, -1	chair: -3.87, -1.84, 4.23 table: -3.85, -2.78, -1	0.19/0.48 0.26/-	Y
6		couch: 0, -3.50, 0 table: 1.0 -3.5, -1	couch: 0.15, -3.50, 0.22 table: 1.12, -3.37, -1	0.15/0.22 0.17/-	Y
7		chair: 4.0, -3.3, 0 table: 4.0, -4.3, -1	chair: 3.84, -3.16, 5.35 table: 3.85, -4.13, -1	0.21/0.93 0.22/-	N
8		chair: 4.5, 3.0, 3.14 table: 4.5, 4.0, -1	chair: 4.37, 3.10, 3.10 table: 4.31, 4.04, -1	0.16/0.04 0.19/-	Y
9	One Bedroom House	table: 3.5, 3.0, -1 couch: 4.5, 2.5, 3.14 chair: 4.0, 4.0, 1.57	table: 3.39, 2.84, -1 couch: 4.32, 2.01, 3.02 chair: 3.89, 3.84, 1.73	0.19/- 0.52/0.12 0.19/0.16	Y
10		chair: 0.5, 0.5, 4.71 table: -0.6, 0.9, -1	chair: 0.66, 0.59, 4.81 table: -0.41, 0.98, -1	0.18/0.10 0.20/-	Y
11		chair: 1.5, 8.0, 1.57 table: 0.5, 8.0, -1	chair: 1.49, 7.82, 1.57 table: 0.49, 7.8, -1	0.18/0.0 0.20/-	Y

12		chair: 4.5, 6.25, 0 table: 4.5, 5.5, -1	chair: 4.31, 6.22, 0.50 table: 4.32, 5.48, -1	0.19/0.50 0.18/-	N
13	Two Bedrooms House	couch: 0, 0.5, 1.57 table: -1.4, 0.5, -1 table: -1.25, 2.0, -1	couch: -0.14, 0.66, 1.48 table: -1.38, 0.68, -1 table: -1.16, 2.13, -1	0.21/0.08 0.18/- 0.15/-	Y
14		table: 1.0, 6.0, -1 chair: 1.0, 7.25, 3.14	table: 0.82, 5.94, -1 chair: 0.82, 7.18, 2.72	0.18/- 0.19/0.42	Y
15		chair: 3.5, 7.5, 3.14 table: 2.5, 7.5, -1	chair: 3.49, 7.28, 3.02 table: 2.76, 7.32, -1	0.22/0.12 0.31/-	Y
16		couch: 1.25, 2.0, 3.14 table: 0.5, 2.5, -1	couch: 1.16, 2.12, 2.93 table: 0.42, 2.68, -1	0.15/0.21 0.19/-	Y

3.8 Summary

The first experiment showed the results of recognition using the Kinect camera. The following three conclusions were made:

1) The furniture detector built by the *normal feature + SVM* classifier provided a reliable approach to detect the static furniture objects in the indoor environment. The Gaussian kernel SVM performed better than the linear kernel version even though it took longer to compute the result.
2) The result of the furniture detector using the deep neural network was not better than the conventional SVM-based detector for small furniture items. However, it gave a more robust performance on the larger furniture items. The reason is that the DNN-based detector can automatically capture more information on features from the furniture data than the hand-edited features.
3) Another conclusion, which is not illustrated in the results table, is that for the chair-shaped furniture items, it was easier to make accurate decisions when they were facing the Kinect camera, which means the favored orientation interval was between 180^{o} and 360^{o}. Although this represents some improvement on detection recognition compared to my preparatory M.S. work, accurate navigation is still a

challenge when the furniture sample (especially a chair shape) has its back to the camera.

The results obtained in the second experiment show the factors that affect performance of orientation detection. The following two conclusions were obtained:

1) The PCL and normal features contribute to improving the orientation estimation accuracy for all kinds of furniture items.
2) It is still a challenge to estimate the orientation of large-size furniture items, i.e., dinner table or bed. It is better for a robot to learn about these furniture samples before navigation as long-term entities.

The third experiment demonstrates the robot's notable accuracy in modeling the environment, as the distance and direction errors were very small in most cases. Since we infer the spatial information by the entire 3D point cloud model rather than the centroid point of the furniture samples, this amount of error did not affect the result of the spatial relationship inference.

Chapter 4. Natural Spatial Language Grounding

4.1 Introduction

A home robot is now expected to go beyond a conversation agent to assist human users on some physical household tasks such as fetching and passing some objects. These tasks need built-in reliable communication and understanding between machine and human on spatial language so that the human can navigate the robot to the right places, using natural intuitive language. We call the procedure of interpreting natural spatial language for robot action as natural spatial language grounding. This chapter is about a system which includes the architecture and algorithm on natural spatial language grounding. The system obtains the natural spatial language commands or descriptions as input, and it segments the sentences to meaningful clauses. Here each clause was interpreted to an action or a spatial relation to describe the next path point which is called "grounding." Then the groundings are places in order. The robot was driven by the behavior models of the groundings until it reaches the destination. Human language and human manipulations were collected to build the models of language grounding and robot behavior.

This chapter include another two sections as background as overview. The second section is a review of my colleague T. Alexenko's work on the part-of-speech tagging. The third section is a conference paper published in IROS 2014, which describes the whole system of spatial language grounding. The fourth section talks about using a long short-term memory (LSTM) network to build the mapping from a natural language clause to a grounding.

4.2 Part-of-Speech (POS) Processing on Human Spatial Language

4.2.1 Introduction

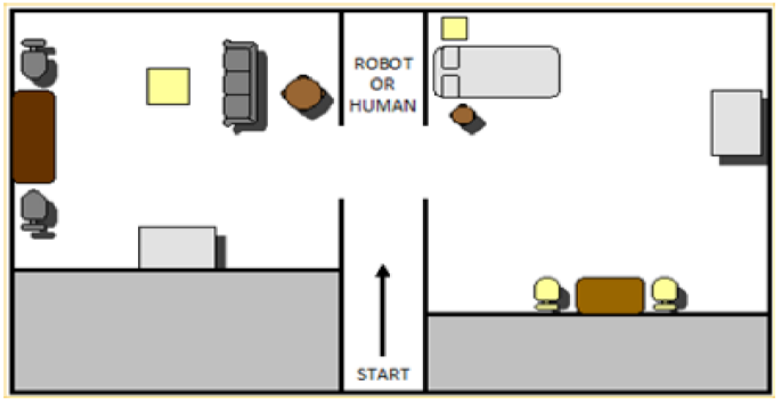


Figure 4.1 The overhead map of the environment

This work was mainly finished by my colleague Tatiana Alexenko from 2012 to 2013. It builds on top of the human robot interaction (HRI) architectures proposed by Skubic and Carlson. The purpose of this section is to discuss the basis for the semantic grammar and the necessary natural language processing (NLP) methods in more detail. Preliminary results for automatic chunking and part-of-speech tagging are presented. The results of chunking became the input of the spatial language grounding system.

4.2.2 Semantic Spatial Language Grammar

The spatial language grammar was developed based on the Carlson-Skubic indoor spatial language (CSISL) corpus containing 1024 spatial language descriptions collected from a human subject experiment first mentioned in Chapter 2.3. In addition to the primary corpus of 1024 actual utterances (CSISL), Carlson et. al. and Skubic [10] also created a template corpus consisting of 149 descriptions based on templates derived from the analysis of the primary corpus. The templates were created manually from words most frequently used by the experimental subgroups. Skubic et. al. (2012) [36] observed that

participants in these experiments very often used furniture in the descriptions they gave. Aside from some changes and additions on nouns, i.e. FUR for furniture and RM for room, the Penn Treebank [37] PoS tag set was used. The use of these semantic tags was meant to simplify aspects of chunking and further steps such as perception.

4.2.2.1 Semantic Grammar and Nested Chunks

Table 4-1 List of semantic chunk labels and their abbreviations.

Outside Room Target Phrase	ORMTP
Outside Room Reference Phrase	ORMRP
Object Target Phrase	ORMRP
Object Reference Phrase	OBRP
Furniture Target Phrase	FURTP
Furniture Reference Phrase	FURRP
Inside Room Reference Phrase	IRMRP
Perspective Indication	PERS
Confusion Indication	CONF

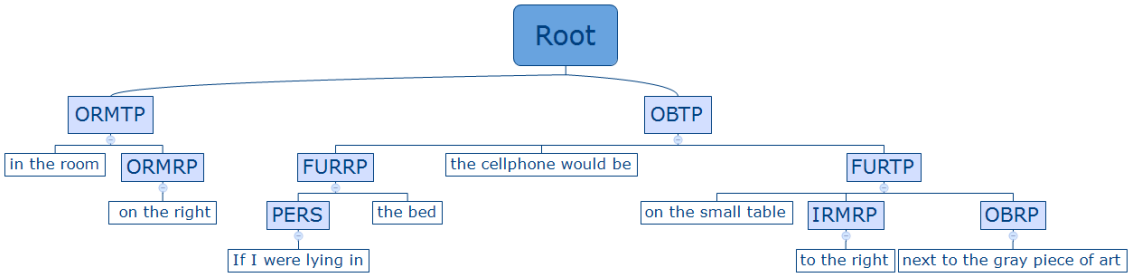


Figure 4.2 The proposed semantic spatial language grammar applied to a real description (CSISL corpus) from human subject experiment conducted by Carlson and Skubic (2011).

Table 4-1 lists the semantic chunk types and their abbreviations and Figure 4.2 shows the proposed grammar applied to a real description. The semantic grammar proposed here is specific to the "fetch" task described in [11, 36], and in the CSISL corpus. The grammar has rules for deciding which labels can be parents to other labels. Target phrases are always parents of reference phrases. For example, generally OBTPs will be parents to other phrases (in some cases the rest of the description, including ORMTP) because this is the "goal" of the description. If the description started with "the cellphone is in the room on the right," as many static "where" descriptions did, OBTP would be the parent of every other node.

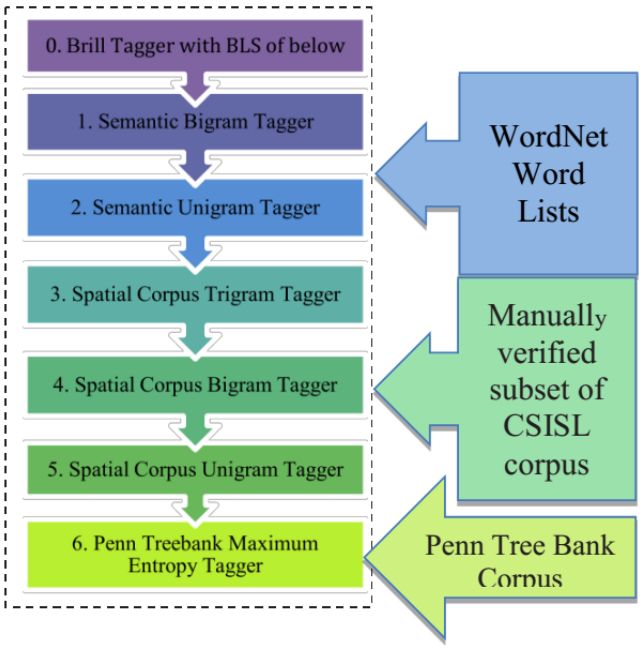


Figure 4.3 Architecture of the back-off semantic PoS tagger and the source of training data. The Brill tagger uses all the other taggers as a Base Line System (BLS) and, therefore, does not have its own training data.

The 1024 real descriptions in the CSISL corpus as well as the corpus of 149 template descriptions were annotated with PoS tags (including the special semantic tags). A hybrid approach of manual and automated annotation was used. NTLK [38] and standard Python

libraries were used for this task. Any punctuation, if present, was stripped because there is no punctuation when speaking, although it could be added back in with automatic comma and period placement methods.

Corpus Annotation

4.2.2.2 Part-of-Speech Tag Annotation

A specialized back-off tagger was created as shown in Figure 4.3. A Brill Tagger [39] was trained using the back-off taggers (1-6 in Figure 4.3) as a base line system. The resulting PoS tagger was then applied to the rest of the spatial language corpus to provide PoS tag annotation.

4.2.2.3 Semantic Grammar Annotation

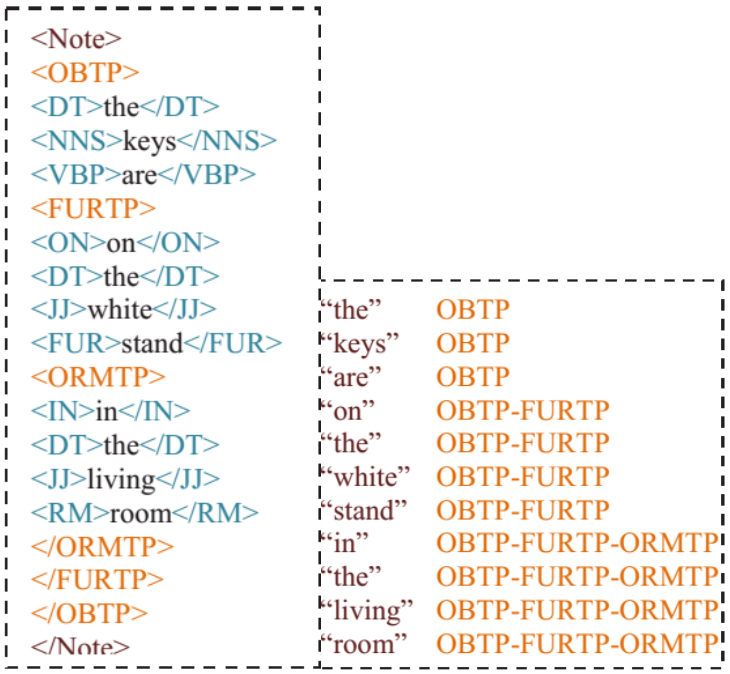


Figure 4.4 A portion of the annotated corpus (left) was stored in XML format, and a nested chunk encoding (right) was applied to this example.

For the semantic grammar annotation, a manual approach was required; however, some automatic techniques were used. To simplify this task and provide a widely-supported

format, the PoS-tagged lists (descriptions) were then parsed with a coarse regular expression parser [40]. Figure 4.4 (left) shows what the end result looked like in an XML format after annotation. The annotators were presented with similar input with the exception of incorrect and missing chunk labels (from RegEx Parser) which they needed to fix. The XML tags around PoS labels were provided to the annotators to speed up the process.

4.2.3 Nested Chunk Encoding

The semantic grammar proposed in this section is nested unlike the traditional "chunks;" the nesting needs to be preserved since it captures useful spatial relationships. This made the commonly used inside-outside-begin (IOB) encoding [40] unusable. The development of nested chunk encoding can serialize the nested chunk labels into a single label and preserve the structure. The method is very simple and the serial chunk labels are shown in Figure 4.4 (right).

4.2.4 Results Discussion

The PoS tagging results were in the upper ninety percentile, which is comparable with the state of the art and sufficient for the task. The text chunking results on the template corpus were very high, nearly 100% in some trials and no less than 93%, which is very high considering the nesting, which greatly increases the number of possible labels, and the small corpus size (149 template descriptions). The reason for the relatively low accuracy of the chunking on the real descriptions (CSISL) is a combination of factors. First there are lingering errors and a possible (not measured, but noticeable) lack of cross-annotator agreement. The accuracy results of the subset annotated by Alexenko were about 7% higher. Since the annotation was done over the course of two months, the annotators

became better at the task over time. The Brill-tagger may also not be the best method for the task, although it was chosen for its speed and availability. More complex models combined with voting schemes are likely to outperform Brill at this task just as they do at other NLP-related classification tasks [41]. Another problem was that the Brill tagger only operates on doubles, which means that only (PoS tag, chunk tag) or (word, chunk tag) could be used for training, not the entire triple, leading to a loss of potentially useful features. A more detailed analysis of the source of errors (likely rare labels such as “PERS” and “CONF”) needs to be conducted.

4.3 IROS2014 Paper: Using Spatial Language to Drive a Robot for an Indoor Environment Fetch Task

Note: *This section represents a word-for-word reprint of an article that appeared in the proceeding of IROS 2014.*

Abstract—This paper proposes a system that allows the use of natural spatial language to control a robot performing a fetch task in an indoor environment. The system processes spatial referencing language and extracts a tree structure of language chunks. The spatial language system is then grounded to a robot navigation instruction in the form of a sequence of actions based on spatial references to furniture and room structure; the best navigation instruction is selected by scoring. In addition, the Reference-Direction-Target (RDT) model is proposed to represent indoor robot actions. To control the robot for the fetch task, a behavior model is designed based on the RDT model. An assistive robot has been designed and programmed based on this system. The proposed spatial language grounding model and robot behavior model are tested experimentally in three sets of experiments. Results show that the system enables a robot to follow spatial language commands in a physical indoor environment even if the referenced furniture items are re-positioned.

I. INTRODUCTION

The aging population is becoming a challenge that will continue to stress the care of seniors in the future. The old-age dependency ratio in the United States was 0.20 in 2012 and will increase to 0.35 at 2050 [1][2]. In other countries, the situation is more severe. For example, the old-age dependency ratio in Japan was 0.39 in 2012 and is forecast to be 0.74 in 2050, which means four Japanese workers per three retired older people (not considering

children) [1][2]. A shortage of labor leads to shortages of healthcare staff. This creates a need for assistive devices such as robots [3]. Surveys have shown that older adults would consider assistive robots for household tasks such as fetching and searching for missed objects [4]. Furthermore, older adults also prefer natural language rather than other communication methods for robot interaction. In this paper, we propose natural spatial language interface methods for communicating with a robot performing the fetch task. Here, we focus on the language translation and navigation of the fetch task. The grasping component is not included [13].

Robot spatial language understanding has been explored previously. Matuszek [5] proposes an idea to convert natural language commands to logic descriptions. Tellex et al. developed a probabilistic graphical model, named generalized grounding graphics, to derive the best grounding solution from natural language commands. It is realized on a forklift robot as a sequence of robot actions [6][7]. Kollar et al. developed an imitation learning policy to convert natural spatial language commands to sequential actions in an unknown environment. The method is tested on a simulation platform [8]. Fasola et al developed a model to generate a global path from using dynamic spatial relation references in a semantic map [9]. They assume the robot has a global knowledge of the working environment. Our work differs from the previous work in that different language structures are supported and we do not assume complete knowledge of the scene. Also, in this paper, we report test results with a real (non-simulated) robot in a physical environment.

Details of our proposed system are included. The next section discusses spatial language grounding, i.e., how to ground natural language chunks to a robot navigation instruction. The Reference-Direction-Target (RDT) model is proposed; a scoring

procedure is used to find the best robot navigation instruction from the chunked natural language description. In addition, we introduce robot behavior models that support both dynamic and static spatial descriptions. Dynamic spatial descriptions use sequential actions such as "*go forward*" and "*turn left*" to navigate a robot to a target location. Static spatial descriptions use objects as references to describe a target location, i.e., "*behind the couch*" or "*on the table next to the bed*". The third section shows the design of an assistive robot and its perceptual capabilities. The fourth section presents the experiment and results in a one-bedroom and one-living room apartment environment. The experiment was run in the physical world, which means the performance is affected by both the spatial language grounding model, as well as the robot's perception and navigation capabilities. Finally, we conclude with discussion and future work.

II. SPATIAL LANGUAGE GROUNDING

When using natural language for a spatial oriented task, people prefer to use relative spatial references rather than precise quantitative terms. For instance, to describe the position of a cellphone, people may say "the cellphone is in the living room on the right on the table behind the couch" rather than "the cellphone is 3.21 meter from the living room door at a 45 degree direction". However, it is not as easy for the robot to understand such human-like descriptions. We refer to the procedure to translate a natural language description into a robot-understandable navigation instruction as grounding. In this paper, the natural language robot fetch command is first grounded (i.e., translated) to a robot navigation instruction and then executed by a pre-defined robot behavior model. To ground natural language to robot navigation instructions, we first use the method discussed in [11] to extract a tree structure in the form of language chunks. Then we use a scoring procedure

to find the best robot command match for each chunk and connect them together to form a robot action sequence.

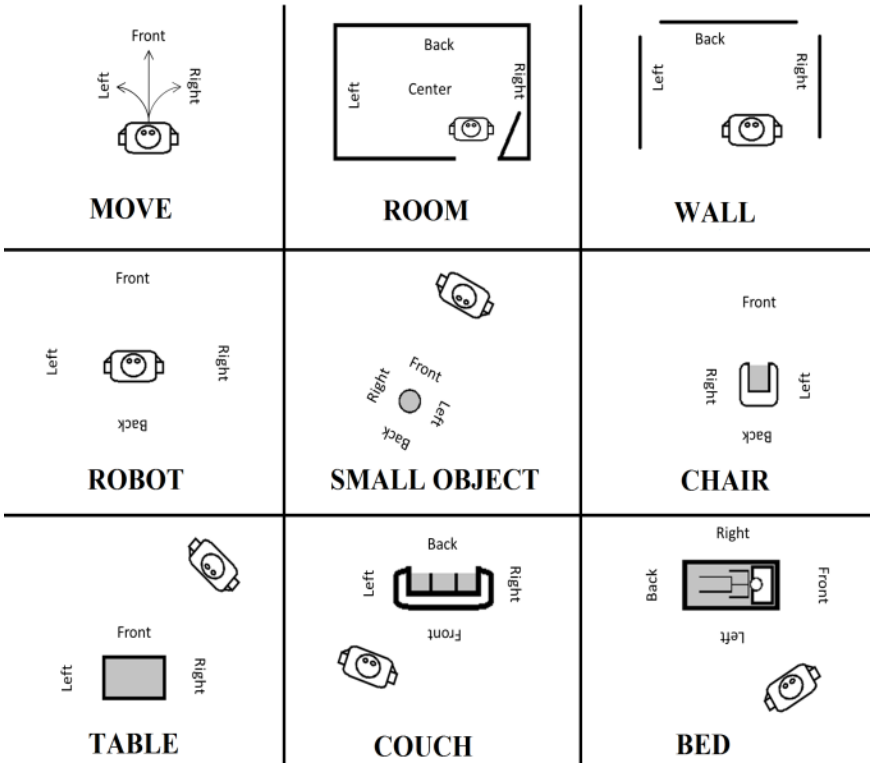


Fig. 1 Reference types and their corresponding directions, defined from human subject experiment [13].

A. Fetch Task Model

In our fetch task, the robot is assumed to have prior knowledge of the room structure as this is fixed. However, we assume that the placement of furniture and daily objects inside the room is not known to the robot. In the fetch task, a human speaker stands in a hallway outside the target rooms and gives the robot a spatial description of the target object. The robot addressee then starts from the hallway, moves to the designated room, and then moves to the target object. The target objects are assumed to be on the surface of furniture items so there is no need to search inside furniture. The robot uses its local perception for navigation and object recognition in this task. The fetch task process is divided into three

sub-tasks which represent three types of groundings which is a bridge between the natural spatial language description and the robot action:

(1) Target Room: Determine the target room and enter the correct room,

(2) Inside-room Navigation instruction: Move close to the target object by following the spatial description.

(3) Target Object: Find the target object designated by the speaker.

B. Reference-Direction-Target (RDT) Model

The most difficult part of the task is to navigate the robot within the target room because the robot has no a priori information of furniture placement within the room, which may be changed by people who live there. The robot will use its own perception for navigation and object recognition. Guided by the human spatial language description, a robot can find a target object more efficiently than aimless searching. The Reference-Direction-Target (RDT) model is proposed which converts the inside-room spatial description into a series of actions with navigation instructions (grounding type 2, above).

In the RDT model, ***Reference*** refers to objects in the room, furniture or even room structure, e.g. wall and door. It can also be a label that informs the robot about the behavior type it should perform. Dynamic commands are defined as a special kind of reference type which has no real reference object but rather uses a sequence of moves, e.g., turn left, go forward. Such reference types are different from static command behaviors that need perception to find an object used for reference in navigation. Several types of references are used in the fetch task, as described below. These references are collected based on human subject experiment [13].

MOVE – This reference represents dynamic spatial language commands in which there is not a real reference object. For example, “turn right” or “go forward”. There may be a target object for this reference type.

ROOM – This reference uses fixed room information in navigation, e.g., “move halfway in” or “to the left part of the room”. The Direction component shows the possible part or direction of a room as destination. Because the room structure is not changed, by using a compass, odometry and prior knowledge of the room structure, the robot can move to the target area. In the experiment, the robot has a semantic room map with walls, and doors for the navigation.

WALL – A wall is used as the reference to define target position, e.g., “to the back wall”. The robot will start searching once a target is in the RDT node.

ROBOT – When using the robot itself as the reference, it does not directly appear in the description, but rather ego-centric references are used, e.g., “behind you”.

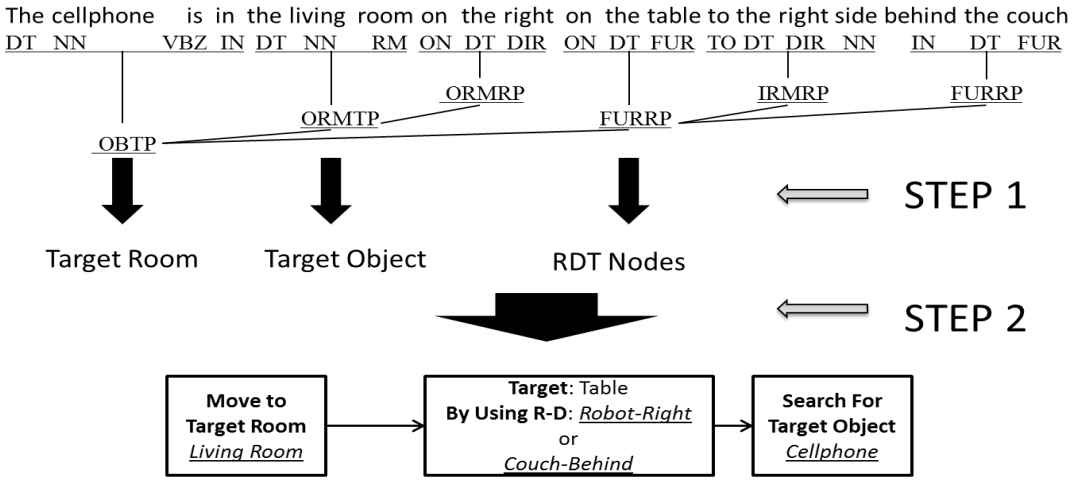
FURNITURE – A furniture item is used as a reference object. Based on previous work [10][11][12][13], the direction of the furniture reference follows human interpretation. For example, “in front of the chair” is defined using the intrinsic frame of the chair while the direction of the table and couch is defined by the viewing angle.

Direction represents the position relationship between objects. It tells the robot where it should move to search for the target. The direction of each reference type is defined based on human speakers’ intentions. It may not be based on object ego-centric coordinates. For MOVE and ROBOT, the direction uses robot ego-centric coordinates. For FURNITURE, the direction sometimes uses viewer angle. For different types of navigation instructions, the reference frame for direction may be defined very differently. The directions used in

the robot fetch commands include: front, left, right, back, central, side, and between. The Histogram of Forces (HoF) is used to represent the direction reference [19][20]. Fig. 1 shows the different reference types and their corresponding directions.

Fig. 2 Spatial Language chunking tree example

Target indicates the target furniture or target object in the navigation instruction.



Sometimes there is not a target furniture word in the spatial language chunk. Often, the target furniture can be derived from content or human intention, usually, a table. There is a natural assumption that people usually put small objects on table-like furniture. An RDT node is built based on a target. It has one target and one or more reference–direction pairs because a speaker may use more than one reference to describe a target position. The FURTP chunk in Fig. 2 shows a multiple reference-pair example. If more than one reference-direction pair is given, the robot will skip remaining pairs once the target is found.

C. Grounding from Chunking Tree to RDT model

The RDT model can support either dynamic or static spatial descriptions with the same framework. The input is a chunking tree extracted by part-of-speech tagging [11]; see Fig.

2. The tree is parsed by a forward direction traversing process through the tree. The result is an action queue.

To convert the chunking tree to a grounded navigation instruction (a sequence of robot actions), a scoring method is designed to find the maximum likelihood match for each chunk. It scores a chunk by two steps. First, it recognizes the grounding type. The grounding types include the target room, the inside-room navigation instruction, and the target object which represent the fetch sub-tasks. Then for the inside-room navigation case, the second step finds the reference, direction and target information (RDT node) of the chunk. Fig. 2 shows the procedure of a grounding example "***The cellphone is in the living room on the right on the table to the right side behind the couch***".

The scoring model is trained using spatial descriptions from a template corpus which summarizes the structure of 1024 collected spatial language descriptions for a robot fetch task [14]. There are 101 unique chunks which cover all the words for six target object fetch description sets. First, we manually label the grounding information of each element in the training chunks. Chunk elements include chunk tag, chunk text, parent chunk tag and children chunk tags.

Extracting the grounding type can be viewed as a classification problem. The final result is the grounding type with the highest score. For a sample of chunk *s*, the scoring equation for grounding type classification is:

$$T=Maximum_{tp}(P(t_p|tag_s)S(tx_s,TX_{tp}))$$

T is the result of the grounding type classification. $P(t_p|tag)S(tx_s,TX_{Tp})$ is the score of the t_p grounding type. tags is the chunk name of the sample chunk s, and the definition of each kind of chunk name can be found in [26]; tx_s is the text of the sample chunk *s*; TX_{tp} is a

corpus of template chunk text with chunk nametags and belongs to grounding type t_p. $S(tx_s,TX_{Tp})$ is the degree of membership for tx_s to TX_{Tp} by weighted-Levenshtein-distance (WLD).

$$S(tx_s,TX_{tp}))=1–Minimum(WLD(tx_s,tx_{tp}))$$

The result of the first step is shown in Fig. 2.

The following scoring equation is used in step 2.

$$G=Maximum_{gd}\ (P(g_d|tag_s)S(\ tx_s\ ,TX_{tp})P(g_d|prt\_Tag_s)P(g_d|phn\_Tag_s))$$

G is grounding result. This equation can be used on all of the groundings in the RDT node, including reference, direction and target furniture/object. g_d is all the possible groundings in a grounding type T which is derived from step 1. $P(g_d|Tag_s)S(tx_s,TX_{tp})$ $P(g_d|prt_tag_s)P(g_d|chn_tag_s)$is the score value of the grounding type. prt_tag_s is the parent tag of the sample chunk and chn_tag_s is the child tag of the sample chunk. Fig.2 shows the second step result for the example command.

A RDT node may be built using more than one chunk. In the example in Fig. 2, the FURTP chunk and its two nested child chunks build a single RDT node. The FURTP chunk is grounded to “target: table”. The IRMRP is grounded to “reference: robot + direction: right”. The FURRP is grounded to “reference: couch + direction: back”. The groundings of IRMRP chunk and FURRP chunk both describe the position of the target table in FURTP.

D. Robot Behavior Model

The robot behavior model is built using the result of the spatial language grounding. The basic behavior of the robot is to compute the best point that fulfills the navigation instruction requirement and then let the robot move to it. The higher tier is a global sequence of three subtasks. The lower tier is for the navigation within the room by RDT

nodes. Dynamic descriptions and static ones are distinguished by using different state machine strategies. Because dynamic descriptions use no furniture as references or targets, they do not need furniture searching and detection behaviors. By using odometry with prior knowledge about the house structure and basic obstacle avoidance by range sensing (e.g., sonar), it is possible for the robot to move to the target location. However, the static command strategy requires the robot to search and recognize the reference and target items and because of the limitation on perception, the robot sometimes should move to an intermediate position to get a better view to improve its perception confidence. The system will try reference-direction pairs sequentially until the target is detected when there is more than one reference for a target. This is an improvement than previous work because it reduced the ambiguous in target searching and it then brought higher success rate in experiment results.

III. ROBOT DESIGN

A. Robot Design

A mobile robot with the intelligence to navigate in an indoor environment and interact with a human has been designed and built to validate the performance of the method discussed in this paper. The robot has a differential drive chassis with an RGB-Depth camera.

A Pioneer 3-DX (P3DX) robot was used as the robot chassis and driving component [22]. The robot has a 16 unit sonar array, eight in front and eight in the back. The tower frame is made of light aluminum and holds a Kinect camera and a laptop computer. The Kinect is popular because it can provide high quality synchronized color and depth data [23]. Usually its effective detection range is from 0.5 meter to 8 meters which is adequate

for indoor work. The controller of the robot is a laptop which runs the perception, robot behavior and human-robot interaction programs. The robot uses the Robot Operating System (ROS) [24][25], as a software platform. ROS provides libraries and tools to help software developers create robot applications. For robot navigation, we manually constructed map of room structure for robot and allowed it to use odometry for localization. However, the robot has no information about furniture items inside the room and it has to use visual perception to explore the furniture map.

B. Visual Perception

The spatial language corpus collected for the fetch task uses furniture items as reference objects [11][14]. Thus, the robot perception for the fetch task consists of two parts which are furniture recognition and furniture pose detection. Our previous work discusses details on the furniture recognition and furniture orientation methods [11].

1) Furniture Recognition

Furniture recognition provides category classification of a furniture sample, whereas furniture pose detection identifies the position and the orientation of a furniture piece. We used features of shape, size, height and color for the furniture recognition challenge, using both RGB and depth images. The furniture recognition results are good but not perfect, which lends a realistic perspective to the experiments.

2) Furniture Pose Detection

Furniture pose detection includes both position and orientation. We use a grid map to represent the furniture positions and the robot because it retains the size and shape information which is used for the HoF-based spatial relations computation. The orientation is defined according to the different furniture categories. The orientation of table- shaped

furniture is defined by the orientation of the short visible edge [11]. The orientation of chair-shaped furniture is defined by the orientation of the chair back [11].

IV. EXPERIMENT AND EVALUATION

1. Spatial Language Grounding Experiment

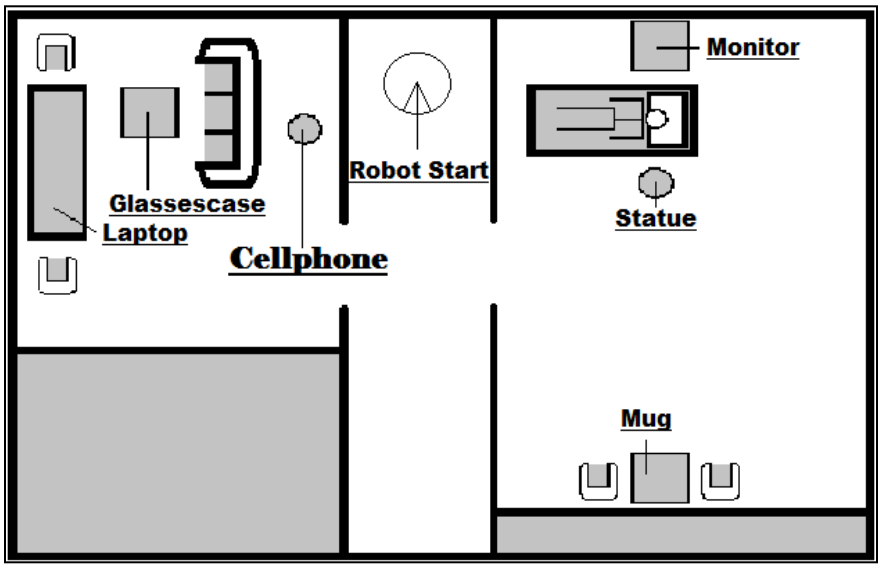


Fig.3 Experimental environment with furniture and object placement

Three experiments are used to evaluate the proposed system. The first experiment evaluates the translation (grounding) of the spatial language chunking tree to robot navigation instructions. We used the template corpus with 149 spatial descriptions, which summarizes the structure of 1024 collected spatial language fetch descriptions [14].Even our ultimately goal is to let the robot can interact with human by natural talking, all the commands are input in text form in this experiment so that the evaluation can be independent from speech recognition. The descriptions were categorized by major syntactic differences across instruction type (how/where) and a function of landmark type (none, goal, path). The *how* commands were mainly dynamic (sequential actions), whereas the *where* commands contained more static descriptions [14]. For different landmark conditions, *none* means no furniture reference was used. A *goal* landmark included a spatial

reference description of a table where the target was located. A *path* landmark means there was furniture used as reference in the description along the path to the target [14]. The object names and positions are shown in Fig. 3. The ground truth for this experiment was manually edited. We used 36 commands for training (six for each of 6 target objects) which were representative of both dynamic and static descriptions. All 149 descriptions were used for testing. The result is shown in TABLE I.

TABLE I Spatial language grounding experiment results (in %)

Types and Landmarks	**How vs. Where**		**Goal vs. Path vs. None**			**Total**
	How	**Where**	**Goal**	**Path**	**None**	
Successful Rate	89.4	81.0	89.5	72.54	100.0	87.9

~~2.~~1. *Robot Behavior Test*

The robot behavior model was evaluated in a two-room environment which has the same structure used for collecting the spatial descriptions. The room map and furniture and object placement are shown in Fig. 3. After the spatial language grounding procedure, there are 33 unique robot instruction combinations generated. To evaluate the robot behavior separately from the spatial language translation, this experiment used manually generated navigation instructions. In a fetch task, the robot is required to start from the hallway, enter the target room, then move along a path to the target furniture and take a picture of the target object. The robot state in each frame for each trial is recorded. An RGB image is taken with the robot's Kinect at the end of each trial. The criterion of success is that the target object is recognizable on the camera picture at the end of the trial. We ran both simulation and real robot experiments with improved robot behavior model. The results are displayed by landmark type in TABLE II with a comparison to the previous simulation experiment [14].

TABLE II Robot behavior experiment results (in %)

Experiment Type	Goal	Path	None	Total
Simulation (Previous)	89.5	40.0	98.0	84.6
Simulation	89.5	86.0	98.0	90.1
Real Robot	50.0	78.6	100.0	81.3

~~3.~~2. *Robot Behavior Model Robustness*

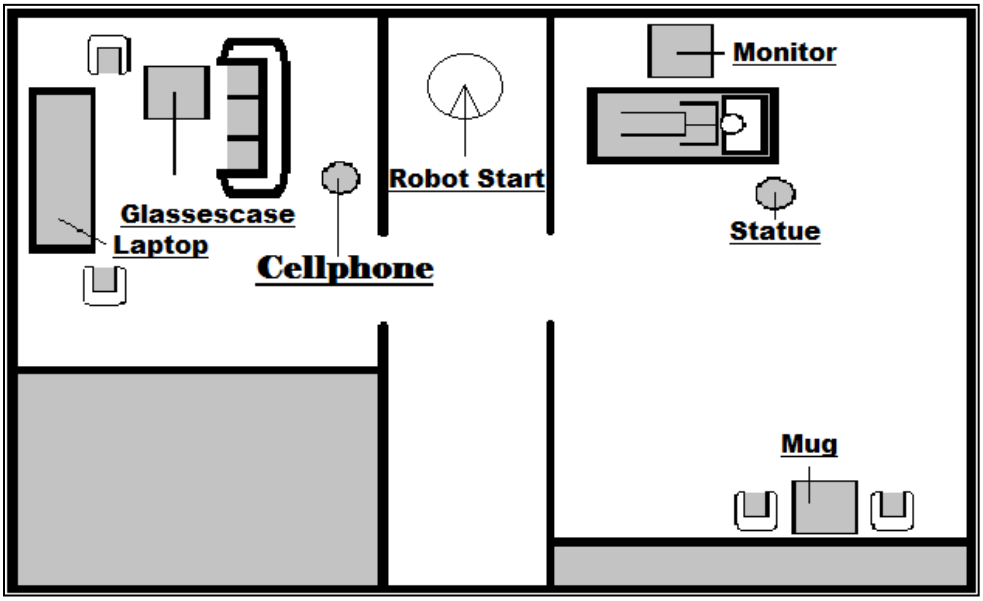


Fig. 4 Modified room placement for Experiment 3.

The robot behavior model was further evaluated for robustness by changing the furniture placement in the scene. In real life, the furniture position may be changed a little bit without notice by people. Such a change usually does not affect spatial relations between furniture objects. Therefore, a robot should have the ability to keep an accurate spatial understanding with a slight furniture position change. A modified furniture placement of the rooms, as shown in Fig.4, was used to test the robot behavior model again using the same navigation instructions. The results of the two experiments are compared in TABLE III.

TABLE III Robot behavior model robustness experiment (in %)

Experiment Type	Goal	Path	None	Total

Original Placement	50.0	78.6	100.0	81.3
Modified Placement	33.0	78.6	100.0	78.1

Fig. 5 shows pictures of the scene and robot view for the fetch description: "***Go into the room on the left. Move about halfway in and then turn right. Go forward to the table against the wall with the chairs and there is the mug.***" Fig.6 shows the robot path of the fetch task. The path consists of a set of purple short lines. Each line represents a robot position in the path. The slope of each short line is the robot orientation.

Fig. 5 The left image is a scene photo taken by an external camera. The right image show a view from robot Kinect camera.

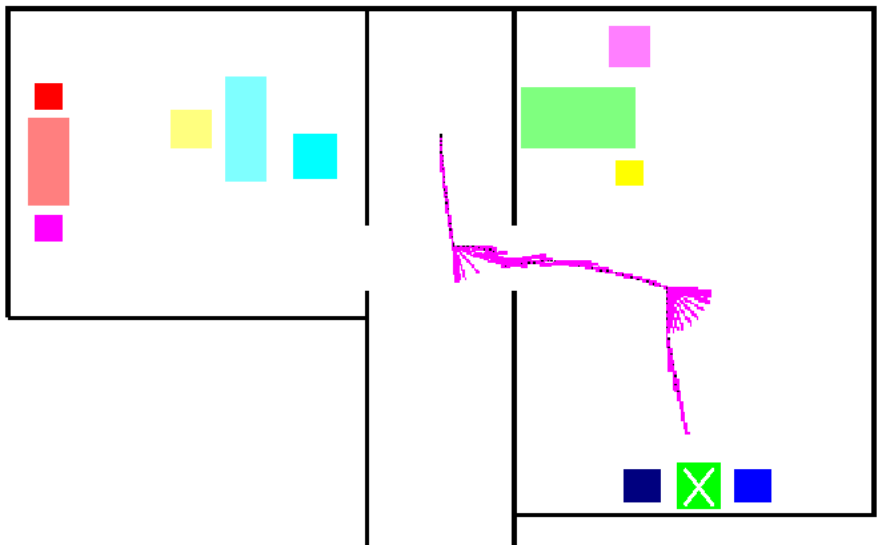


Fig. 6 The robot path for the fetch task, the X is the position of the target furniture.

V. CONCLUSION

The work in this paper enables the robot to follow spatial language descriptions for a fetch task. The methods proposed here can be expanded to a standard spatial language understanding model. In the paper, we discuss a method to ground chunked spatial descriptions to robot navigation instructions. We defined a Reference-Direction-Target model which supports both dynamic and static spatial descriptions for indoor navigation instructions. With the RDT model and the HoF, which models spatial relations, the robot behavior can be built dynamically. A robot system was built to evaluate the system introduced in this paper. The spatial language grounding experiment shows good results for both dynamic and static descriptions, and includes improvement over the work in [14]. The robot behavior model experiment evaluated the basic method in a real world environment. However, the perception of the real world robot yielded a lower performance overall compared to the simulation experiment. In our result, the "*Path*" and "*None*" landmark type result is better than simulation due to an improved grounding algorithm in RDT node building. An analysis of the robot trace shows that the robot sometimes incorrectly detects furniture which results in the wrong reference and direction selection. This decreased the performance in the "*Goal*" landmark case. The robot behavior model robustness experiment demonstrated the robot is robust on small furniture position changes that retain basic spatial relationships between furniture items.

We will continue to improve the spatial language grounding system and the corresponding robot behavior model. Future plans include an experiment on a larger corpus collected from older adults rather than the templates. The grounding system will be improved for the end-user. Moreover, we will also improve the perception by building

more precise furniture models to solve occlusion problems. We will continue simulated and real robot experiments to evaluate the robustness of the system in new room structures and with varying object placement. Our ultimate goal is to build reliable robot to assist elderly people in the home environment.

Acknowledgement

The authors gratefully acknowledge Laura Carlson, Jared Miller and Xiaoou Li for their contributions to our early work and human spatial description data collection.

REFERENCES

[1]. *Old-age dependency ratios*. The Economist. May 9th, 2009

[2]. Age dependency ratio, old (% of working-age population). World Bank Data.

[3]. J. Beer, M. Jenay, L. C. Tiffany, P. Akanksha, L. M. Tracy, C. K. Charles, & A. R. Wendy, "The domesticated robot: design guidelines for assisting older adults to age in place", *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pp. 335-342, IEEE, 2012.

[4]. M. Scopelliti, V. Giuliani, and F. Fornara, "Robots in a Domestic Setting: A Psychological Approach," *Universal Access in the Information Society, 4,* pp. 146-155, 2005.

[5]. C. Matuszek, E. Herbst, L. Zettlemoyer, & D. Fox, "Learning to parse natural language commands to a robot control system", *In Proc. of the 13th Int'l Symposium on Experimental Robotics (ISER),* 2012

[6]. S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller & N. Roy, "Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation," *Proc., Conf. on Artificial Intelligence (AAAI),* 2011.

[7]. S. Tellex, P. Thaker, J. Joseph, and N. Roy, "Learning perceptually grounded word meanings from unaligned parallel data," *Machine Learning,* pp. 1–17, 2013.

[8]. F. Duvallet, T. Kollar, A. Stentz, (2013, May), "Imitation learning for natural language

direction following through unknown environments", *In Robotics and Automation (ICRA),* pp. 1047-1053. 2013 IEEE.

[9]. J. Fasola, & J. Mataric, "Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots", *In Intelligent Robots and Systems (IROS),* pp 143-150, 2013 IEEE/RSJ.

[10]. M. Skubic, Z. Huo, L. Carlson, X. Li, J. Miller, "Human-Driven Spatial Language for Human-Robot Interaction", *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[11]. M. Skubic, T. Alexenko, Z. Huo, L. Carlson, J. Miller, "Investigating Spatial Language for Robot Fetch Commands", *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.

[12]. M. Skubic, L. Carlson; X. Li, J. Miller, Z. Huo, "Spatial language experiments for a robot fetch task", *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on. IEEE,* 2012.

[13]. L. A. Carlson, M. Skubic, J. Miller, Z. Huo, and T. Alexenko. "Strategies for human-driven robot comprehension of spatial descriptions by older adults in a robot fetch task", *In Proc*.

[14]. M. Skubic, Z. Huo, T. Alexenko, L. Carlson, and J. Miller, "Testing an assistive fetch robot with spatial language from older and younger adults." *In RO-MAN, 2013 IEEE*, pp. 697-702. IEEE, 2013.

[15]. G. A. Radvansky, S. A. Krawietz, and K. T. Andrea, "Walking through doorways causes forgetting: Further explorations", *The Quarterly Journal of Experimental Psychology 64, no. 8 (2011),* pp. 1632-1645, 2011.

[16]. G. A. Radvansky and D. E. Copeland, "Walking through doorways causes forgetting: Situation models and experienced space", *Memory & cognition 34.5 (2006),* pp. 1150-1156, 2006.

[17]. C. R. Arkin, "Behavior-based robotics", *MIT press*, 1998.

[18]. L. A. Carlson, and P. L. Hill, "Formulating spatial descriptions across various

dialogue contexts", *Spatial Language and Dialogue 1.9 (2009)*, pp. 89-10, 2009.

[19]. M. Skubic, D. Perzanowski, S. Blisard, A. Schultz, W. Adams, Magda Bugajska, and D. Brock, "Spatial language for human-robot dialogs", *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 34.2 (2004)*, pp. 154-167, 2004.

[20]. P. Matsakis, and L. Wendling, "A new way to represent the relative position between areal objects", *Pattern Analysis and Machine Intelligence, IEEE Transactions on 21.7 (1999)*, pp. 634-643, 1999.

[21]. R. Siegwart, and I. R. Nourbakhsh. "Introduction to autonomous mobile robots", *MIT press*, 2004.

[22]. P3DX Robot Introduction, Internet: http://www.mobilerobots.com-/researchrobots/pioneerp3dx.aspx

[23]. Leyvand, Tommer, et al. "Kinect identity: Technology and experience."Computer 44.4 (2011): 94-96.

[24]. Quigley, Morgan, et al. "ROS: an open-source Robot Operating System." ICRA workshop on open source software. Vol. 3. No. 3.2. 2009.

[25]. ROS Wiki, Internet: http://wiki.ros.org

[26]. Z. Huo. "Robot methods for human-robot spatial language interaction", Master Thesis, University of Missouri--Columbia, 2013.

4.4 Long Short-term Memory-based Spatial Language Grounding Model

4.4.1 Introduction of Recurrent Neural Network (RNN)

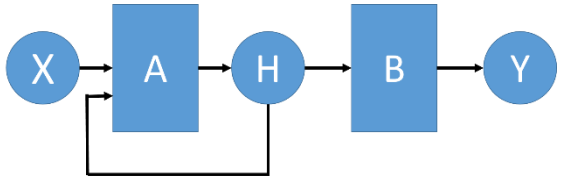


Figure 4.5 Conventional recurrent neural network

Since deep learning is such a huge component of artificial intelligence research, it is reasonable to try it on the spatial language grounding system. Even though the recurrent neural network (RNN) does not have many layers in its structure, it is still considered a type of deep neural network for its complicated connection form and its unfolding to a multi-layer full network when it is being trained [42]. The structure of a conventional RNN is shown in Figure 4.5. This figure shows that the input of the network not only comes from new incoming data but also comes from a branch at the last output. This indicates that the output of an RNN is determined by both input and the last output, and the last output is calculated from the last input and the output before the previous output. Thus, we can conclude that the RNN will predict an output by both input and previous output which has been proved by several research works [43].

However, the conventional RNN system has a problem of long-term dependencies. The problem is that even though the output can connect to the previous information of the present task, the learning of the connection from the previous information to the present output will become unstable and unreliable when the gap increases. Here the gap is the distance between the previous information and the present. For example, when using the RNN to predict the word in a sentence "*the car is on the* ***road***", the only information to predict the word ***road*** is the previous word *car*. Here the gap is small since we only predict the word ***road*** using the information in the same sentence. However, for the case "*it is raining outside, and I will take an* ***umbrella***", we can only predict the ***umbrella*** when we trace back to the word *raining*; thus, there is a much larger gap between the two words. The conventional RNN may work for the first case when the previous information is closely related but would have difficulty training a long-term connection in the second case.

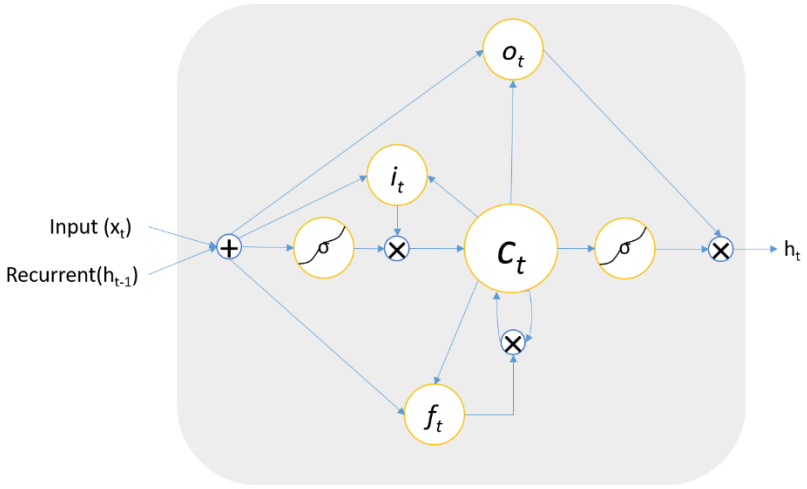


Figure 4.6 An LSTM cell for the LSTM network

Long short-term memory (LSTM) networks, introduced by Hochreiter[44] and Gers (2000, p. 131) [45] are a type of the recurrent network that can learn "long-term" dependencies. LSTM is explicitly designed to avoid the long-term dependency problem caused by the differences in structure. The complexity of the paths and gates from the inputs and outputs from the input and last output to the present output is much more noticeable in LSTM. The core idea behind LSTM is to control information through using the elements. Four main elements make up an LSTM node, also referred to as a memory cell, which includes three gates: 1) input gate, 2) output gate and 3) forget gate, followed by 4) a neuron with a self-recurrent connection. The structure of an LSTM cell is shown in Figure 4.6. The input gate allows or blocks the incoming data (new and recurrent) to alter the parameter of the cell. The output gate allows or prevents the state of the memory cell to influence the next neurons in sequence. Finally, the forget gate controls the self-recurrent connection, allowing the cell to remember or forget its previous state as needed.

The equations to compute the elements in Figure 4.6 are:

$$f_t = \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right)$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i)$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o)$$

$$\boldsymbol{c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c)} \qquad \textbf{4-1}$$

Here x_t is the new input vector, h_t is the output of the hidden layer, and c_t is the cell state vector. The variables *W*, *U* and *b* are weight matrices and bias vector. The activation function σ_g is a sigmoid function, and σ_c is a hyperbolic tangent function.

4.4.2 LSTM-based Spatial Language Grounding Model

Our LSTM-based spatial language grounding model will use the words *vector of the leaves* phrase in the chunking tree as input and output of the grounding information. The model is designed to replace the old grounding model discussed in Section 4.2.II.C, which determines the RDT node by matching the nearest phrase in the dictionary.

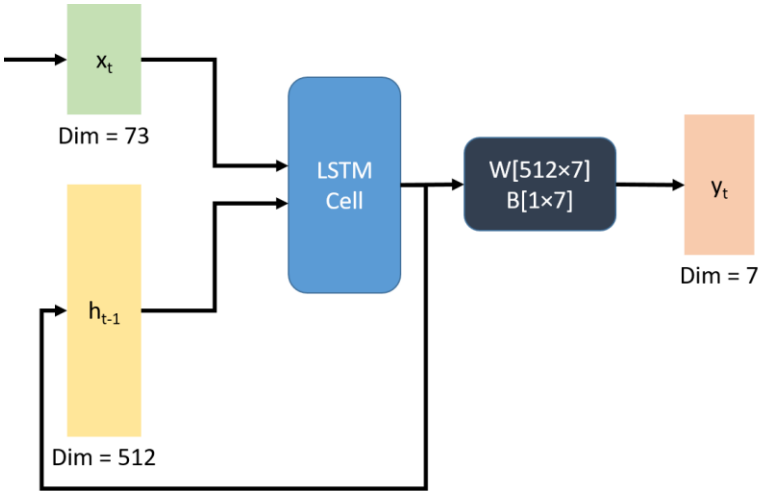


Figure 4.7 The LSTM-base spatial language grounding model

The structure of the LSTM grounding model is shown in Figure 4.7. The input is a tagged utterance chunk presented as a binary vector. The vector is concatenated by two one-hot feature vectors which are chunk type and chunk word. The chunk type is one of the types shown in Table 4-2 ([15]), and the word is one from the dictionary extracted from

the CSISL template [8]. The output is the grounding class decided by the softmax function. We grouped the groundings into five types which are *target room*, *target object*, *reference*, *direction*, and *target*. Five LSTM networks were trained for the five grounding types, and for each grounding type, we had $N_{type}+1$ output at the softmax output layer, where N is the number of the grounding classes for the ground *type*. The last output represents outlier. To build a universal model for grounding, we set the same structure for the LSTM network of each grounding type and used different data and strategy to train the networks. The input vector along with the late output hidden layer vector become the final input of the LSTM node. Here, we set the nodes of the hidden layer to be 512. The hidden layer with 512 nodes was then connected to the final output layer by a weight matrix plus a bias vector. The linear activation function was used for this connection. The cross-entropy cost function and gradient descent strategy was used to update the weights in the network.

Table 4-2 List of semantic chunk labels and their abbreviations.

Outside Room Target Phrase	ORMTP
Outside Room Reference Phrase	ORMRP
Object Target Phrase	OBTP
Object Reference Phrase	OBRP
Furniture Target Phrase	FURTP
Furniture Reference Phrase	FURRP
Inside Room Reference Phrase	IRMRP
Perspective Indication	PERS
Confusion Indication	CONF

We used Tensorflow [46] to train the LSTM network. An "unfolded" version of the network, which contained a fixed number (*num_steps* = 10) of LSTM inputs and outputs was created to make the training tractable. The model was then trained by fill the inputs of the length *num_steps* at a time, and a backward pass was performed after each such input

block. The weights in the LSTM network were updated by the average of the update variants in each node with the *softmax_cross_entropy_with_logits* function in Tensorflow.

4.4.3 Experiment and Result

We extracted 818 chunks, each of which had a chunk type and a phrase from the CSISL template for training and testing. All the chunks were vectorized to a 73-length (7 chunk types + 66 words) binary feature vector. The batch size was set to four so that the network would densely update during training. The iteration number was set to 10000.Two-thirds of the 818 samples were randomly selected for training and the remaining one-third were used for testing. We bootstraped the training for 10 steps and show the average accuracy results in Table 4-3.

The results of the LSTM-based grounding model include the accuracies of the prediction on each kind of grounding. The accuracy was computed by $\frac{number\ of\ corrected\ prediction}{total\ number\ of\ test\ samples} \times 100\%$. The item *All* represents the accuracy of labeling a short spatial language clause on all the grounding types. The comparison between the previous approach and the LSTM network is shown in Table 4-4.

Table 4-3 Results of the LSTM-base language grounding model for each type of grounding

Grounding Type	Accuracy (%) by LSTM network
Target room	98.5
Target Object	97.1
Reference	97.1
Direction	96.0
Target	100
The whole spatial command	89.7

Table 4-4 The comparison on the accuracy of the whole command between the previous approach and the LSTM network.

Accuracy (%) by Previous Approach	Accuracy (%) by LSTM network
87.9	89.7

4.4.4 Conclusions

The LSTM-base spatial language grounding model provided an alternative option to the WLD-based dictionary-query system for language grounding. Their performances were very close. Since we did not develop the system in large-scale data, which is a foundation to efficiently implement DNN, it was not easy to arbitrarily determine which model was going to give the best performance. However, the LSTM model provided a more flexible and generative approach to build the spatial language grounding model when new corpus, spatial concepts or entities were imported into the system. Even the old spatial language model predicts the results based on the word frequency of the human spatial language corpus; thus, we could not avoid manually editing some of the rules to refine the model. LSTM provided a solution to make the model converge to a state with better performance without human correction.

However, the LSTM network had the essence of a black-box model which made it unpredictable in some cases. The WLD-based model can always give a good prediction for the spatial language clauses stored in the dictionary, while the LSTM may predict the wrong result even for the learned cases.

Chapter 5. Building Robot Behavior Policy for Spatial Language Commands by using Programming by Demonstration (PbD)

5.1 Introduction

5.1.1 Motivation

The work in this chapter is a further exploration on directing robots using spatial language. It was developed on the basis of previous work [13-15, 36] using a robot project designed for robot fetch tasks. In the previous work, a framework was introduced, which extracted a grounding model from a natural language command [13, 14]. The grounding model was defined by a reference-direction-target (RDT) model which came from actual human spatial language experiments conducted by Carlson and Skubic [11]. This represents an action sequence of the robot task. After getting the action sequence, the robot then executes actions which are determined by a robot behavior policy until it reaches the target object. In the robot system fetch task, the robot behavior policy model maps the RDT model to the robot moving action.

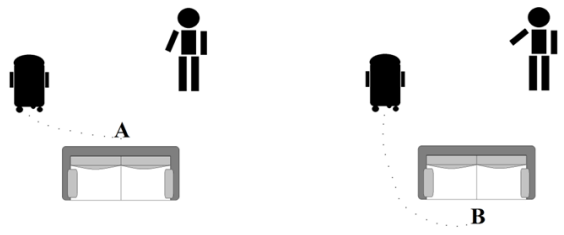


Figure 5.1 Ambiguity in understading spatial language: When the human user directs the robot to "*go to the front of the couch,*" the move-to targets can be considered as place A or place B, which are both possible to a human addressee.

Programming the policy model of a robot is a challenge. In a robot system, policy defines the rule of the robot action with a given input. In our robot system, the input

included a spatial command from the human user and sensory information. In the previous robot system, a policy was fixed for a certain kind of command. Policies were manually edited by the robot developer, which represents a significant amount of programming work. Moreover, during the human subject experiments, we found that human addressers can have different explanations for one spatial command. For example, in Figure 5.1 the user's choice is position *A*, which is the target place of the clause "*in front of the couch*," but another user will choose position *B* based on the same command and position. This causes an ambiguity problem in policy programming and robot developers have not been able to build a rule to determine the move-to target before interaction with the robot user. The robot programmer must prepare different policy models to adapt to various user needs. Overall, building robot behavior policies not only requires professional skills based on robotics science but also requires programming knowledge to enable the robot to perform the given task in a proper working environment with an understanding of the user and how he or she will interpret, respond and communicate with a social robot. The development of a robotic system by hand is challenging and time-consuming. As a result, machine learning was applied to policy programming. In this chapter, we illuminate a framework to build a robot behavior policy model by robot learning rather than by hand, which was the strategy used in our previous work. Our system uses a particular approach to robot learning which is programming by demonstration (PbD) [47]. Within the PbD, a policy is learned from demonstrations provided by a teacher. We defined examples as sequences of input-output pairs that are recorded during the demonstration of the desired robot behavior. The input includes information from two sources: (1) from sensory information which assesses the spatial relations between objects observed in the robot working environment and (2) from

the human user's ability to use and interpret spatial commands. We represent the sensory information by an in-house developed original world state feature (WSF), which uses a spatial command as an RDT node. By applying machine learning to the examples, we are able generate a rule to find a desired move-to target for each kind of spatial command.

5.1.2 Organization

Details of the proposed framework are included. In Section 5.2, we introduce some related work on robotic language control (especially spatial language control) and robot learning. In Section 5.4, we describe our world state feature model, which is a quantified representation based on the spatial relations between objects detected in the environment by robots. Section 5.5 introduces the human demonstration interface, and work on how a policy for a spatial command was built by using PbD. Section 5.6 presents a robot learning experiment and results in an indoor environment. The experiment was run in both simulation and in the physical world. Its purpose was to examine the feasibility of using robot learning in a spatial language navigated robot. Finally, we conclude with a summary discussion and recommendations for future work.

5.2 Related Work

5.2.1 Natural language control robot

The literature addressing natural language control robots is limited but growing. Lauria et al. [48] developed an instruction-based learning (IBL) model grounding the natural language to robot understandable symbols, which worked using a road map. Matuszek et al. [49] proposed an idea to transform natural language commands to actions and control structures and trained a parser based on example pairs of natural commands and corresponding control language expressions. Fasola et al. [50] built a model to generate a

global path in a semantic map by using pragmatic fields and tested it in a pick-and-place task. Tellex and Kollar et al. [51-53] used a probabilistic graphical model, referred to as generalized grounding graphics (G^3), which used the structured nature of human language to learn actions from the command corpus and environment feature. To demonstrate moving action to a mobile robot, Skubic and Chronis [54-56] designed a method of using sketched route maps to navigate a robot, which first generated a linguistic description of the moving direction and then plotted a path on it.

5.2.2 Robot learning

The problem of learning a rule of mapping between world state and robot action, which is called "policy," is becoming a very challenging work due to the increasing complexity of robot tasks and working environments. Thus, the problems of policy programming are formulated to the development of machine learning system and skills [57]. In addition, there is an increasing need for an easy programming learning method for unexperienced users when a robot is becoming ubiquitous. Programming by demonstration (PbD) is an approach to robot learning, which was developed as a solution for a problem. In a humanoid robot, PbD has been an important method to generate a more stable bipedal gait by learning from a good teacher, i.e., a human [58]. PbD is also gaining popularity in industry as humans and robots cooperate as training robots learn techniques from the demonstration of human workers [59]. For work on language controlled mobile robots, Kollar et al. [60] developed an imitation learning policy to teach the mobile robot to detect the move-to target by command and observe the relevant world state. Their work simplifies the environment map by converting it to a graphic model, and the policy is to select a node that best matches the command. Compared to the previous work, ours is a user-oriented system

which has a friendlier and more efficient demonstration interface for non-expert users. Moreover, the robot can work in a partially unknown environment and support online training.

5.3 The Multi-Layer Model of Spatial Information

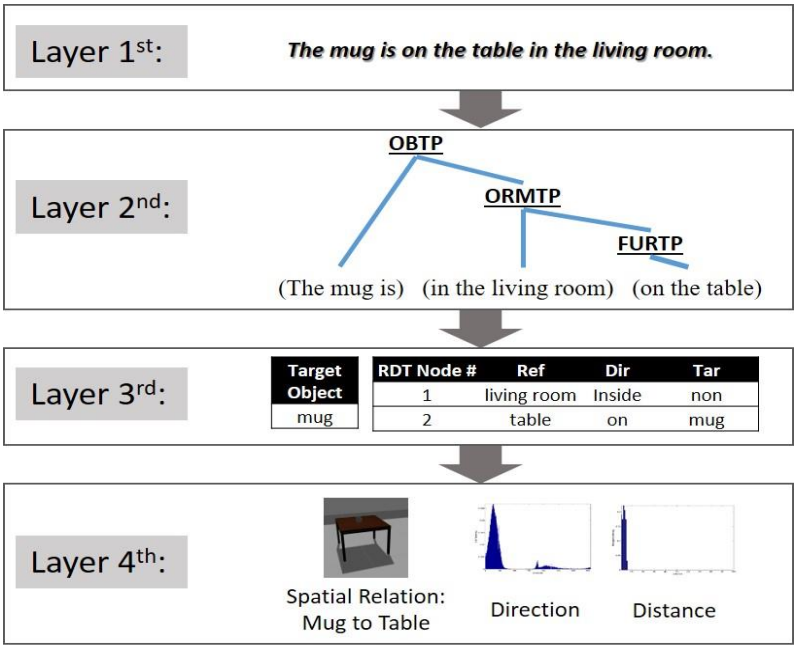


Figure 5.2 The four-layer spatial information model.

The spatial language grounding system builds a bridge between human spatial language and robot actions by converting the spatial information from text formatted natural language to numerical formatted robot control parameters. Due to the great complexity in spatial information, it is very difficult to model the spatial information from the natural language domain directly to robot action. Our system breaks the problem down into small steps. It shows the spatial information in four layers (Figure 5.2):

(1) The first layer is the natural spatial language.

(2) In the second layer, the words in the natural language description are grouped into chunks with meaningful tags by using part-of-speech algorithms [15, 39]. In this layer, the

words containing spatial information are detected and tagged, and the natural language is converted to a tree structure.

(3) In the third layer, the tree structure is translated into a grounding model in the form of the reference-direction-target (RDT) format which was proposed by [13, 14]. The RDT model eliminates the uncertainty and ambiguity in human language and conveys a robot understandable message of a sequential action list or reference-based descriptions which allows the robot to move to find the target object.

(4) The fourth layer is a numerical format representation called world state feature (the detail will be discussed in Section 5.4), which describes the spatial relations of both direction and distance between the objects in the environment. The data of this layer will be taken into the robot behavior model to infer a coordinate as the destination of the RDT node. After obtaining a move-to coordinate, the robot can be controlled by any path planning algorithm to move to the target. The system reduces the ambiguity of the spatial language and represents spatial information in a more understandable form to machine layer by layer.

5.4 World State Feature (WSF)

A robot behavior policy is a response to outside input by the robot. That is, it can be seen as mapping from the input information to the output action. For an indoor mobile robot, the input includes the user's command and sensory information from the outside environment. To give a robot concise and accurate information for following spatial directives, we designed a world state feature (WSF) model. The model registers the objects in the robot's working environment and the spatial relations between them. These spatial relations between objects are quantified as histograms in our model to the robot. They are

then connected to a certain RDT formed spatial command grounding model by training so that the robot can understand the spatial concept of a specific command. A WSF describes an instance of a spatial description situated in the environment using spatial relations. Each spatial relation is defined by a reference object, a described target object, spatial type (either distance or direction) and a numerical histogram. They work together as a unit and represent a spatial relation variable (SRV).

5.4.1 Entity

We use "entity" to describe objects in the robot's working space. In the WSF model, an entity $e=\{c,\rho,\theta\}$ includes the information of class name c, a 2-D point cluster ρ which draws the projection region of the object on the floor plan and orientation θ. We count four different kinds of entities, which appear in the WSF model's working environment. Figure 5.3 illustrates these entities, which are:

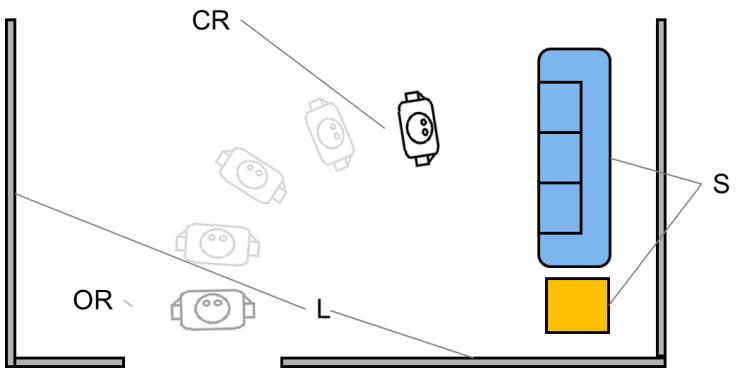


Figure 5.3 Four different kinds of entities: Going clockwise, we can start with CR (current-robot entity), then go down to the left-hand corner for OR (original-robot entity), travel straight across to L (long-term entity), and end up with S (short-term entity)

Long-Term Entity (LE): A long-term entity is a permanent structure in the environment such as a wall or door. Currently the long-term entities are all rooms or room structural elements. The front of an intrinsic coordinate is defined by the outside

direction of the exit. The long-term entities are recorded in a pre-prepared occupancy grid map.

Short-Term Entity (SE): A short-term entity is a small and movable object like furniture, an electric appliance, a plant, and daily objects. They do not have a fixed and known location in the robot's working environment and the robot will not store their positions before a task. They are detected and labeled in real-time during a fetch task.

Current-Robot Entity (CR): A current-robot entity represents the robot state at the current time. It is used only to assess the possibility of becoming the move-to target of a pose vector.

Original-Robot Entity (OR): An original-robot entity is the robot state at the starting time when following an RDT node.

Entities are not only identified according to a spatial scale but also by time in the robot working environment. For example, the CR entity describes the robot at the current time of observance, while the OR entity represents the starting time for the robot when it first received the command. Both entities describe the same object.

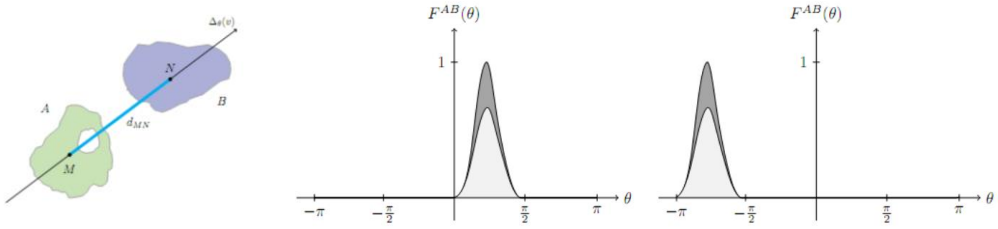
5.4.2 Spatial relation variable (SRV)

Spatial relation variable (SRV) is a quantified representation of the spatial relation between entities in the robot working environment. An SRV includes the following components: (1) the name of the reference entity e_r, (2) the described entity e_d, (3) the spatial relation type t and (4) the spatial relation feature vector f of that type. We use φ to denote an *SVR*, $\varphi=\{e_r, e_d, f_t\}$. The three types of SRVs are direction SRV (DIRSRV), distance SRV (DISTSRV) and rotation SRV (ROTSRV). The feature vector of direction or distance is the histogram of a linguistic spatial variable while rotation is the angle at

which an entity rotates from the origin time to current time. We only compute the rotation for the robot in our system. It should be noted that direction type and distance type of an SRV have directionality, which means that for two entities, A and B, the A-to-B SRV are usually not equal to B-to-A.

Histogram of Forces

The histogram of forces approach is used to compute the relative position between two objects. The objects used for this method can be either crisp or fuzzy. To measure the weight of the spatial relation in an angle θ that covers *A to B*, and assuming $\Delta_\theta(v)$ is a batch of vectors, of which have an interaction with *A* and *B*. The disjoint segment of the interaction that $\Delta_\theta(v)$ has with *A* and $\Delta_\theta(v)$ with *B* is the weight of angle θ.



(a) (b) (c)

Figure 5.4 (a) The two objects A and B, (b) histograms of forces for A wrt B, and (c) histograms of forces for A wrt B.

Assuming the two objects, A and B, with an angle θ, the theory of forces (Hof) shows the weight of how much "A is in direction θ of B". A typical histogram of forces is shown in Figure 5.4.

The histograms of constant (dark gray) and gravitational (light gray) forces for objects, A and B, are shown in Figure 5.4(b) and (c). By the histograms of forces the weight of each direction can be determined.

Direction Spatial Relation Variable (DIRSRV)

A DIRSRV describes the direction from a reference to a target by a histogram vector of weights on linguistic direction variables. Let $SVR_{dir}=\{e_r, e_d,, f_{dir}\}$ to denote an example.

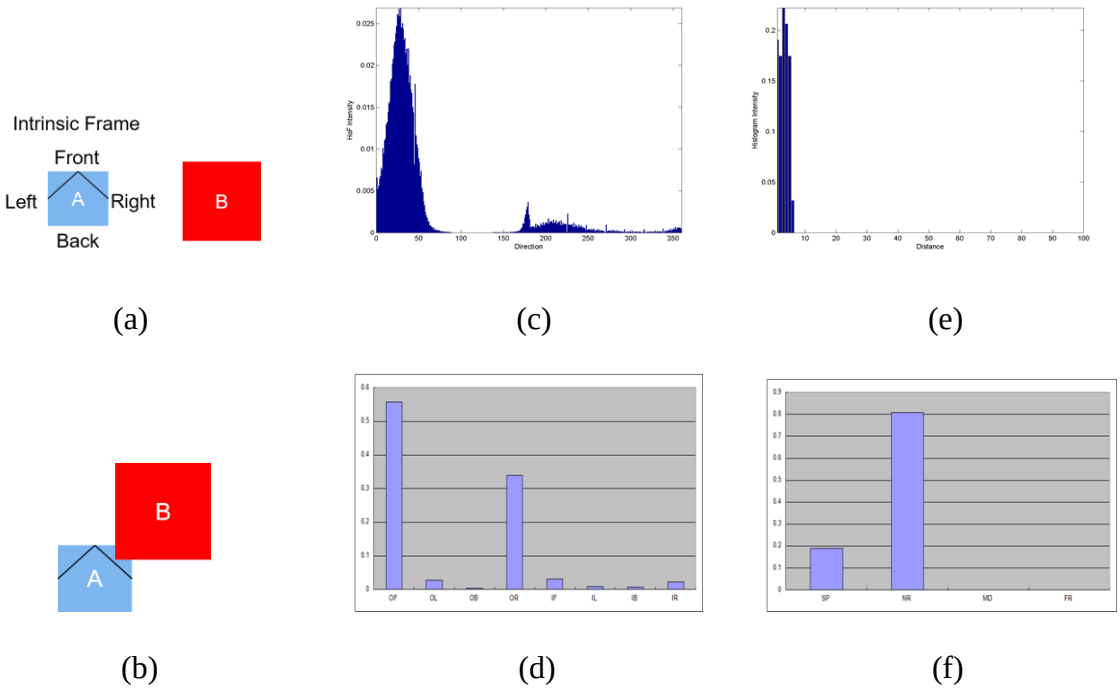


Figure 5.5 (a) The intrinsic frame of entity A, (b) entity A and entity B, (c) histogram of direction: B to A, (d) DIRSRV: B to A; (e) histogram of ditance: B to A, and (f) DISTSRV: B to A.

To compute the vector *f*, we first use the theory of histogram of forces (HoF) as described in [61] to compute the feature vector of the direction type SRV. The HoF of e_d to e_r is calculated by the method introduced by [62]. An HoF is a 181-length array, which represents the weights of equally divided directions from 0 to 360 degrees. Assuming two objects, A and B, (Figure 5.5(a)) has the spatial relation like Figure 5.5(b), let us define

h(A,B) as the HoF vector B to A. In our system, we consider A to be the e_d and B as the e_r. To draw a distinction between the two situations where e_d is inside and outside of e_r, we separately compute the HoF by dividing e_d into two parts, with e_{di} (inside part) and e_{do} (outside part). Then we have $h(e_r,e_{di})$ and $h(e_r,e_{do})$ (Figure 5.5(b)) and can connect them as a 362-length vector. We do this because the HoF vectors for these two situations are in the same form even though they are different in the spatial relation and spatial concept. After we compute the two HoF vectors, we weight them by the area ratio. The new HoF vectors F_i ad F_o are expressed as:

$$\boldsymbol{F_i = \frac{h(e_r,e_{di})}{Sum(F(e_r,e_{di}))}\frac{N_{di}}{N_{di}+N_{do}}} \quad \textbf{5-1}$$

$$\boldsymbol{F_o = \frac{h(e_r,e_{do})}{Sum(F(e_r,e_{do}))}\frac{N_{do}}{N_{di}+N_{do}}} \quad \textbf{5-2}$$

where N_{dx} represents the number of points in the x part of the entity e_d. The sum is the summation of the HoF vector. Then, we can obtain $\Sigma F_i+\Sigma F_o=1$, which normalizes the HoF vectors.

To increase the execution speed, we do not directly use HoF as a direction feature but define eight linguistic direction variables *front*, *left*, *back*, *right*, *inner-front*, *inner-left*, *inner-back* and *inner-right* (Figure 5.5(c)), and transform the HoF vectors according to their weight. The weights show how much the HoF can support those directions. For a specific direction q, assuming that the angle ρ has the highest weight to support that direction, we define the weight of q by the HoF vector F, which is expressed as:

$$\boldsymbol{w_q = \sum_{x=\rho-\frac{\pi}{4}}^{\rho+\frac{\pi}{4}} \frac{\|x-\rho\|}{\pi/4} F(x)} \quad \textbf{5-3}$$

and the DIRSRV vector f_{dir} is:

$$\boldsymbol{f_{dir} = [w_{outer-front}, \dots, w_{inner-right}]} \quad \textbf{5-4}$$

This transform reduces the computation but keeps the directional information so that it is possible to support decisions in real time.

Distance Spatial Relation Variable (DISTSRV)

A DISTSRV describes the distance from a reference to a target by a histogram vector of weights on different distance variables. To build a DIST*SRV={*e_r, e_d, f_{dist}*}*, we first generated a 100-length vector, which is the distance histogram of e_d to e_r for the range from 0 to 10 meters with an 0.1 meter interval (Figure 5.5(d)). Let L_{dist} denote the distance histogram. The equation to represent the unit at distance x in this vector is:

$$\boldsymbol{L_{dist}(x) = \frac{N_{d,x}}{N_d}} \qquad \textbf{5-5}$$

where $N_{d,x}$ is the points number of e_d, of which the minimum distance to e_r is x.

By the same procedure, we run on DIRSRV, where the distance histogram vector is transformed to a vector of four linguistic distance variables: *superposition, near, middle and far* (Figure 5.5(e)), which represents four common distance relations between two objects. The weight value for each distance variable r is obtained by the equation:

$$\boldsymbol{w_r = \sum_{x \in X_r} L_{dist}(x)} \qquad \textbf{5-6}$$

and the DISTSRV vector f_{dist} is:

$$\boldsymbol{f_{dist} = [w_{superPosition}, \cdots, w_{far}]} \quad \textbf{5-7}$$

where X_r is the distance range of the linguistic variable r.

For a certain distance, x, all the linguist variables are given positive weights so that the distance variable vector f_{dist} can continuously change with the L_{dist}.

Rotation SRV (ROTSRV)

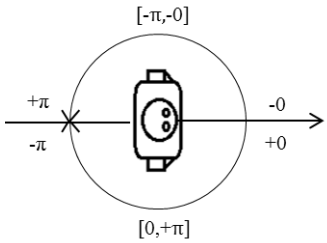
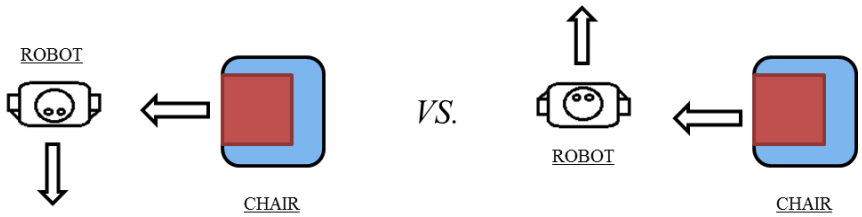


Figure 5.6 Rotation angle coordinate.

Rotation SRV is defined by the difference of orientation angle between two entities. This spatial property is not easy to describe by DIRSRV or DISTSRV. It ranges in the scale of $[-\pi,\pi]$ (Figure 5.6). Since we only have the robot movable during a robot task, we will use ROTSRV to describe the rotation from OR to CR.



(a) (b)

Figure 5.7 The asymmetry of DIRSRV: For the two cases shown in (a) and (b), $DISTSRV_{robot\_to\_chair}$ are the same, but $DIRSRV_{chair\_to\_robot}$ are different.

It should be noted that the mirrored spatial relations pairs, which have the same entities, are not one-to-one correspondence. For example, as shown in Figure 5.7, we assume the robot is a round shaped object when the robot entity $e_r=\{c,\rho,\theta\}$ rotates with θ as it changes. The ρ remains the same. Then the SRV using e_r as the described entity will not change. In addition, if the reference entity has no intrinsic direction defined, there will be no DIRSRV defined by it. To reduce the computation, we ignore the SVRs composed by unrelated entities and only keep the CR, OR, *reference* entity (*ref*) and *target* entity (*tar*) as they appear in the RDT node in the SRVs. The possible SRV to be generated in a world state feature includes:

DIRSRV: Cartesian_product({CR,OR, ref, tar}, {CR,OR, ref, tar}).

DISTSRV: Cartesian_product({CR,OR, ref, tar}, {CR,OR, ref, tar}).

ROTSRV: *CR to OR*

After we collect these SRVs in the environment, we build a WSF which is a bag of them. The WSF is then used as the input of a policy to decide an action.

5.5 Train the Robot Behavior Policy Model

For our robot system, we define the procedure of a fetch task as a sequence of RDT nodes. The workflow of the action for each node is shown in Figure 5.8, which includes four steps: (1) Collect sensory data; (2) build a world state feature (WSF) model; (3) make decision on the move-to target of the current RDT; (4) drive the robot to the target location.

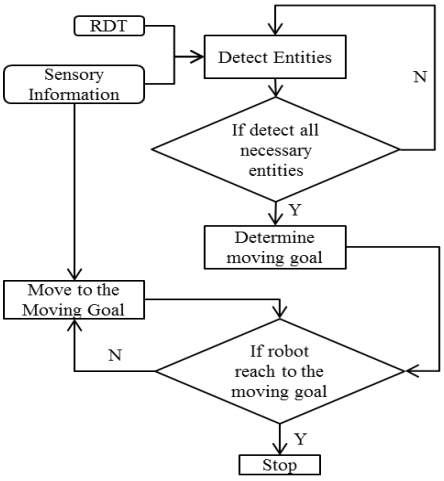


Figure 5.8 The flowchart of executing an RDT node.

The robot behavior policy model is divided into two levels. The higher level is to detect the move-to target for the current RDT, and the lower level is path planning and robot moving control. The reason for building a hierarchical system is that it is difficult to build an immediate model of the mapping from the input spatial command and the world state to the output of robot speed control parameters. However, for different RDTs, we can use the

same scheme program for path planning and robot wheel control. And for the high-level part, we will use robot learning to generate the policy model for each kind of RDT.

5.5.1 Programming by Demonstration

The work to build the high-level part of the policy model for our robot is based on a general model of programming by demonstration (PbD). PbD uses demonstration examples to build policies, which can drive robots to take the same action as the example cases. Let W denote the world state, μ be the spatial command sent to the robot, and A be the action the robot should take. The policy model function f with command μ for the robot can be written as $A=f_\mu(W)$. When we use PbD to f_μ for the command μ, we show the command μ to the human demonstrator and record the corresponding action A_d and world state W_d. Then, we use machine learning algorithms to derive the f_μ by A_d and W_d. The diagram of our robot learning model is shown in Figure 5.9.

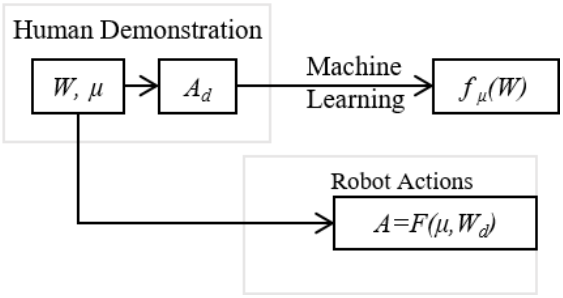


Figure 5.9 PbD procedure diagram.

In our system, the high-level policy model works as a probability model, and the goal of the policy is to detect a pose where the robot has the highest probability as the user's expected move-to target. For a command μ, we set $p(W)$ as the probability density of the user's expected world state feature W. The robot can build up the environment mode and change W by perception and moving. After the robot gets enough sensory information, the

policy model for the robot is to find the *CR* and let the *W* containing *CR* serve as the most possible move-to target. The target *CR* is:

$$\boldsymbol{CR_{target} = argmax_{CR}\, P(CR = target|W_{CR})} \quad \textbf{5-8}$$

To find the *W* which can best follow μ, we first need to construct *W* by entities and their corresponding *SRVs*. We denote a set $\varepsilon=\{e_1,e_2,\ldots e_N\}$ as a set of *N*-related entities for command μ. Then, we compute *SRVs* from ε to form the set *W*. Let $\varphi_k(e_x,e_y)$ be the *k* type *SRV* of entity *x* to entity *y*; then,

$$W=\{\varphi_{DIRSRV}(e_1,e_2),\varphi_{DIRSRV}(e_2,e_1)\ldots,\varphi_{DISTSRV}(e_1,e_2),\varphi_{DISTSRV}(e_2,e_1),\ldots\varphi_{ROTSRV}(CR,OR)\}.$$

Using Bayes rule, we have the equation:

$$\boldsymbol{p(CR = target|W_{CR}) = \frac{P(W_{CR}|CR=target)p(CR=target)}{p(W_{CR})}} \quad \textbf{5-9}$$

Write W_{CR} as the form of a WSF having an *M* number of SRVs , and we have:

$$\boldsymbol{p(CR = target|W_{CR}) = P(CR = target|\varphi_1, \varphi_2, \ldots \varphi_M)} \quad \textbf{5-10}$$

Because the SRVs are independent from each other, we have:

$$\boldsymbol{P(CR = target|\varphi_1, \varphi_2, \ldots \varphi_M) \propto \prod_{m=1}^{M} P(\varphi_m|CR = target)} \quad \textbf{5-11}$$

That means we can decide the move-to target by computing $P(\varphi|CR=target)$ as the distribution of φ when the robot is in the move-to target mode.

5.5.2 Human Teaching Interface

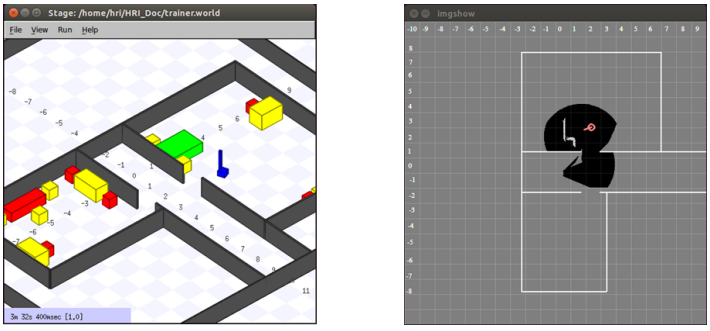


Figure 5.10 Stage-based human demonstration interface.

To teach a robot the spatial navigation concepts of a human, we built an interface to collect human demonstration samples. The system was developed as a robot simulator based on ROS stage [63, 64]. The simulator runs a mobile robot in a two-room apartment environment shown in Figure 5.10. The apartment environment is furnished by common furniture items such as a bed, couch, chair, table and the target object. When giving a demonstration example, the human demonstrator is allowed to use a keyboard to control the movement of the robot and has the same vision scale of the environment with object recognition ability. The robot is placed at the middle of the hallway at the beginning of the demonstration. Then the human teacher controls the robot by the guidance of the sequence of RDT nodes. The system simultaneously records the environment model acquired by perception and the robot state during the demonstration. The demonstration procedure is shown in Figure 5.11.

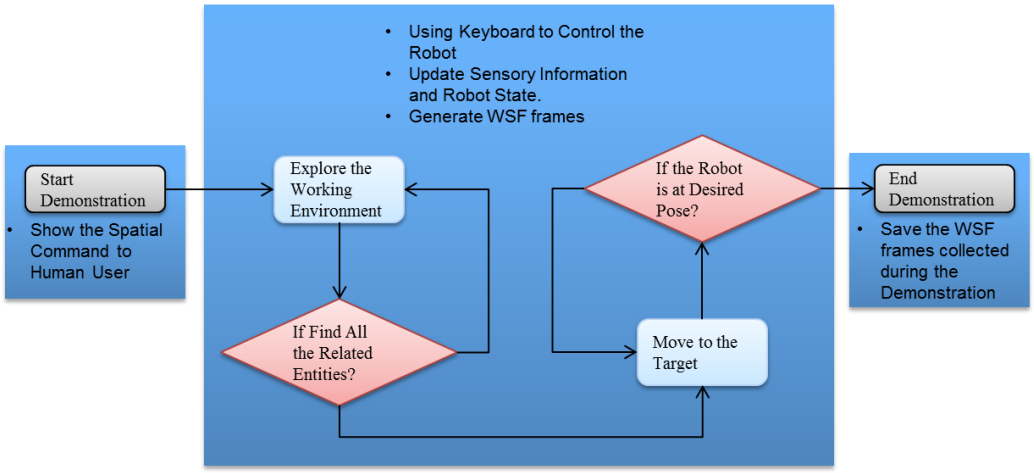


Figure 5.11 The flow chart of a demonstration

5.5.3 Train the Robot Behavior Policy Model

In our system, the output of the policy model controls the behavior of the robot, which is to decide how to achieve a target pose that most accurately fits the requirement of a

spatial command. Due to the variety of concepts concerning spatial commands, a policy model is exclusive for a spatial command. After the inference of the move-to target, the robot can then plan a path and be controlled to arrive there by implementing a universal path planning and moving control approach. The process of learning a policy is the training of a classifier with the WSF feature. Given the independence of the SRV in the WSF feature, we can build a classifier for each kind of SRV and fuse them together as a WSF classifier. For a WSF of a SRV, we define two labels, "yes" or "no" to guide the move-to target, so the classifiers are binary. Compared to a classical binary classification problem, there are two challenges in our WSF classification model. One is that it is difficult to label the training samples. There are multiple WSF features collected during a demonstration while only the first (the robot at the starting pose) and the last (the robot at the move-to target) WSF can have crisp labels (First: "no"; Last: "yes"). Another challenge is that the size of the training sample set of a spatial command is too small, and it is very easy for the classifier to become trapped into a local optimization or overfitting. To overcome these difficulties, we modified the parameter updating step in the training procedure. Assuming a demonstration example with $t+1$ frames, we only used the WSFs, W_0 *and* W_t, which are the first and last frame for updating. W_t is a positive sample, and **all** the SRVs fulfill the move-to target, while W_0 is a negative sample and **not all** the SRVs fulfill the requirement of the move-to target.

The problem must look to the world state feature, which can maximize the value of the objective function. However, we cannot individually tell the distribution of probability for each SRV as a target state. Moreover, the robot may observe some "unrelated" SRVs, which means the distribution of the SRVs is not correlated to the distribution of the move-

to target. Since there is only a limited number of demonstration examples for training, the model to train may be very difficult to converge if we use a tradition supervised learning framework with Gaussian kernel and gradient descent to adjust the model parameters.

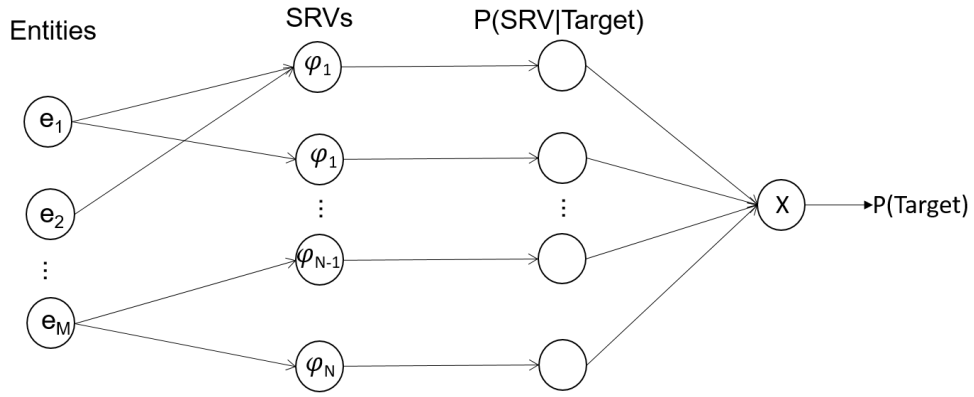


Figure 5.12 The probability inference model for the determination of a move-to target.

Instead, to solve this problem we define a linear form approximate representation model of the distribution of SRV. The SRV value, which has a higher probability of being a target, will get a higher score for the model. We will also import a punishment function to pick out "unrelated" SRVs. For the diagram of the probability model as shown in Figure 5.12, the distribution of a SRV vector $\Phi=\{\phi_1,\ldots,\phi_k\}$ is $p(\Phi|target)=V\Phi$. The linear vector $V=\{v_1,\ldots v_k\}$ is the parameter to learn. For training, we let the vector V starts to be all zero. The update of v is:

$$\boldsymbol{v_k(t+1) = max(v_k(t), \phi_k(t))} \quad \textbf{5-12}$$

Then after all the training loops, we let

$$\boldsymbol{v_k = \frac{v_k}{\max(V)}} \textbf{5-13}$$

The probability of a k-length vector SRV as a target is:

$$\boldsymbol{y = \prod_{k=1}^{K} v_k \phi_k} \quad \textbf{5-14}$$

and the probability of a world state feature $W=\{\Phi_1,\ldots,\Phi_M\}$ with M *SRV*s is:

$$z = \prod_{m=1}^{M} V_m \Phi_m \quad \textbf{5-15}$$

z is in the value of [0,1].

The result of *v* has the following properties:

- Because the maximum of *y* is equal to 1.0, the scale of the output is [0, 1] which can be the probability of distribution for an SRV.
- The more similar the SRV is to the target state, the closer the *y* is to 1.0.
- By training the robot in different environment settings, e.g., from a different starting point, in a different room or with a different reference and target furniture, the value of *z* will be closed to 1.0 at all the settings, which means the distribution of *P(target|Φ)* is flattened.

To improve the computation speed and reduce the risk of false negatives, we use a punishment function on "unrelated" SRVs and remove them. The character of an "unrelated" SRV is that the *P(target|Φ)* is very close to 1.0 for every possible *Φ*. It means the distribution of *P(target|W)* and *P(target|W-Φ)* are very close. Therefore, we can remove the component of *Φ* in our probability model. It not only simplifies the model but also reduces the risk of a false negative. From such a character of an "unrelated" SRV, we can score an "unrelated" rate *u* of an SRV by the equation of:

$$u = \frac{\sum_{k=1}^{K} v_k}{K} \quad \textbf{5-16}$$

where *K* is the length of an SRV vector. We will remove an SRV from the WSF if $u > 0.8$

5.5.4 The Move-to Target Inference

After getting the policy model, we will use it to search for the move-to target. In our system, the robot can affect the world state feature by moving to different poses and

changing the *CR* entity. When *CR* changes, the SRVs that involves the *CR* changes as well. Therefore, the problem changes to finding the target *CR*. To find the *CR*, we should try to compute each possible *CR* value which is almost impossible because it takes too much time. We implement the spirit of particle filter which is an iteration method to gradually reduce the searching scale and find the target *CR* that most closely matches a move-to target. The steps of searching are shown below:

1) Initialize the searching start pose p_0 at the current pose, searching scale *S*, which is equal to the working space *D*. Begin the initial searching interval. I_0 = *{1m, 45 degrees}*,which means that during the searching process, CR will shift 1 m in either the x or y axis and 45 degrees in robot rotation.
2) In iteration, *i*, select pose p_i with maximum output of $z(w_{CR})$. Let $D=p_i+\{[-1,1]\}$, $I=I/2$.
3) Terminate the iteration when $z_i < z_{i-1}$.

The procedure to search for a move-to target is like the scene shown in Figure 5.13

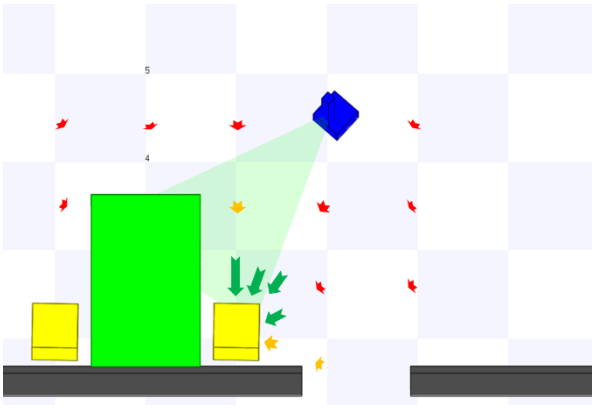


Figure 5.13 The probability map of the RDT node “bed-right-table” (on the table to the right of the bed). The color of an arrow turns from red to green and the length grows with the increase of probability.

5.6 Evaluation

The goal of our evaluation is to assess whether a robot can be taught to build policy models which follow the spatial commands by human demonstrations and how these

learned policy models perform in a robot task loop. We designed two experiments for evaluation. One was the robot training assessment which directly examines the trained robot behavior policy model in a robotics simulation platform. Another assessment is the end-to-end test, which lets the robot use the learned policy models in a fetch task. The test was run on a physical robot. We constructed a different environment from the training scene to test the robot.

5.6.1 Robot Training Assessment

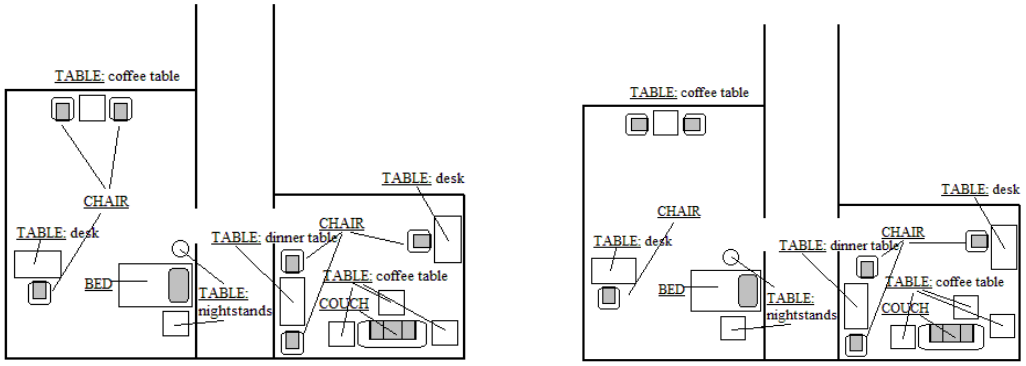


Figure 5.14 (a): The map for training; (b) The map for testing.

In this assessment, we tested the performance on policy model training. We selected 13 RDT nodes from the CSISL corpus (discussed in Chapter 2, section 2.3), which has seven different references. The robot behavior policy model was trained by the human user, who used the interface discussed in Section 5.5.2 and the map shown in Figure 5.14(a). Each RDT was trained by five demonstrations, which were given different robot starting poses. To test the robot, we slightly changed the arrangement of the training environment to be like Figure 5.14(b). For a test on each RDT, we randomly selected the robot starting point and drew an expected target pose in each training scenario. Table 5-1 shows the performance of three tests for each policy model. In this table, *d* is the distance between

the ultimate position of the robot and d is also the expected target position. θ is the angle between the direction from the robot to the target object and the direction that the robot faces toward. The first two columns show the natural language command of the RDT model. The third and the forth columns show the values of the mean and standard deviation of d and θ for each kind of policy model. The metric T at the last column shows the number of times that the target furniture can be captured by the robot depth camera. The T value will be '-' when T is invalid for a non-target command.

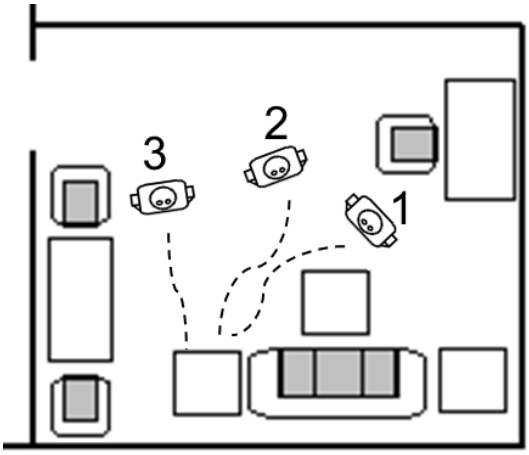


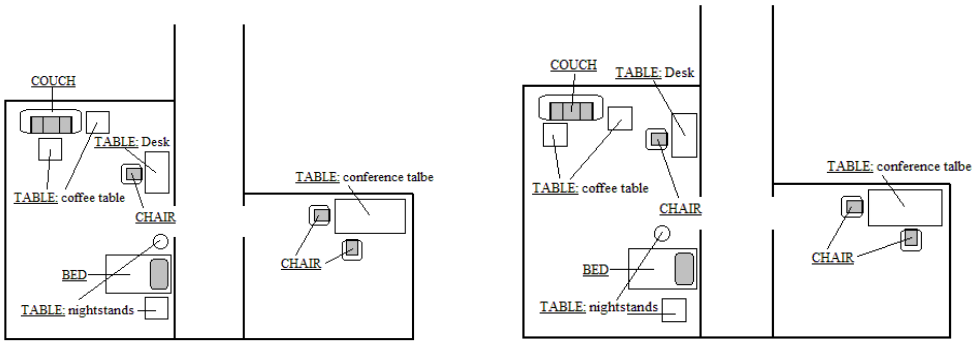
Figure 5.15 The three training trials to train the policy model of RDT "*couch-left-table.*"

Table 5-1 The result of RDT node policy learning. NSLD: Natural Spatial Language Description; RDT: reference-direction-target model; d: The distance distance between the ultimate position of the robot and expected target position, θ, is the angle between the direction from the robot to the target object and the direction that the robot faced toward, T, and the number of times that the target furniture can be captured by the robot depth camera

#	NSLD	RDT	d (m) (Mean/STD)	θ (rad) (Mean/STD)	T (Success/Total)
1	go/walk/move forward	move-front-non	0.55/0.34	0.03/0.01	-
2	turn left	move-left-non	0.03/0.06	0.03/0.01	-
3	turn right	move-right-non	0.03/0.06	0.02/0.01	-
4	table on the/your left	robot-left-table	0.36/0.24	0.13/0.22	3/3
5	the table is on/to the/your right	robot-right-table	0.66/0.28	0.38/0.03	3/3
6	table on/to the left of the bed	bed-left-table	0.20/0.05	0.08/0.09	3/3
7	table on/to the right of the bed	bed-right-table	1.01/1.47	0.34/0.41	3/3
8	table in front of the couch	couch-front-table	0.35/0.19	0.63/0.46	3/3
9	table to/on the left of the couch	couch-left-table	0.08/0.03	0.02/0.01	3/3
10	table to/on the right of the couch	couch-right-table	0.73/0.33	0.77/0.75	3/3
11	table beside a chair	chair-beside-table	0.53/0.08	0.14/0.18	3/3
12	go to the center of the room	room-center-non	0.91/0.31	0.67/0.82	-
13	move to the front wall	wall-front-non	0.67/0.29	0/0	-

5.6.2 End-to-end Assessment on the Physical Platform in the Training Environment

To make evaluation of our system, robot training assessment is not enough because we need to prove that the robot can work on a fetch task by relying on the learned policy model. Therefore, we ran an end-to-end assessment on our robot system. An end-to-end assessment requires testing the whole system, which includes all of its components to ensure that each is functioning as intended. The functional side requires that we run such an assessment to check against requirements, and it is more about the actual flow through a system in a more realistic end-user scenario. This assessment tested the entire workflow of our robot system, which gave an overall view of the system performance.



(a) (b)

Figure 5.16 (a) The end-to-end map and (b) an altered end-to-end to test the the robustness of our system.

We ran all the end-to-end tests inside the test room. In each test, we placed a mug on a table and let it be the target of a fetch task. The robot first stood at the door of the room. Then, it was given a description of the location of a target object. The natural language description was then parsed to an RDT form of navigation instructions, and the robot used its learned policy model to follow the RDT nodes to find the target object. A spatial command may contain at least one RDT. We manually edited 12 spatial commands, which navigated the robot to move to six different move-to targets. For each move-to target, we edited two commands. One was the dynamic command which gives the robot movement description [9]. The other one was the static command, which uses a reference to describe the move-to target [9]. The editing of words and phrases in a command refers to our template corpus in [14]. The content of commands did not contain any ambiguous information which could confuse the robot. The experiment environment map is drawn in Figure 5.16(a). The tests were run in both the simulated and physical world. As in our previous work in (Huo 2014, p. 81}, we slightly changed the placement of furniture and reran the experiment again (Figure 5.16(b)). These changes can still be described by the

same spatial commands. This test was to validate that the robot behavior policy model is adjustable to an alternative environment. To measure the robot performance, we used *d*, which is the distance where the robot end points toward the mug, and *v* is the area of the mug, shown in the robot vision. Then we used *r* to represent whether we determined this task successful or not. The end-to-end results are shown in Table 5-2.

Table 5-2 Results of the end-to-end test.

Target Object Placement	NSLD	RDT	Furniture Placement	d (m)	V (0-1)	S/N
1	Move to the center of the room, turn right. Then go forward. The mug is on the table in front of the couch.	room-center-non move-right-non move-forward-non couch-front-table	#1	1.07	0.96	Y
			#2	1.10	0.58	Y
	Go forward and then turn right; then move forward and then turn left. You will see the mug	move-front-non move-right-non move-front-non move-left-non	#1	2.60	0.92	N
			#2	2.09	0.91	Y
2	The mug is on the table on your right	robot-table-right	#1	1.03	0.72	Y
			#2	1.25	1.00	Y
	Turn right, and you will see the mug	move-right-non	#1	1.27	0.92	Y
			#2	1.25	1.00	Y
3	Turn left. You will see the mug on the table to the right of the bed.	move-left-non bed-right-table	#1	0.97	0.69	Y
			#2	1.35	1.00	Y
	Turn left and you will find the mug	move-left-non	#1	1.16	0.93	Y
			#2	1.14	1.00	Y
4	Move to the center of the room and turn right; then, go to the front wall. The mug is on the table to the left of a couch	room-center-non move-right-non wall-front-non couch-left-table	#1	1.57	0.81	Y
			#2	1.15	1.00	Y
	Go forward and then turn right and walk forward; you will see the mug	move-front-non move-right-non move-front-non	#1	1.48	0.72	Y
			#2	1.41	0.83	Y
5	Go forward the turn left. Move to the front wall then go left again and you will see the mug on the table to the left of the bed.	move-front-non move-left-non wall-front-non move-left-non bed-left-table	#1	0.40	0.92	Y
			#2	0.46	0.90	Y
	Go forward and then turn left. Move forward again then turn left. Then go forward and you will see the mug	move-front-non move-left-non move-front-non move-left-non move-front-non	#1	0.51	0.98	Y
			#2	0.81	0.94	Y
6	Go to room center. The mug is on the table beside a chair	wall-front-non chair-beside-table	#1	1.67	0.59	Y
			#2	1.74	0.58	Y
	Walk forward and then you will find the mug.	move-front-non	#1	1.68	0.78	Y
			#2	1.68	0.78	Y

The average and stand deviation of *d* and *V* are shown in Table 5-3.

Table 5-3 The average and STD of *d* and *V*

	Mean (m)	Standard Deviation (m)
Distance	1.09	0.70
V	0.87	0.14

5.6.3 End-to-end Assessment on the Extended Simulation Environment

To further validate the performance of our system, we ran another experiment on the 3D simulator introduced in Section 3.4. We manually edited another 79 spatial language commands and descriptions of the 24 target objects (six for each world) following the syntax and morphology of the CSISL corpus. The words and phrases used in the new spatial descriptions are all in the CSISL corpus setting. The overall success rate is 77.2% (61/79). The raw spatial language and results are shown in the Appendix Table A1, and the number of failed cases is shown in Table 5-4. There are four kinds of failures observed in the end-to-end experiment. The most failed cases in the experiment was the "not visible" failure. In these cases, the robot moved to a position which was too far from the target object or not towards the target object so that the object was not visible. This is because the system failed to infer a good move-to target for the scene. There were four failure cases caused by incorrect part-of-speech tagging. Since the Brill tagger previously had a bad performance with cases outside the training data and we used some words and phrases not included in the CSISL corpus in these cases, it is not surprising to have these failures. The perception failure cases are due to the wrong recognition on the furniture samples. The collision failure cases were caused by failure in map path planning. The ambiguity failure case was caused by the ambiguous language used in the spatial description.

Table 5-4 The numbers of failed cases.

Item	Number

Success	61
Failure: not visible	7
Failure: part-of-speech (pos)	4
Failure: Perception	4
Failure: Collision	2
Failure: Ambiguity	1
All	79

5.7 Conclusions and Future Work on Robot Policy Models

This chapter presented an approach to build policy models for different spatial commands. We developed the world state feature (WSF) model and employed it to convert a human demonstration to an understandable representation of spatial concept for robots. The approach allows a non-expert user to teach robots how to follow natural language commands. Two primary benefits came from this study: 1) a robot developer is no longer needed to do the time-consuming and challenging work of manually building up the robot behavior policy, and 2) it is now easy for the user to add new spatial commands to the robot for a new working environment.

In the first experiment, the system performed well overall. In some failures, the robot detected more than one reference object or target, which led to an ambiguous selection of entities in building spatial relations. If unrelated entities are learned, the path navigation may fail. We will extend our demonstration environment and run more demonstrations to improve the detection performance of reference and target objects.

The second and the third experiments showed that robots can always capture the fetch target when finished executing the command following actions. This indicates that the efficiency of the policy model is good enough to replace the hard-code policy model for an end-to-end task. In a dynamic spatial language case, the robot failed in approaching the target object. The reason is that errors in the move-to target command were accumulated

during the navigation without any correction causing the robot to move in the wrong direction. This problem will be addressed by designing a retrieval mechanism in the next development. The system has shown the possibility of using PbD to teach robots the spatial concepts of human users. We will continue to work on improving the system to robustly handle spatial commands. Future research will use the approach developed here to train robots in a more complex environment.

Chapter 6. Natural Spatial Language Generation

6.1 Introduction

This chapter introduces a spatial language generation system which can generate the spatial description of the position of a target object using the state and perception information of the robot. The second section of this chapter is the RSS workshop paper [17], which is reprinted word for word and in full. It illustrates the details of the system and the preliminary results. Section 6.3 is the result of the follow-up experiment which has 48 language generation tasks in the different working spaces. The last section is the conclusion.

6.2 RSS2016 Workshop Paper: Natural Spatial Language Generation for Indoor Robot

Abstract—This paper proposes a spatial language generation system to find short, accurate and human-like descriptions for robots to communicate with a human user about the location of an object. The research focuses on building static spatial descriptions which use reference objects and directions to describe spatial relations. The system generates a natural spatial description in three steps. In the first step, it collects the sensory information and robot state to extract an environment model. Then, it builds a grounding model that describes the location of the target object, based on landmarks in the scene. After that it will generate the natural language description by imitating a human's talking style. A corpus of 149 spatial language commands for an indoor environment fetch task is used to train the system. An early-stage experiment was conducted and the results illustrate good potential for further development.

Keywords-spatial language; language generation; robotics

I. INTRODUCTION

The interest in how a robot can be of assistance in our daily life continues to grow. For the robots working on household tasks, there is an increasing need for the capability to interact with human users; the interaction using spatial language is getting more attention from researchers. For robots that can interact with humans using spatial language, there are two complimentary robot challenges in a home-like environment. One is understanding natural language directives. For example, a human user directs a robot to fetch a target object by giving a spatial command. Another is spatial language generation, which lets a robot answer to a human user with the location of a target object by using natural spatial language. This paper focuses on the second challenge by building a language generation system for indoor robots.

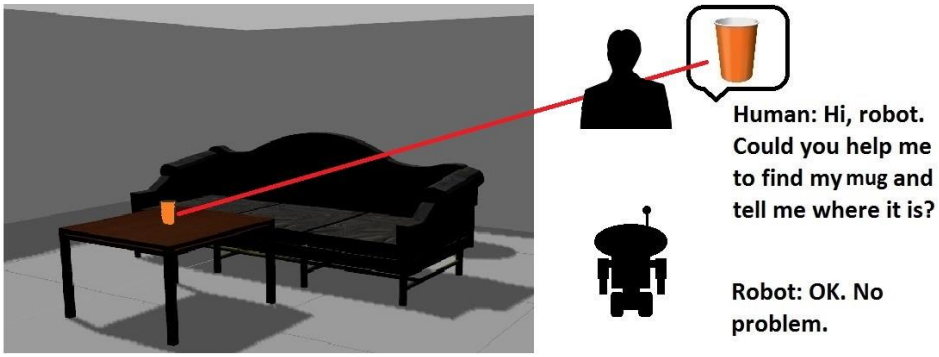


Figure 1. The scenario of an object searching and language generation task performed by a robot in a home environment. The Human said: "Hi, robot. Could you help me to find my mug and tell me where it is?" The robot answered: "OK. No problem". Then the robot left to search for the mug.

Figure 1 shows an example of the spatial language generation task performed by a robot in an indoor environment. The human user is standing in the hallway between the living room and the bedroom, and he wants the robot to find the mug and tell him the location of it so that he can easily go right to it when he needs it. In this scenario, the human user expects the robot to give a description like "*Walk into the living room, then turn right and*

move forward, you will see the mug on the table," or "*The mug is on the table in front of the couch in the living room*" which is a natural and friendly way of assisting and provides enough information to assure successful retrieval. Here, we focus on the generation of static spatial language which is the second example sentence above. The concept of static spatial language has been introduced in [1]. A spatial description of this type uses objects as references to describe a target location, i.e., "*behind the couch*" or "*on the table next to the bed*". The language generation task for indoor robots uses the sensory information collected from the environment to generate the static spatial language description. The generated description includes the spatial information in a large area so that it may be long and may have a complex structure which will make it difficult to be generated by a language template. This makes it different from other work on robot language generation and makes it a more challenging task. However, this kind of spatial language is human-like and provides more intuitive navigation information for a human user, particularly an elderly user.

There has been some significant work on the language generation. Reiter and Dale systemically described the approach to generate natural language with a probabilistic system [2]. Chen and Mooney presented a novel algorithm, Iterative Generation Strategy Learning (IGSL), for deciding which events to comment on in a soccer game [3]. The work in [4] introduced a novel model to generate spatial language. Angeli, et al proposed a multi-layer system generating natural language by two steps: content selection and surface realization [5]. Our work is the generation of spatial language for robots in an indoor environment which is a different task. However, all of the work faces the same problem, which is generating human-like language using raw and unabstracted data. In the related

work by Angeli et al., the process of language generation is split into two steps: the first one is content selection which selects the information to present from the raw data; and the second one is surface realization which infers the natural language from the selected content.

To enable the robot to provide easily understood spatial descriptions to a human user, we designed a multi-step system that follows the two steps mentioned above. The system first models the content of groundings from the sensory information collected in the environment, and then generates natural language from this intermediate result.

II. METHODOLOGY

A. The Multi-Layer Model of Spatial Description

The language generation system is based on our previous work on modeling spatial language and understanding spatial language directives, which has been developed to be a multi-layer system [6]. This system represents a natural spatial language description using four layers (Figure 2). The first layer is the natural language command. In the second layer, the words in the natural language description are grouped into chunks with meaningful tags by using part-of-speech algorithms [7]. In this layer, the words containing spatial information are detected and tagged, and the natural language is converted to a tree structure. In the third layer, the tree structure is translated into a grounding model in the form of the reference-direction-target (RDT) format presented in our previous work [1][6]. The RDT model is a standard representation with the information of landmark and spatial relation. In the RDT model, reference refers to an object that is used as a landmark reference to describe the location of another object. Direction represents the position relationship between objects, e.g., in front or to the left. It tells the robot where to search

for the target. Target indicates the target furniture or target object being sought by the robot. It minimizes the uncertainty and ambiguity in human language and conveys a robot understandable message to let it seek the destination. Given a long spatial description with a complex structure, the chunks in the tree structure can be converted to RDT nodes, which describe a sequential action list or reference-based descriptions allowing the robot to move to find the target object. The fourth layer is a numerical representation of the spatial relations of both direction and distance between the objects in the environment. The data of this layer will be taken into the robot behavior model to infer the destination of RDT node.

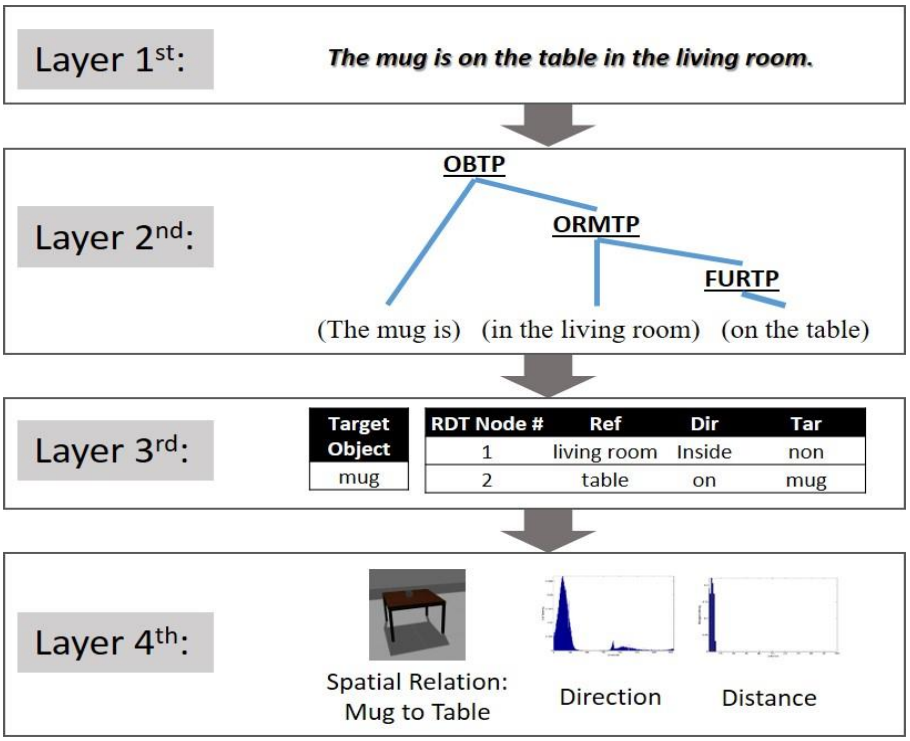


Figure 2. The multi-layer model of spatial language in our system.

B. System Overview

The goal of this work is to generate a natural language description of the position of a target object. For the example shown in Figure, the expected corresponding description is

"*The mug is on the table in front of the couch in the living room*". The spatial description contains information about the environment. To deliver the position of the target object to a human user correctly, the robot should detect the environment and extract the spatial information that can best describe the position and then present them in natural language terms. In such a task, we let ε denote the information of the environment, and p denote the location and the orientation of the human user. Consider an objective function $h(\varphi,p,\varepsilon)$ of the natural description φ. The robot will search for a spatial description φ' with the largest function value:

$$\boldsymbol{\varphi' = argmax_{\varphi} h(\varphi, p, \varepsilon)} \quad \textbf{(1)}$$

The objective function determines the policies to select a spatial description which should: a) have accurate information for the human user to reach the target object, b) match the human spatial language syntax and human's language style and c) use the fewest number of words. However, to directly train the cost function by samples of φ, p and ε is a problem of great complexity. Here, we propose a multi-step process that splits the workflow into three steps:

(1) Model the Environment: the robot will build an environmental model which includes all the detected objects in its working environment until it finds the target object. All the objects in the environment are recorded. The information about an object is described by an Entity model that includes a category name, a coordinate vector, an orientation value and a unique ID of the object.

(2) Content Selection: Content selection is to decide what to say in a spatial description [2]. In our system, the content is represented by the RDT format grounding model presented in our previous work of spatial language grounding. In spatial language

grounding, the RDT model is a result of inference from natural language. Here, the RDT model is built from the environment model and is a reverse procedure of the inference to robot destination.

(3) Surface Realization: Surface Realization determines how to convert spatial information into natural language [2]. After getting the RDT grounding model, the system generates natural language using a model trained by a 149-sentence template corpus which has been extracted from the CSISL spatial language corpus introduced in [8]. The CSISL contains 1024 indoor spatial descriptions collected from human volunteers, and the 149-sentence template represents all of the different types of language structures that were captured by the 1024 participant descriptions. The surface realization model takes the RDT model as input to select words and phrases to construct a human-like natural language sentence.

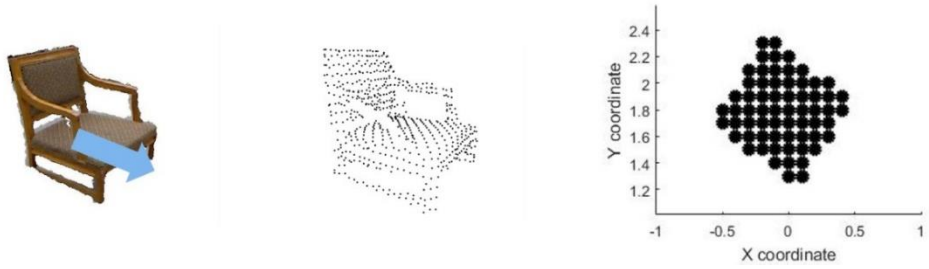


Figure 3. An example of using entity model to describe a chair. LEFT: the chair sample (The arrow illustrates its direction); MIDDLE: the 3-D point cloud of the chair; RIGHT: the 2-D point cluster ρ. The direction angle is $4\pi/7$rad (or 315°). The entity model $e=\{\text{“}chair\text{”},\rho,7/4\pi\}$.

C. Build Environment Model

The first step to generate a static spatial language description is to build an environment model which is created by the robot perception system. Our system uses a depth camera as the robot sensor. With prior internal knowledge about the objects in the working environment, the robot can recognize the objects and capture their geometric

features such as size, shape and orientation. The information of each object is integrated into a standard description named an Entity model. The Entity model is used to represent semantic objects handled in human spatial language. An Entity has: (1): an ID; (2): a name; (3): a coordinate vector; and (4): an orientation. The ID is the unique identification of an object in a robot task. The ID number of an entity is given by the sequence of detection. The name is the category of the object. The coordinate vector is a 2D point cloud representing the object's projection on the floor. To reduce the computation and noise we down-sample the raw point cloud to the positions of cells in a grid map. The orientation of an entity is defined as the direction value of its functional front side in the ego-centric reference, e.g., a chair has its functional front as the direction that a person faces when sitting on it. The example of a chair entity is shown in Figure 3.

During a language generation task, the robot will keep building the environment model when seeking the target object in the working space until it finds the target. Thus it can build the environment model as a set of N entities $\varepsilon=\{e_1,...,e_N\}$ in the working environment. Figure 4(a) shows the scene for an object seeking task and Figure 4(b) the environment built from it.

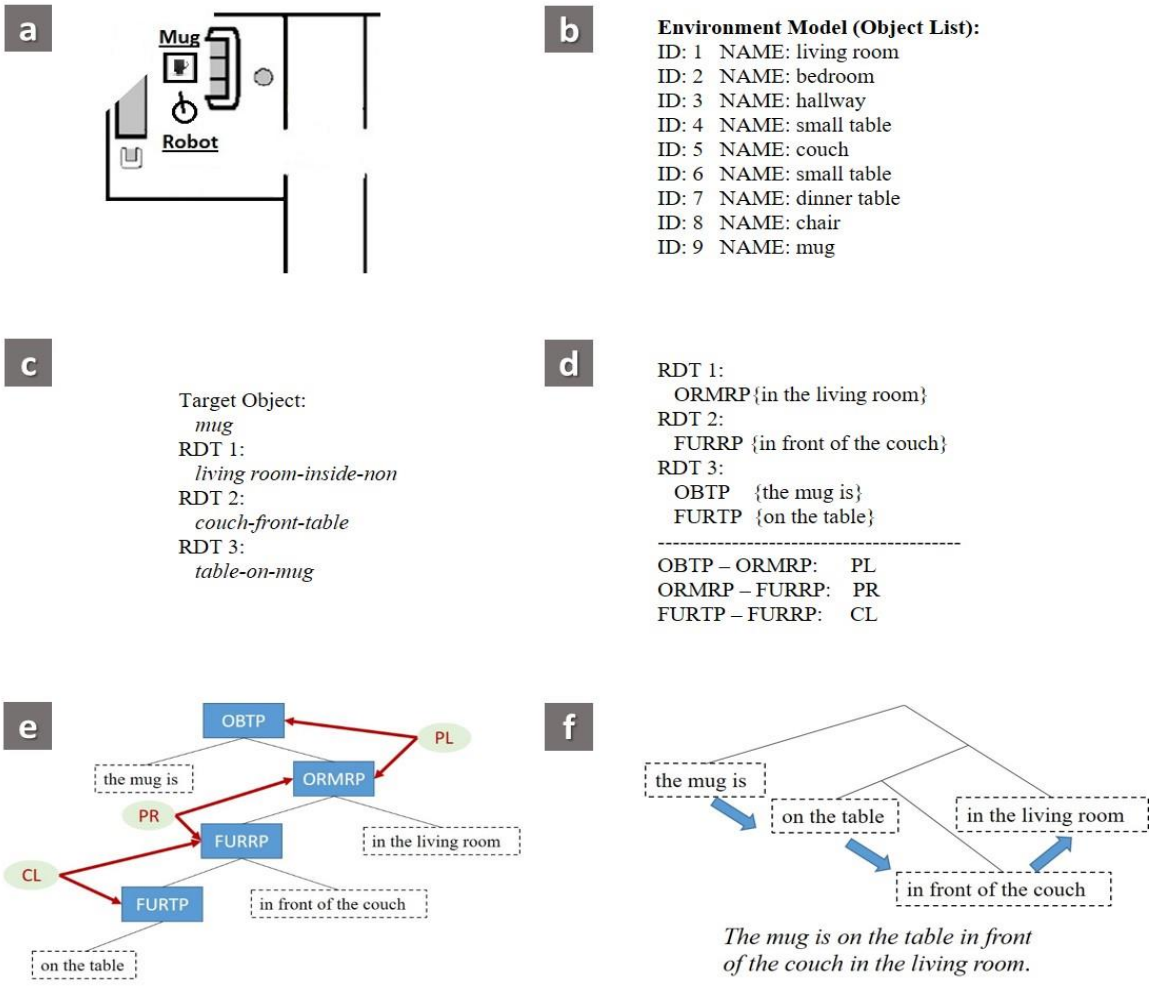


Figure 4. The procedure of natural spatial language generation. (a) The scenario when the robot detects the target object – mug. (b) The environment model. (c) The result of content selection. (d) The chunks used to infer the RDT nodes and their relations. (e) The tree structure built from the chunks and their relations. (f) The natural language description generated, the result of surface realization.

D. Content Selection

Next, the robot generates an RDT grounding model with several RDT nodes from the environment model. The entities list ε generated from the last step is used to build a spatial relation list $\Gamma(\varepsilon)=\{\gamma_1,\ldots,\gamma_M\}$. The list Γ includes M combinations between any two entities. For each combination, we use $\gamma_m=\{F_{direction}(e_a,e_b),F_{distance}(e_a,e_b)\}$ to represent two histogram vectors of direction and distance as the features of a spatial relationship. The spatial relation

list $\Gamma(\varepsilon)$ is also called the world state (WS), which describes the spatial relations in the environment. The WS is then used to calculate the probability $P(y|\Gamma)$ of each possible RDT node y which will be used later in the objective function. In the spatial language grounding system, $P(y|\Gamma)$ is used to infer the destination that the robot should move to the RDT node y. Here the robot is considered as an entity e_{robot} which is equivalent to other object entities in Γ. Since the positions of the other entities are fixed, the inference to the destination is to adjust the robot to a pose where the e_{robot} for the Γ can maximize the probability $P(y|\Gamma)$. In the language generation system, e_{robot} is set by the pose where the robot finds the target and stops. The WS Γ is then built by e_{robot} and other entities detected in the environment.

To seek the best solution over all RDT nodes, an objective function is proposed. Let $\{y_1,...,y_K\}$ denote K RDT nodes that can be extracted from the environment (K is smaller than the number of all the possible RDT types). The decision on whether to select an RDT node is represented by a binary weight value w_k. The w_k is 1 when the RDT node y_k is selected to generate the spatial language description and is 0 if not selected. A number v_{k1k2} is a value between 0 and 1 which is the conditional probability $P(w_{k1}|w_{k2})$ for the selection of the two RDT nodes y_{k1} and y_{k2}. This value is learned from the RDT nodes extracted from the 149-sentence template corpus. Since the Γ is fixed in this step, we let $P_{yk}=P_\Gamma(y_k)$ which is the probability of y_k in the environment. Then we can compose the following objective function for the combination of all the K RDT nodes which is:

$$\boldsymbol{O(W)} = \frac{\sum_{k=1}^{K} \boldsymbol{w_k P_{y_k}}}{\sum_{k=1}^{K} \boldsymbol{w_k}} + \sum_{\{k1,k2\}\in KK} \boldsymbol{v_{k1k2} w_{y_{k1}} w_{y_{k2}} P_{y_{k1}} P_{y_{k2}}} - \boldsymbol{\alpha} \frac{\sum_{k=1}^{K} \boldsymbol{w_k}}{\boldsymbol{K}} \quad \textbf{(2)}$$

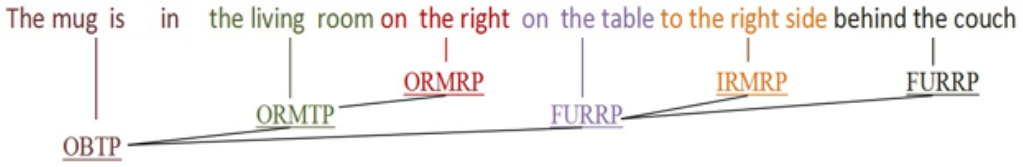
The $W=[w_1,...,w_K]$ is a vector which includes all the w_k values. $KK=\{(1,2),(1,3),(2,3),...,(K\text{-}1,K)\}$ denotes a set of combinations of any two different numbers in vector $[1,...,K]$. The three parts in $O(W)$ represent different restrictions on the

content to select. The first part encourages high probability groundings and the second part encourages the appearance of two related groundings that work together in the spatial description. The last part is used to get the shortest description. The constants $\alpha>0$ is adjusted by the training content data and we have α=0.1 in our system. Here W is the only variable to be sought in the objective function. To get the best RDT model, we will infer a solution W' to maximize the objective function $O(W)$ which is:

$$W' = \boldsymbol{argmax}_{W} \boldsymbol{O}(\boldsymbol{W}) \qquad \textbf{(3)}$$

The pose of human addressee is another restriction on the content to select. For example, when the robot and the person are in the same room, there is no need to present the information of room in the content. This restriction will work as a filter to remove some content.

Figure 4(c) shows the result of content selection from the environment model in Figure 4(b).



Outside Room Target Phrase	ORMTP
Outside Room Reference Phrase	ORMRP
Object Target Phrase	OBTP
Furniture Target Phrase	FURTP
Furniture Reference Phrase	FURRP
Inside Room Reference Phrase	IRMRP

Figure 5. A chunking tree structure of a spatial description and the explanation of the chunk types.

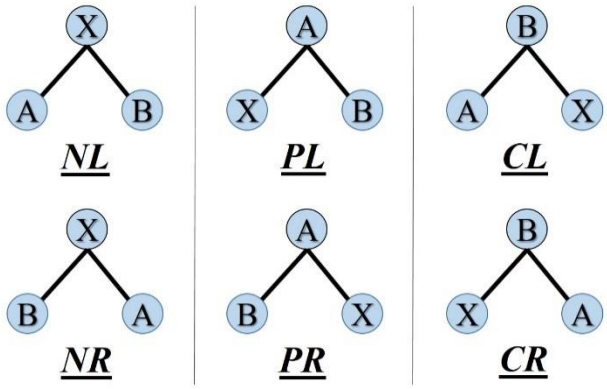


Figure 6. The six examples of the six possible relations between two chunks. For an instance, *NL*, the corresponding example demonstrates that how *A* is to be the *neighbor-left* to *B*.

E. Surface Realization

After inferring the best RDT model, the last step is the transition from the RDT nodes to the natural language description presenting the location of the target object. Considering the diversity and uncertainty of human-like spatial language, it is difficult to use a fixed prototype framework on language generation. Inspired by our previous work in [9], we consider the output natural language description as a tree structure constructed by several clauses. An example of a tree-structured description is shown in Figure 5, which shows a language model grouping words into chunks (word phrases). Each chunk $c=\{\tau,\eta\}$ consists of a clause of text τ and a chunk type η. The chunk types and explanations are also shown in Figure 5. Thus the surface realization is to construct a tree structure with all the chunks placed in the best places. The tree structure is inferred by a probabilistic model counting the text and the relations between chunks. The potential relations that chunk *A* can have to chunk *B* include six possibilities: *neighbor-left(NL), neighbor-right(NR), parent-left(PL), parent-right(PR), child-left(CL), child-right(CR)* (Shown in Figure 6). Assuming we have already inferred the best grounding model, which includes an RDT chain $y=\{y_1,...y_J\}$ including *J (J≤K, K* is the number of possible RDT node before content selection*)* RDT

nodes. Let let $\upsilon_{\eta A \eta B} \in \{NL,NR,PL,PR,CL,CR\}$ denotes the relation between two chunks with the names ηA and ηB where $\upsilon_{\eta A \eta B} \in \{NL,NR,PL,PR,CL,CR\}$. $\Upsilon = \{\upsilon_{\eta 1 \eta 2},\ldots,\upsilon_{\eta J\text{-}1 \eta J}\}$ is the set of all the relations. Then the language generation work is to determine the Υ which can maximize the probability $P(\Upsilon)$ to generate a tree structure, which can be written as:

$$\boldsymbol{P}(\Upsilon) = \boldsymbol{P}\left(\{(\boldsymbol{\tau_1}, \boldsymbol{\eta_1}), \boldsymbol{y_1}\}, \ldots \{(\boldsymbol{\tau_J}, \boldsymbol{\eta_J}), \boldsymbol{y_J}\}, \Upsilon\right) \propto \prod_{j=1}^{J} \boldsymbol{P}(\boldsymbol{\tau_j}, \boldsymbol{\eta_j} | \boldsymbol{y_j}) \prod_{j1=1}^{J} \prod_{j2=1}^{J} \boldsymbol{P}\left(\boldsymbol{\upsilon}_{\boldsymbol{\eta_{j1}} \boldsymbol{\eta_{j2}}}, | \boldsymbol{\eta_{j1}} \boldsymbol{\eta_{j2}}, \boldsymbol{j1 \neq j2}\right) \quad (1)$$

The conditional distribution can be trained by the 149 template descriptions that were derived directly from CSISL corpus collected from older adults. Figure 4(d) shows the chunks of the RDT nodes extracted in content selection and best matched relations of the chunks. Assume we extract P relations $Y=\{\upsilon_1,\ldots,\upsilon_P\}$ between the chunks in this step.

Algorithm 1

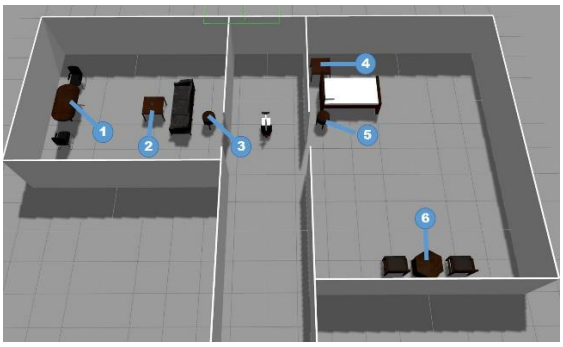
init: $A=\{c_1,\ldots,c_J\}$, $B=\{\upsilon_1,\ldots,\upsilon_P\}$, $t=1$, $TREE.ROOT=c_{obj}$
while $t<T$ **and** isempty(A) **is false**:
  for each υ in B:
    c_x,c_y=get_two_involved_chunks(υ)
    if ifintree(c_x) **xor** ifintree(c_y) **is true**:
      move $c_{notintree}$ to *TREE* by υ
      remove $c_{notintree}$ in A
      remove υ_p in B
    endif
  endfor
endwhile

After obtaining the chunks and their relations, the system then uses them to construct the tree structure. We use the chunk of the target object as the root of the tree. Two pools are created. Pool *A* contains the chunks not assigned to the tree and pool *B* contains the relations. An iterative algorithm is run on pool *B*, which places the chunks in pool *A* to the tree by the relations it involves in pool *B*. Then it removes the chunks from pool *A* and removes the relation in pool *B* (Algorithm 1). The iteration ends when pool *A* is empty or the iteration limit is reached. Figure 4(e) shows the tree structure generated from the chunks

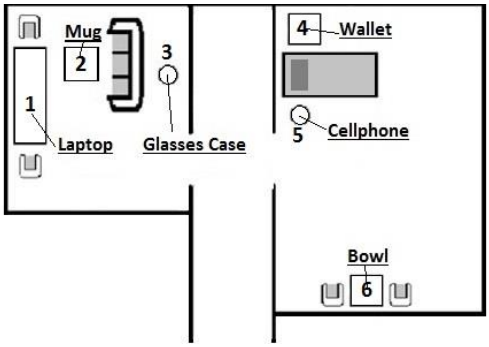
and relations presented in Figure 4(d). After building the tree, the system will generate the natural spatial language description using the in-order traversal of the tree [9].

The result (Figure 4(f)) is determined not only based on the words and tag of each grounding unit but also on their relations of nesting and ordering. This enables the system to mimic a human-like style in spatial language descriptions.

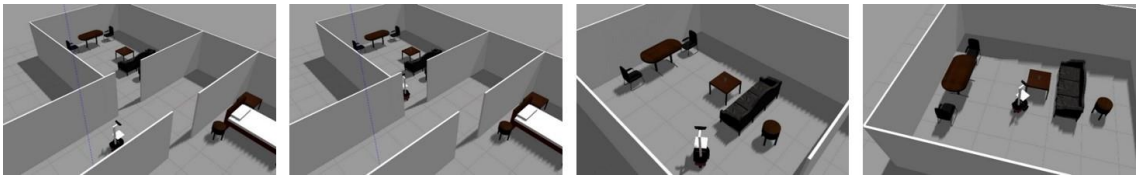
III. EXPERIMENT



(a)



(b)



(c)

Figure 7. (a) The 3D simulation scene built in Gazebo; The numbers label the furniture items where the target objects are placed on during the experiment. (b) The 2D floor plan of the scene for the experiment. (c) The object seeking procedure (from LEFT to RIGHT).

To evaluate the system, an experiment will be performed first in a simulated indoor environment which includes a bedroom, a living room and a hallway between them (Figure 7(a)). Both rooms have relevant furniture pieces. This setting has been used in our previous work on spatial language grounding and matches our physical lab space [6]. The simulation environment is built using Gazebo3D platform [11]. The perception data and the control

function of the robot were programmed the same as the version working in the physical environment so that system can also be migrated to the real world environment.

TABLE I The results of the early-stage experiment which includes six language generation tasks.

#	Object Target	RDT nodes	Natural Language Description
1	Laptop	*living room-inside-non* *table-beside-chair* *table-on-laptop*	*The laptop is on the table in the living room beside chairs.*
2	Mug	*living room-inside-non* *couch-front-table* *table-on-mug*	*There is the mug in the living room on the table in front of the couch.*
3	Glasses Case	*living room-inside-non* *couch-behind-table* *table-on-glasses case*	*The glasses case is in the living room on the table to the back of the couch.*
4	Wallet	*bedroom-inside-non* *bed-left-table* *table-on-wallet*	*The wallet is in the bedroom on the table to the left of the bed.*
5	Cellphone	*bedroom-inside-non* *bed-right-table* *table-on-cellphone*	*The cellphone is on the table in the bedroom to the right of the bed.*
6	Bowl	*bedroom-inside-non* *chair-beside-table* *room-right-non* *table-on-bowl*	*The bowl is on the table in the bedroom beside chairs to the far right wall.*

In a language generation test, the robot is initially positioned in the middle of the hallway and then starts to search for a target object after it receives the object name from the human user. It will keep on roaming in the working environment and builds the environment model until it finds the target. The target object can be placed in one of six different locations (Figure 7(b)). For each location, a static natural spatial language description will be generated. Here we list the results of an early-stage experiment in TABLE I which includes six descriptions generated by the robot. Although there are several metrics to score the performance of language generation, e.g., F-1 [12] and BLEU [12], which compare the similarity between the results and the ground truth, the best

approach to assess a language generation result is to have it scored by a human. To give a more reliable assessment to our system, we will employ volunteer test subjects to score the spatial descriptions that are generated by the robot.

IV. CONCLUSION

The development of this blueprint was an effort to achieve a natural spatial language generation system. Our preliminary work addresses some of the challenges. The results of the early experiments confirm a decision to not use language templates but rather to use a human spatial language corpus to program a language generator.

This system is trained by the 149-sentence template corpus and tested by six cases in the same scene where the corpus was collected. There are two limitations of the current experiment. First, the number of test cases is too small. Additionally, since the test scene is the same as the scene used to train the language model, it is not enough to validate the system's suitability to other environments. In the future, the number of the scenes for testing will be increased and the furniture placement will be alternated. Even the results present accurate and human understandable descriptions, the language has a lack of variety. We will also compare our approach with other machine learning methods like inverse reinforcement learning and recurrent neural network. To improve this aspect, we will also test additional features and train the system by other corpuses.

Acknowledgment

This work was funded by the NSF grant IIS-1017097.

References

[1]. Skubic, Marjorie, Zhiyu Huo, Tatiana Alexenko, Laura Carlson, and Jason Miller. "Testing an assistive fetch robot with spatial language from older and younger adults." In *RO-MAN, 2013 IEEE*, pp. 697-702. IEEE, 2013.

[2]. Reiter, Ehud, Robert Dale, and Zhiwei Feng. *Building natural language generation systems*. Vol. 33. Cambridge: Cambridge university press, 2000.

[3]. Chen, David L., and Raymond J. Mooney. "Learning to sportscast: a test of grounded language acquisition."*Proceedings of the 25th international conference on Machine learning*. ACM, 2008.

[4]. Tse, Rina, and Mark Campbell. "Human-robot information sharing with structured language generation from probabilistic beliefs." Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on. IEEE, 2015.

[5]. Angeli, Gabor, Percy Liang, and Dan Klein. "A simple domain-independent probabilistic approach to generation."*Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2010.

[6]. Huo, Zhiyu, Tatiana Alexenko, and Marjorie Skubic. "Using spatial language to drive a robot for an indoor environment fetch task." In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 1361-1366. IEEE, 2014.

[7]. Brill, Eric. "A simple rule-based part of speech tagger." In *Proceedings of the workshop on Speech and Natural Language*, pp. 112-116. Association for Computational Linguistics, 1992.

[8]. Carlson, Laura, Marjorie Skubic, Jared Miller, Zhiyu Huo, and Tatiana Alexenko. "Strategies for Human-Driven Robot Comprehension of Spatial Descriptions by Older Adults in a Robot Fetch Task." *Topics in cognitive science* 6, no. 3 (2014): 513-533.

[9]. Alexenko, Tatiana, Marjorie Skubic, and Zhiyu Huo. "Spatial Language Processing for Assistive Robots with" Deep" Chunking and Semantic Grammars." *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.

[10]. Schoenmakers, Berry. "Inorder traversal of a binary heap and its inversion in optimal time and space." *Mathematics of Program Construction*. Springer Berlin Heidelberg, 1992.

[11]. Koenig, N., and A. Howard. "Gazebo-3D multiple robot simulator with dynamics (2003)." URL: http://gazebosim. org 3 (2013).

[12]. Powers, David Martin. "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation." (2011).

[13]. Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation." In Proceedings of the 40th annual meeting on association for computational linguistics, pp. 311-318. Association for Computational Linguistics, 2002.

6.3 New Experiments and Results

The workshop paper in the last section only demonstrated six cases of the spatial language generation system, which was not enough for validation. To further evaluate the indoor spatial language generation system, we added more scenes and tasks.

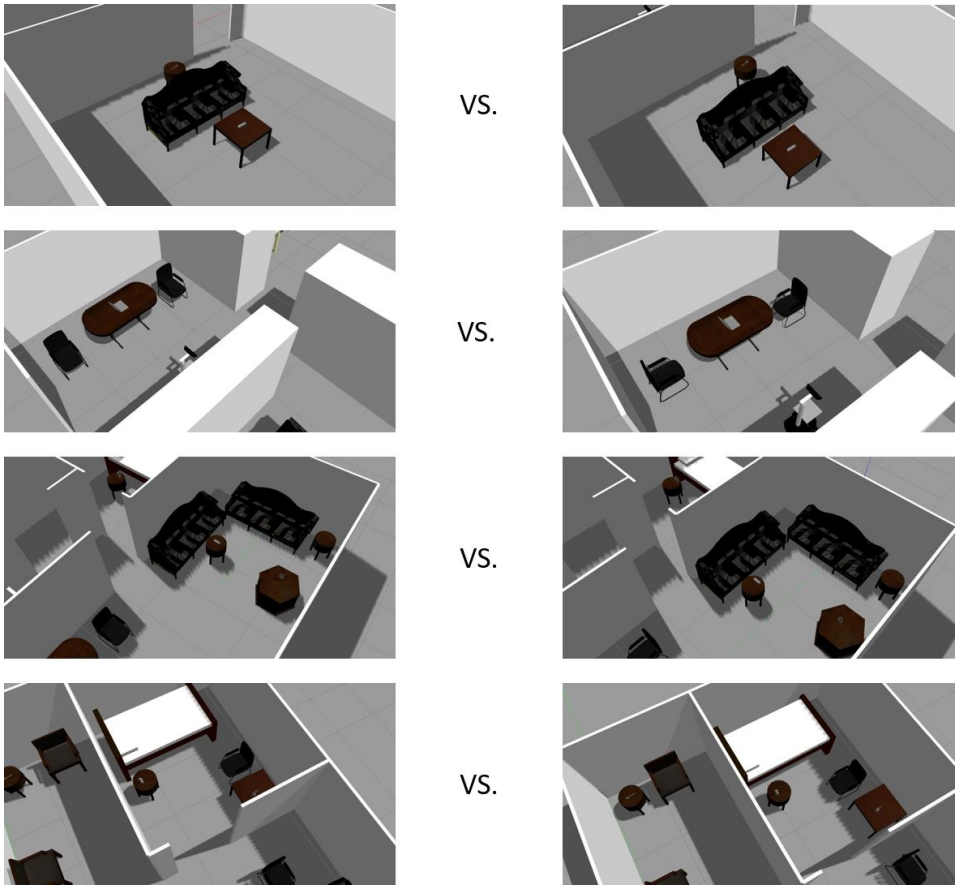


Figure 6.1 Alternate positions and orientations of the furniture items in the simulated worlds

The new experiment included 48 language generation tasks. We ran tests on the robotic simulator introduced in Section 3.4 which expanded the simulation environment described in Section 6.2 of the reprinted publication ([16, 17]). The spatial descriptions of the 24 target objects placed in different locations in the four simulation worlds were generated. We also slightly changed the positions of the furniture items which placed the target objects and the neighbor furniture pieces, but kept the same topology links and the spatial relationship to build another 24 alternative scenes to add another 24 language generation tasks into the experiments. Figure 6.1 shows some the new furniture item positions. The 48 spatial descriptions given by our language generator are shown in Table 6.1.

Table 6-1a The language generation results.
The spatial descriptions generated from the original scene.

#	World	Target Object	Spatial Description	If the SR Match
1	Apartment	Fork	*The fork is in the living room on the table to the back of the couch.*	*Y*
2		Glasses Case	*There is the glasses case in the living room on the table in front of the couch.*	*Y*
3		Laptop	*The laptop is on the table in the living room beside chairs.*	*Y*
4		Statue	*The statue is on the table in the bedroom to the right of the bed.*	*Y*
5		Monitor	*The monitor is on the table in the bedroom to the left of the bed.*	*Y*
6		Mug	*The mug is on the table in the bedroom beside chairs to the far right wall.*	*Y*
7	Hk-studio	Fork	*The fork is in the studio room on the table beside chairs to the far right wall.*	*Y*
8		Glasses Case	*There is the glasses case in the office to the far right wall.*	*Y*
9		Laptop	*The laptop is on the table beside chairs in the meeting room.*	*Y*
10		Statue	*The statue is in the studio room on the table beside chairs to the far right wall.*	*Y*
11		Monitor	*The monitor is in the studio against the left wall.*	*Y*
12		Mug	*Move halfway in the mug is in the office on the table in front of the couch.*	*Y*
13	One-bedroom-house	Fork	*The fork is in the meeting room on the table beside chair.*	*Y*
14		Glasses Case	*There is the glasses case on the table in the living room beside chairs in front of the couch.*	*Y*
15		Laptop	*The laptop is on the table beside chairs in the office.*	*Y*
16		Statue	*The statue is on the table in the bedroom to right of the bed.*	*Y*
17		Monitor	*The monitor is in the bedroom to the far right wall against the left wall against the left wall on the table.*	*N*
18		Mug	*The mug is on the table in the living room to the left of the couch.*	*Y*
19	Two-bedrooms-house	Fork	*The fork is to the far right wall on the table beside chairs in the office.*	*N*
20		Glasses Case	*There is the glasses case on the table in the living room beside chairs*	*Y*
21		Laptop	*Move halfway in the laptop is in the meeting room.*	*N*
22		Statue	*The statue is on the table in the bedroom to the right of the bed.*	*Y*
23		Monitor	*Move halfway in the monitor is on the table beside chairs in the office.*	*Y*
24		Mug	*The mug is at the back in the living room to the far right wall.*	*Y*

Table 6.1b The spatial descriptions generated by the modified scene

#	World	Target Object	Spatial Description	If the SR Match
1	Apartment	Fork	*The fork is in the living room on the table to the back of the couch.*	*Y*
2		Glasses Case	*There is the glasses case in the living room on the table in front of the couch.*	*Y*
3		Laptop	*The laptop is on the table in the living room beside chairs.*	*Y*
4		Statue	*The statue is on the table in the bedroom to the right of the bed.*	*Y*
5		Monitor	*The monitor is on the table in the bedroom to the left of the bed.*	*Y*
6		Mug	*The mug is on the table in the bedroom beside chairs to the far right wall.*	*Y*
7	Hk-studio	Fork	*The fork is in the studio room on the table beside chairs to the far right wall.*	*Y*
8		Glasses Case	*There is the glasses case in the office to the far right wall.*	*Y*
9		Laptop	*The laptop is on the table beside chairs in the meeting room.*	*Y*
10		Statue	*The statue is in the studio room on the table beside chairs to the far right wall.*	*Y*
11		Monitor	*The monitor is in the studio against the left wall.*	*Y*
12		Mug	*Move halfway in the mug is in the office on the table in front of the couch.*	*Y*
13	One-bedroom-house	Fork	*The fork is in the meeting room on the table beside chair.*	*Y*
14		Glasses Case	*There is the glasses case on the table in the living room beside chairs in front of the couch.*	*Y*
15		Laptop	*The laptop is on the table beside chairs in the office.*	*Y*
16		Statue	*The statue is on the table in the bedroom to right of the bed.*	*Y*
17		Monitor	*The monitor is in the bedroom to the far right wall against the left wall against the left wall on the table.*	*N*
18		Mug	*The mug is on the table in the living room to the left of the couch.*	*Y*
19	Two-bedrooms-house	Fork	*The fork is to the far right wall on the table in the office.*	*Y*
20		Glasses Case	*There is the glasses case on the table in the living room beside chairs*	*Y*
21		Laptop	*Move halfway in the laptop is in the meeting room.*	*N*
22		Statue	*The statue is on the table in the bedroom to the right of the bed.*	*Y*
23		Monitor	*Move halfway in the monitor is on the table beside chairs in the office.*	*Y*
24		Mug	*The mug is on the table in the living room to the far right wall.*	*Y*

6.4 Conclusions

Our natural spatial language generation system can compose the answer of a query on the position of a daily object using the information of the observed scene in an indoor environment. The system accomplishes the spatial language generation task from scene to text using a cascaded structure. Compare with other language generation works [50] [51] [60], our system is more specific on the indoor spatial description task but can compose long and complicated sentences. The system first symbolizes and quantifies the scene to an environment model. Then it scores and predicts the possible groundings. A language model then examines the groundings to eliminate any redundant or conflicted information. Finally, a parsing tree was built and the objective language was generated by in-order traverse through the tree. We compared our results with the descriptions of the same scenes generated by humans which are considered as ground truth. The machine generated language was scored by the number of spatial relationships found in the human edited language. An arbitrary spatial description is then successfully generated if it captures all the spatial relationships in the human language. All 48 comparisons are shown in Table 6-1. The overall successful rate is 89.6% (43/48).

The results demonstrate that our system has the capacity to generate precise and human-like spatial language in a complex environment. This enable the robots' ability to navigate in a human spatial language domain. Compared with dynamical spatial commands, the static spatial descriptions are more understandable for a human since they better match the human user's habits in spatial language. The use of references and spatial relationships are also helpful in reducing the ambiguity sometimes found in the spatial language.

We observed some failures and inappropriate cases in our results. The spatial description phrase *move halfway in* appears too often in the language which created redundancy and ambiguity in the language. The reason is that this spatial relationship can always be matched when the location of the target object is not against the wall or corner, which means it will survive in the content selection step with a high score. In addition, the language model refining step did not detect and remove this reductant information. Adding another refining stage from a generalized human language model may be needed to solve the problem.

Chapter 7. Conclusion and Perspective

7.1 Contribution of the dissertation

The dissertation presents a framework to allow robots to interact with human in the indoor environment. The framework can process the natural spatial language data and the perception data to determine the action of the robot. It can also generate the natural spatial language to the human users. A point-cloud based furniture robot perception system was developed to map the robot working environment. To evaluate the performance of the system in various environment, we built a 3D robot simulator which contain four different worlds.

This research presented herein contributes to the advancement of robotic technology. The advancements are based on the development of an autonomous robotic system and a spatial language interaction system. Future plans involve researching and developing a robot that understands and uses human-like spatial language.

The autonomous robot system developed a trajectory planning and tracking algorithm for differential drive robots, which are programmed to enable P3DX robots to move in an indoor environment. The robot perception experiment utilized a Microsoft Kinect camera as its main sensor to build a model for small working environments with an emphasis on apartment complexes devoted to providing assisted care for the elderly. Leveraging the work of my M.S. thesis, a new furniture object recognition system based on point cloud data was developed, evaluated and utilized in a new robot system. The new perception abandons instance level recognition and gets a better performance on category level recognition of furniture. It also decreased errors in furniture orientation estimation. The

perception system helps the robot to build its working environment model, which contains the category, position and orientation of the objects (emphasized on the furniture items) in the robot working space.

The development of a spatial language interaction system started with collecting a human spatial language corpus. This part was conducted by Dr. Carlson's group at Notre Dame. They collected thousands of utterances on spatial description from both elder adults and younger adults. This corpus was then reorganized into a template, which is a prototype of expression on spatial description. Alexenko developed a part-of speech (PoS) tag system for the template by using the natural language tool kit (NLTK). The system can tag the words in a spatial description and then chunk these words to form clauses with the tags of spatial information and generate a tree structure. The PoS system was improved and extended to a larger size template so that it can adjust to a more complicated spatial language input. A novel spatial language grounding model, which can interpret the spatial description to robot actions was designed and evaluated on both a simulated and physical platform.

To solve the disadvantages of hard-coded robot behavior models, we developed an algorithm to program the robot behavior model by demonstration. The demonstration was the recording of actions given by non-expert users to the robot for following spatial language commands. The algorithm parameterized demonstration records to world state features and used them to build moving target estimators for each kind of RDT-form grounding. This algorithm enabled the robot to learn its behavior model by data rather than manually editing. It not only helped to reduce the labor of programing but also clarified the understanding of spatial concepts.

The completed capability of using spatial language not only includes understanding spatial language but also provides the means to use it. This phase of the research is to convert the robot from receiver to sender of spatial commands or information. A spatial language generation system was designed to enable a robot to give spatial descriptions to humans. The system learns how to build a spatial description from a spatial language corpus. The scenario of the task requires the robot to leave its human user after being asked to search for a target object. It then builds the environment model during the searching. After it finds the target object, it will generate the most adequate spatial description to inform the human user of the position of the target object.

Compared with other systems proposed by [50-52, 60], this research proposes a general framework for both static and dynamic spatial language and a completed solution of human-robot spatial language interaction. The system also provides the method and the interface to model human demonstrations on spatial concepts so that the robot can quickly learn new spatial language through very few demonstration examples. During the evaluation of the system, we did not ignore the challenges and counted all the uncertainties in the perception, language and robot state. The results of the end-to-end experiments validates the feasibility of our system and provided a perspective for further development.

References

1. Hayward, W.G. and M.J. Tarr, *Spatial language and spatial representation.* Cognition, 1995. **55**(1): p. 39-84.
2. Talmy, L., *How language structures space*. 1983: Springer.
3. Landau, B. and R. Jackendoff, *Whence and whither in spatial language and spatial cognition?* Behavioral and brain sciences, 1993. **16**(02): p. 255-265.
4. Levinson, S.C., *Frames of reference and Molyneux's question: Crosslinguistic evidence.* Language and space, 1996: p. 109-169.
5. Levelt, W.J., *Perspective taking and ellipsis in spatial descriptions.* Language and space, 1996: p. 77-107.
6. Herskovits, A., *Language and spatial cognition: An interdisciplinary study of the prepositions in english.* Cambridge: Cambridge University Press, 1986.
7. Schober, M.F., *Spatial perspective-taking in conversation.* Cognition, 1993. **47**(1): p. 1-24.
8. Carlson, L., et al., *Strategies for Human‐Driven Robot Comprehension of Spatial Descriptions by Older Adults in a Robot Fetch Task.* Topics in cognitive science, 2014. **6**(3): p. 513-533.
9. Carlson, L.A., et al. *Assessing the effectiveness of older adults' spatial descriptions in a fetch task*. in *Paper submitted to the 35th Annual Conference of the Cognitive Science Society. Berlin, Germany*. 2013.
10. Skubic, M., et al. *Human-Driven Spatial Language for Human-Robot Interaction*. in *Human-Robot Interaction in Elder Care*. 2011.
11. Skubic, M., et al. *Spatial language experiments for a robot fetch task*. in *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*. 2012. IEEE.
12. Huo, Z., *Robot methods for human-robot spatial language interaction*. 2013, University of Missouri--Columbia.
13. Huo, Z., T. Alexenko, and M. Skubic. *Using spatial language to drive a robot for an indoor environment fetch task*. in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. 2014. IEEE.
14. Skubic, M., et al. *Testing an assistive fetch robot with spatial language from older and younger adults*. in *RO-MAN, 2013 IEEE*. 2013. IEEE.
15. Alexenko, T., M. Skubic, and Z. Huo. *Spatial Language Processing for Assistive Robots with" Deep" Chunking and Semantic Grammars*. in *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.
16. Huo, Z. and M. Skubic. *Natural spatial description generation for human-robot interaction in indoor environments*. in *Smart Computing (SMARTCOMP), 2016 IEEE International Conference on*. 2016. IEEE.
17. Huo, Z. and M. Skubic, *Natural Spatial Language Generation for Indoor Robot.* 2016.
18. Davis, R.L., B.A. Therrien, and B.T. West, *Working memory, cues, and wayfinding in older women.* Journal of Applied Gerontology, 2009. **28**(6): p. 743-767.

19. Lai, K., et al. *Sparse distance learning for object recognition combining rgb and depth information*. in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. 2011. IEEE.
20. Blum, M., et al. *A learned feature descriptor for object recognition in rgb-d data*. in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. 2012. IEEE.
21. Aldoma, A., et al., *Point cloud library.* IEEE Robotics & Automation Magazine, 2012. **1070**(9932/12).
22. Tang, S., et al., *Histogram of oriented normal vectors for object recognition with a depth sensor*, in *Computer Vision–ACCV 2012*. 2012, Springer. p. 525-538.
23. Rusu, R.B., *Semantic 3d object maps for everyday manipulation in human living environments.* KI-Künstliche Intelligenz, 2010. **24**(4): p. 345-348.
24. Howard, A., *Gazebo 3D Robot Simulator.* Internet: playerstage. sourceforge. net/gazebo/gazebo. html, 2006.
25. Konolige, K. and P. Mihelich, *Technical description of Kinect calibration.* Tech. Rep., Willow Garage, 2011.
26. Choi, S., T. Kim, and W. Yu, *Performance evaluation of RANSAC family*. Journal of Computer Vision, 1997. **24**(3): p. 271-300.
27. Dillencourt, M.B., H. Samet, and M. Tamminen, *A general approach to connected-component labeling for arbitrary image representations.* Journal of the ACM (JACM), 1992. **39**(2): p. 253-280.
28. Aldoma, A., et al., *Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation.* IEEE Robotics & Automation Magazine, 2012. **19**(3): p. 80-91.
29. Szegedy, C., A. Toshev, and D. Erhan. *Deep neural networks for object detection*. in *Advances in Neural Information Processing Systems*. 2013.
30. Hinton, G.E. and R.R. Salakhutdinov, *Reducing the dimensionality of data with neural networks.* science, 2006. **313**(5786): p. 504-507.
31. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
32. Kavukcuoglu, K., et al. *Learning convolutional feature hierarchies for visual recognition*. in *Advances in neural information processing systems*. 2010.
33. Graves, A., et al., *A novel connectionist system for unconstrained handwriting recognition.* IEEE transactions on pattern analysis and machine intelligence, 2009. **31**(5): p. 855-868.
34. LeCun, Y., Y. Bengio, and G. Hinton, *Deep learning.* Nature, 2015. **521**(7553): p. 436-444.
35. Henry, P., et al. *Patch volumes: Segmentation-based consistent mapping with RGB-D cameras*. in *3D Vision-3DV 2013, 2013 International Conference on*. 2013. IEEE.
36. Skubic, M., et al. *Investigating spatial language for robot fetch commands*. in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*. 2012.
37. Marcus, M.P., M.A. Marcinkiewicz, and B. Santorini, *Building a large annotated corpus of English: The Penn Treebank.* Computational linguistics, 1993. **19**(2): p. 313-330.

38. Bird, S. *NLTK: the natural language toolkit*. in *Proceedings of the COLING/ACL on Interactive presentation sessions*. 2006. Association for Computational Linguistics.
39. Brill, E., *Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging.* Computational linguistics, 1995. **21**(4): p. 543-565.
40. Bird, S., E. Klein, and E. Loper, *Natural language processing with Python*. 2009: " O'Reilly Media, Inc.".
41. Collobert, R., et al., *Natural language processing (almost) from scratch.* The Journal of Machine Learning Research, 2011. **12**: p. 2493-2537.
42. Graves, A., A.-r. Mohamed, and G. Hinton. *Speech recognition with deep recurrent neural networks*. in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. 2013. IEEE.
43. Deng, L., G. Hinton, and B. Kingsbury. *New types of deep neural network learning for speech recognition and related applications: An overview*. in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. 2013. IEEE.
44. Hochreiter, S. and J. Schmidhuber, *Long short-term memory.* Neural computation, 1997. **9**(8): p. 1735-1780.
45. Gers, F.A., J. Schmidhuber, and F. Cummins, *Learning to forget: Continual prediction with LSTM.* Neural computation, 2000. **12**(10): p. 2451-2471.
46. Abadi, M., et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems.* arXiv preprint arXiv:1603.04467, 2016.
47. Billard, A., et al., *Robot programming by demonstration*, in *Springer handbook of robotics*. 2008, Springer. p. 1371-1394.
48. Lauria, S., et al., *Mobile robot programming using natural language.* Robotics and Autonomous Systems, 2002. **38**(3): p. 171-181.
49. Matuszek, C., et al. *Learning to parse natural language commands to a robot control system*. in *Experimental Robotics*. 2013. Springer.
50. Fasola, J. and M.J. Mataric. *Using semantic fields to model dynamic spatial relations in a robot architecture for natural language instruction of service robots*. in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. 2013. IEEE.
51. Kollar, T., et al. *Toward understanding natural language directions*. in *Human-Robot Interaction (HRI), 2010 5th ACM/IEEE International Conference on*. 2010. IEEE.
52. Tellex, S., et al., *Approaching the symbol grounding problem with probabilistic graphical models.* AI magazine, 2011. **32**(4): p. 64-76.
53. Tellex, S., et al. *Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation*. in *AAAI*. 2011.
54. Chronis, G. and M. Skubic. *Sketch-based navigation for mobile robots*. in *Fuzzy Systems, 2003. FUZZ'03. The 12th IEEE International Conference on*. 2003. IEEE.
55. Chronis, G. and M. Skubic. *Robot navigation using qualitative landmark states from sketched route maps*. in *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*. 2004. IEEE.

56. Skubic, M., et al., *Generating multi-level linguistic spatial descriptions from range sensor readings using the histogram of forces.* Autonomous Robots, 2003. **14**(1): p. 51-69.
57. Argall, B.D., et al., *A survey of robot learning from demonstration.* Robotics and autonomous systems, 2009. **57**(5): p. 469-483.
58. Schaal, S., *Is imitation learning the route to humanoid robots?* Trends in cognitive sciences, 1999. **3**(6): p. 233-242.
59. Friedrich, H., et al., *Robot programming by demonstration (RPD): Supporting the induction by human interaction.* Machine Learning, 1996. **23**(2-3): p. 163-189.
60. Duvallet, F., T. Kollar, and A. Stentz. *Imitation learning for natural language direction following through unknown environments*. in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. 2013. IEEE.
61. Matsakis, P. and L. Wendling, *A new way to represent the relative position between areal objects.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, 1999. **21**(7): p. 634-643.
62. Matsakis, P., et al., *Linguistic description of relative positions in images.* Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 2001. **31**(4): p. 573-588.
63. Quigley, M., et al. *ROS: an open-source Robot Operating System*. in *ICRA workshop on open source software*. 2009.
64. Gerkey, B., R.T. Vaughan, and A. Howard. *The player/stage project: Tools for multi-robot and distributed sensor systems*. in *Proceedings of the 11th international conference on advanced robotics*. 2003.

APPENDIX

Table A1 ***Raw result of the robot training assessment.***

	NSLD	RDT	Target Pose (x, y, rotation)	Final Pose (x, y, rotation)	If Capture Target
1	go/walk/move forward	Move-front-non	4.3, 2.7, 1.5	5, 2.5, 0	N/A
			2.7, 4.3, 91	3, 5, 90	
			0.4, -4.8, 223	0.24, -4.8, 225	
2	Turn left	Move-left-non	2, 2.5, 90	2, 2.5, 88	N/A
			3, 2, 180	3, 2, 178	
			2, -3, 315	2, -3.1, 316	
3	Turn right	Move-right-non	2, 2.5, 270	2, 2.5, 271	N/A
			3, 2, 0	3, 2, 1	
			2, -3, 135	2, -3.1, 133	
4	On the Table on the/your left	Robot-left-table	2, 2, 180	2.1, 2.1, 180	Y
			0.5, -5, 180	0.6, -5.3, 180	Y
			1, 5.5, 90	0.4, 5.6, 68	Y
5	On the table on the/your right	Robot-right-table	5.5, 5, 0	5.3, 5.3, 339	Y
			-1, -7, 135	-0.3, -7.1, 159	Y
			2, 7, 180	2.2, 6.1, 159	Y
6	On the table to the left of the bed	Bed-left-table	-1, 3, 270	-1.1, 2.8, 272	Y
			-1.5, 2.1, 50	-1.6, 2.1, 350	N
			-1, 3, 270	-1.1, 2.8, 271	Y
7	On the table to the right of the bed	Bed-right-table	1.5, 2.5, 225	-1.1, 2.8, 271	Y
			1, 3, 270	1, 2.9, 268	Y
			1.5, 2, 180	1.6, 2.2, 191	Y
8	On the Table in front of the couch	Couch-front-table	0, -4.5, 225	0, -5, 158	N
			0.5, -5, 180	0.1, -5.1, 159	Y
			0, -5.5, 135	0.1, -5.4, 156	Y
9	On the table to the left of the couch	Couch-left-table	-0.8, -3.5, 180	-0.8, -3.4, 188	Y
			-1, -3, 225	-0.9, -3.0, 224	Y
			-1.5, -3, 270	-1.5, -2.9, 271	Y
10	On the table to the right of the couch	Couch-right-table	-1, -6, 225	-0.6, -5.9, 223	Y
			-0.5, -6.5, 180	-0.5, -5.8, 223	Y
			-1, -7, 135	-0.6, -6, 223	Y
11	On the table Beside a chair	Chair-beside-table	0.5, -7.2, 0	0.3, -6.8, 358	Y
			-0.5, -3, 90	0.1, -3.1, 110	Y
			5.5, 5, 0	5, 4.8, 358	Y
12	"go to the center of the room"/"move about halfway in"	Room-center-non	2.5, 4.5, 90	3.4, 4.2, 68	N/A
			2.5, 4.5, 90	1.3, 4.7, 358	
			0.5, -4.5, 270	1, 4.2, 268	
13	Move to the front wall	Wall-front-non	-1, 5, 180	0, 5, 180	N/A
			5.5, 5, 0	5, 5, 0	
			0.5, -6.5, 270	0.5, -6, 270	

Table A2. ***Raw results of the simulated end-to-end assessment.***

The 79 new edited spatial commands or descriptions (labeled by "*world-name_target-object"*)

1–2	apartment_fork: the fork is in the living room on the right on the table directly to the right go to the living room and the fork will be on the table behind the couch
3–6	apartment_mug: go to the bedroom then move about halfway in and then turn right go forward and there is the mug go to the bedroom then move about halfway in and then turn right go forward to the table with the chairs and there is the mug the mug is in the bedroom on the table to the far right. the mug is in the bedroom to the left on the table by the chairs at the far right end.
7–9	apartment_statue: the statue is in the bedroom on the table to the right of the bed. go to the bedroom then turn left you will see the statue on the table. go to the bedroom and the statue will be on the nightstand ahead to your left.
10–13	apartment_monitor go to the bedroom on the left you will find the monitor on top of the nightstand to the left of the bed. the monitor is in the bedroom to the left on the left wall. the monitor is in the bedroom on the table to the left of the bed. go straight and turn left and go straight about halfway in and turn left then go forward until you hit the wall and then left again and you will find the monitor.
14–16	apartment_laptop: go forward and turn right and walk straight until you are at the table at the back and you will find the laptop go to the living room and go straight until you are at the wall and you will find the laptop on the table with the chairs the laptop is in the living room on the back table with the chairs
17–18	apartment_glassescase: the glasses case was in the living room on the table in front of the couch. go to the living room and go forward and turn right and you will find the glasses case on the table in front of the couch.
19–22	hkstudio_fork: go to the studio and the fork is on the table ahead to the left the fork is in the studio on the table ahead to your left the fork is in the studio room on the table on the left side wall the fork is in the studio on the table by the chair
23–25	hkstudio_mug: go to the office and the mug is on the table behind the couch go to the office and the mug is on the table in front of the couch go to the office and go about halfway in and the mug will be on the table
26–27	hkstudio_statue: go to the studio and the statue is on the table at the end of the room go to the studio and the statue is on the table by the chair at the end of the room

28–32	hkstudio_monitor: the monitor is in the studio on the table on the right. go to the studio then go forward then turn right and go forward and you will find the monitor on the table. go to the studio then go forward then turn right and go forward and you will find the monitor on the table with the chair. the monitor is in the studio on the table to the far right in the studio. go to the studio and the monitor is on the table with the chair to the right of the room
33–40	hkstudio_laptop: the laptop is on the table to the left. the laptop is on the table with the chairs. the laptop is on the table to the left by the chairs. turn left then you will find the laptop on the table. turn left then you will find the laptop on the table by the chairs. the laptop is on the table in the meeting room. the laptop is on the table on the left in the meeting room. the laptop is in the meeting room on the table on the left.
41–43	hkstudioglasses case: go forward and turn right and then go forward and go straight until you are at the back and you will find the glasses case. the glasses case is in the office on the table with the chair. go to the office and the glasses case is on the table by the chair.
44–47	onebedroom_fork: the fork is on the table in the meeting room the fork is in the meeting room on the table by the chair go to the meeting room the fork will be on the table by the chair go to the meeting room then the fork will be on the table
48–51	onebedroom_mug: turn right and then you will find the mug on the table the mug is on the table to the left of the couch the mug is on the table on the right turn right then go forward you will find the mug on the table
52–55	onebedroom_statue: the statue is in the bedroom on the nightstand to the right of the bed the statue is on the table to the right of the bed in the bedroom go to the bedroom then go forward you will find the statue on the table go to the bedroom then go about halfway in you will find the statue on the table to the right of the bed
56–59	onebedroom_monitor: the monitor is in the bedroom on the table with the chair go to the bedroom the turn right you will find the monitor on the right the monitor is in the bedroom on the table to the left of the chair go the bedroom then you will find the monitor on the table on the right
60–61	onebedroom_laptop: go to the office then you will find the laptop on the table by the chair the laptop is on the table in the office room

62–65	onebedroom_glassescase: the glasses case is on the table in front of the couch the glasses case is in the living room on the table in front of the couch go forward then turn right you will find the glasses case on the table go forward then turn right you will find the glasses case on the table in front of the couch
66–68	twobedrooms_fork: go to the office and then turn right and the fork is on the table go to the office and the fork will be on the right the fork is in the office room on the table to the right end of the room
69–70	twobedrooms_mug: the mug is in the living room on the table to the right of the room the mug is on the table in the living room
71–73	twobedrooms_statue: the statue is on the table in the bedroom go to the bedroom then you will find the statue on the table on the left. go to the bedroom and the statue is on the table to the right of the bed.
74–75	twobedrooms_monitor: the monitor is in the office room on the table by the chair go to the office and then you will find the monitor on the table
76–78	twobedrooms_laptop: go forward and then turn left and you will find the laptop on the table go forward and the laptop is on the table ahead to your left the laptop is on the table by the chairs go forward and the laptop is on the table by the chairs ahead to your left
79	twobedrooms_glassescase: the glasses case is in the living room on the table in front of the couch

Table A3. Raw results of the 79 rounds robot fetch tasks in the four simulated environments.

#	Name	Target Position (m, m)	Final Pose (m, m, degree)	If the target object captured
1	apartment_fork_0	4, 2	2.93, 1.89, 358	Y
2	apartment_fork_1		3.43, 1.59, 43	Y
3	apartment_laptop_0	4.5, 7		(pos)
4	apartment_laptop_1		3.97, 5.58, 76	Y
5	apartment_laptop_2			(pos)
6	apartment_monitor_0	6, -2	6.18, -3.52, 91	Y
7	apartment_monitor_1			(not visible)
8	apartment_monitor_2		6.09, -3.50, 90	Y
9	apartment_monitor_3			(not visible)
10	apartment_mug_0	-1, -5	-0.44, -4.60, 178	Y
11	apartment_mug_1		0.32, -4.9, 201	Y
12	apartment_mug_2			(pos)
13	apartment_mug_3			(not visible)
14	apartment_statue_0	4, -2	3.85, -3.12, 88	Y
15	apartment_statue_1		2.28, -1.95, 358	Y
16	apartment_statue_2		2.69, -2.07, 358	Y
17	apartment_glassescase_0	4.3, 4.1	3.25, 4.01, 1	Y
18	apartment_glassescase_1		3.47, 3.61, 44	Y
19	hkstudio_fork_0	-4, 0.5	-2.40, 0.54, 178	Y
20	hkstudio_fork_1		-2.44, 0.39, 178	Y
21	hkstudio_fork_2		-3.27, 1.56, 245	Y
22	hkstudio_fork_3		-2.17, 0.71, 155	Y
23	hkstudio_glassescase_0	4, -4.5	3.14 -1.60, 271	Y
24	hkstudio_glassescase_1			-(not visible)
25	hkstudio_glassescase_2			-(perception)
26	hkstudio_laptop_0	-4, -3	-2.33, -2.90, 181	Y
27	hkstudio_laptop_1		-2.27, -2.59, 195	Y
28	hkstudio_laptop_2		-2.42, -2.93, 178	Y
29	hkstudio_laptop_3		-1.92, -2.92, 178	Y
30	hkstudio_laptop_4		-1.84, -2.92, 178	Y
31	hkstudio_laptop_5		-2.50, -3.99, 181	Y
32	hkstudio_laptop_6		-2.32, -2.92, 180	Y
33	hkstudio_laptop_7		-2.27, -2.90, 181	Y
34	hkstudio_monitor_0	4.5, 4		(not visible)
35	hkstudio_monitor_1		-0.08, 2.80, 0	Y
36	hkstudio_monitor_2		0.01, 2.90, 0	Y
37	hkstudio_monitor_3			(not visible)
38	hkstudio_monitor_4			(perception)
39	hkstudio_mug_0	0.9, -3.8		(pos)
40	hkstudio_mug_1		1.85 -2.75, 204	Y
41	hkstudio_mug_2			(collision)
42	hkstudio_statue_0			(not visible)
43	hkstudio_statue_1		-2.06, 2.96, 133	Y
44	onebedroom_fork_0	0.5, 8	1.39, 5.71, 88	Y
45	onebedroom_fork_1		1.48, 6.67, 110	Y
46	onebedroom_fork_2		1.43, 6.53, 110	Y
47	onebedroom_fork_3		1.45, 5.83, 88	Y
48	onebedroom_glassescase_0	3.5, 3	2.74 1.95, 46	Y
49	onebedroom_glassescase_1		2.85, 3.37, 316	Y
50	onebedroom_glassescase_2		1.77, 3.01, 1	Y

51	onebedroom_glassescase_3		3.18, 2.20, 316	(perception)
52	onebedroom_laptop_0		0.31, 1.82, 223	Y
53	onebedroom_laptop_1		0.72, 1.51, 178	Y
54	onebedroom_monitor_0	4.4, 5.5	3.41, 5.47, 0	Y
55	onebedroom_monitor_1		2.90, 5.30 1	Y
56	onebedroom_monitor_2		2.85, 5.31 1	Y
57	onebedroom_monitor_3		2.98, 5.42, 1	Y
58	onebedroom_mug_0	4.4, 0.8	2.90, 0.53, 1	Y
59	onebedroom_mug_1		3.16, 0.61, 358	Y
60	onebedroom_mug_2		3.05, 0.73, 1	Y
61	Onebedroom_mug_3		2.07, 0.52, 358	Y
62	onebedroom_statue_0	2.7, 6.7	3.65, 6.50, 178	Y
63	onebedroom_statue_1		3.63, 6.52, 178	Y
64	onebedroom_statue_2		2.99, 5.22, 88	Y
65	onebedroom_statue_3		2.96, 5.20, 89	Y
66	twobedrooms_fork_0	4.1, 6	2.29, 5.78, 2	Y
67	twobedrooms_fork_1		3.05, 6.02, 2	Y
68	twobedrooms_fork_2		3.00, 5.90, 1.	Y
69	twobedrooms_laptop_0	1, 6	1.40, 5.19, 358	Y
70	twobedrooms_laptop_1		0.39, 5.86, 1	Y
71	twobedrooms_laptop_2			(collision)
72	twobedrooms_laptop_3		-0.63, 5.97, 1.68	Y
73	twobedrooms_monitor_0	2.5, 7		(perception)
74	twobedrooms_monitor_1		2.32, 5.82, 88	Y
75	twobedrooms_mug_0	-1.1, 2		(ambiguity)
76	twobedrooms_mug_1		-0.55, 3.02, 268	Y
77	twobedrooms_statue_0	3.9, 1.6	2.53, 2.54, 314	Y
78	twobedrooms_statue_1		2.99, 1.82, 1	Y
79	twobedrooms_statue_2		3.03, 1.67, 20	Y

Vita

Zhiyu Huo

Suite A, 1010 Commercial Street, San Carlos, CA, 94070

The author, Zhiyu Huo, was born in Shenyang, Liaoning, China in 1988. He attended Northeastern University from 2006 to 2010 and received a Bachelor of Automation in June 2010. He began work toward a Master of Science in Electrical Engineering at the University of Missouri-Columbia College of Engineering in August 2010 and earned the M.S. degree in June 2013. He earned his Doctor in Philosophy in Electrical and Computer Engineering in June 2017.

Research interests include robotics, machine learning, computer vision, and natural language programming; Work experience includes graduate research assistance in the University of Missouri, teaching assistance in the University of Missouri. Software Engineer in Rokid Inc.