Abstract of Human-guided Robot Object Search, by Thao Nguyen, Ph.D., Brown University, May 2024.

Object search is a central problem for human-robot interaction, as finding, localizing, and then grasping an object is a first step for almost anything a person would want the robot to do in the physical world. Additionally, natural language and gesture are the two most popular communication modalities due to the familiarity and comfort they afford the majority of human users. Human-guided object search is a difficult problem as the robot must identify objects based on the natural language and gesture specifications, which might lack information and be ambiguous. Furthermore, object detection accuracy can suffer from sensor noise and the robot's partial observation of the environment.

This dissertation integrates language-conditioned visual models with a model-based decision-theoretic framework to enable effective robotic object search with complex natural language and gesture specifications. I will first present our research on affordance-based object retrieval which learns to encode language and visual inputs into a joint embedding space. Next, I will discuss our work on incorporating the language-conditioned visual detector's uncertainty into the planner's observation model for improved state estimation and object search. Lastly, I will describe our project on utilizing pointing gesture information in robot object search.

BROWN UNIVERSITY

DOCTORAL DISSERTATION

Human-guided Robot Object Search

Author:

Thao Nguyen

Advisor:

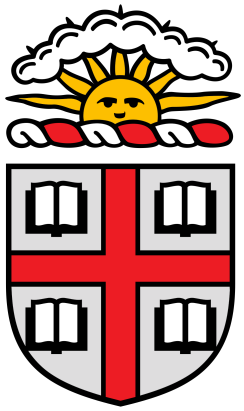
Stefanie Tellex

A dissertation submitted in fulfillment of the requirements

for the degree of Doctor of Philosophy

in the

Department of Computer Science



May, 2024

This dissertation by Thao Nguyen is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date _____ _____

Stefanie Tellex, Advisor

Recommended to the Graduate Council

Date _____ _____

Ellie Pavlick, Reader

Date _____ _____

Srinath Sridhar, Reader

Approved by the Graduate Council

Date _____ _____

Thomas A. Lewis, Dean of the Graduate School

Curriculum Vitae

Thao Nguyen was born and raised in Hanoi, Vietnam. She graduated from Hanoi-Amsterdam High School for the gifted, and went on to receive a B.A. in Computer Science and an Economics minor with honors from Vassar College. She then pursued a Ph.D. in Computer Science at Brown University and received her M.Sc. in 2020. She was the Lead Lecturer for the Artificial Intelligence course at Brown in Fall 2023. She successfully defended her thesis in April 2024. Thao's research is motivated by the desire to develop home and office robot assistants to help people with daily tasks, with a focus on improving robots' natural language and visual understanding for effective human-robot interaction and task completion.

"If I have seen further, it is by standing on the shoulders of giants."

Isaac Newton

Acknowledgments

My advisor, Stefanie Tellex, is an incredible person and role model. I have learned so much from her wisdom and guidance, and would not have completed my Ph.D. without her unwavering support. It has been an honor to be her student, and I hope to continue to make her proud.

I would like to thank George Konidaris, Ellie Pavlick, and Srinath Sridhar for all of their invaluable time and help. I am grateful to Nakul Gopalan for being an amazing mentor and friend that can always talk sense into me. My undergraduate professor, Eric Aaron, is the person who introduced me to research and robotics. He has impacted my life in so many wonderful ways and I cannot thank him enough.

My family has given me so much love and support and helped make me the person I am today. I am forever indebted and thankful to them. My partner, Eric Rosen, has been my rock and biggest champion. Thank you for helping me believe in myself and for all the love and happiness you share with Milo and I.

My friends at Brown and Vassar have made this journey infinitely easier and more fun. I am truly grateful to Deemer, who has helped me more times than I can count and is like the brother I never had. Ifrah has brought so much warmth and kindness into my life and always inspires me to become a better person. Lauren Clarke and Kerri Varela are absolutely awesome at their jobs and have never failed to brighten up my day. I am very glad to have gotten to know them and will miss them dearly.

I am so lucky to have received so much love. Thank you, thank you, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The grand promise of robotics is the development of autonomous agents that can act as personal assistants and perform a wide-variety of tasks for humans—from cleaning our homes to preparing meals. Despite this, robots have by and large been deployed in contexts where they are isolated from end-users and programmed by robot experts. One major reason for this is that robot programming requires expertise in a wide range of specialized tools and robot systems. A new paradigm is required to democratize the ability to program robots by better matching how humans naturally interface with each other. Developing robots that can understand natural language and gestures will enable end-users to seamlessly interact and collaborate with their robot assistants. However, natural language and gestures involve varying levels of complexity, ambiguity, and open-endedness, which makes understanding them a highly challenging problem.

Additionally, object search is a central problem for human-robot interaction, as finding, localizing, and then grasping an object is a first step for almost anything a person would want the robot to do in the physical world. Human-guided object search is a difficult task as the robot must infer information from the human's instructions and make sequential decisions under uncertainty on how to interact with the world and human user to efficiently search for objects. Our approach integrates language-conditioned visual models with a model-based decision-theoretic framework to enable effective robotic object search with complex natural language and gesture specifications.

Chapter 2 describes our system that learns to encode language and visual inputs into a

joint embedding space, enabling robots to take images of objects and reason about which object(s) is suitable for performing an action (expressed in natural language as verbs). Humans could ask for an object to act as a tool in some task (e.g., “something to cut food with"), just as naturally as they would with another person, and the robot can inspect objects in the scene and retrieve an appropriate one based on its appearance (e.g., having sharp edges). These results have been published at the 2020 Robotics: Science and System Conference (RSS) [64], and in the Autonomous Robots Journal (AURO) in 2022 [65]. In Chapter 3, I present a novel class of observation models, *language-conditioned observation models (L-COMs)*, that enable humans to easily interface with a robot via natural language to direct object search. Humans can express any information they know about the object, and my algorithm generates an observation model—using off-the-shelf pretrained visual-language object detectors—whose noise model adapts to the detectors' uncertainty. This enables the robot to effectively leverage a wide variety of natural language hints to efficiently and accurately locate the target object. This work was published at the 2023 International Conference on Intelligent Robots and Systems (IROS) [66]. Chapter 4 discusses our work on understanding human pointing gestures. We evaluated five different approaches to resolve pointing gestures to locations in the environment, as well as developed a probabilistic observation model for pointing gestures. This observation model can be incorporated with L-COMs to allow human users to naturally communicate with both unstructured language and pointing gestures, and robots to jointly utilize the multi-modal information for improved object search. This work will be published at the 2024 Conference of the Cognitive Science Society.

Chapter 2

Affordance-based Robot Object Retrieval

In this project, we developed a data-driven language-conditioned visual model to enable affordance-based robot object retrieval.

2.1 Introduction

A key bottleneck in widespread deployment of robots in human-centric environments is the ability for non-expert users to communicate with robots. Natural language is one of the most popular communication modalities due to the familiarity and comfort it affords a majority of users. However, training a robot to understand open-ended natural language commands is challenging since humans will inevitably produce words that were never seen in the robot's training data. These unknown words can come from paraphrasing such as using "saucer" instead of "plate," or from novel object classes in the robot's environments, for example a kitchen with a "rolling pin" when the robot has never seen a rolling pin before.

We aim to develop a model that can handle open-ended commands with unknown words and object classes. As a first step in solving this challenging problem, we focus on the natural language object retrieval task — selecting the correct object based on an indirect natural language command with constraints on the object's functionality. More specifically, our work focuses on fulfilling commands requesting an object for a task specified by a verb such as "Hand me a box cutter to **cut**." Being able to handle these types of commands is highly useful

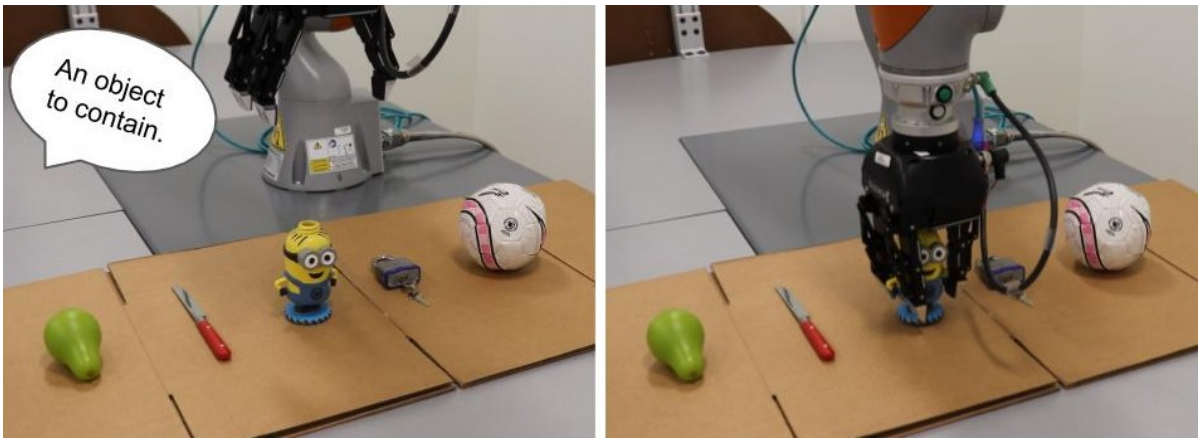


FIGURE 2.1: Our robot receives segmented RGB images of the objects in the scene and a natural language command “An object to contain," and correctly retrieved the Minion (yellow cartoon character)-shaped container.

for a robot agent in human-centric environments, as people usually ask for an object with a specific usage in mind. The robot would be able to correctly fetch the desired object for the given task, such as cut, without needing to have seen the object, a box cutter for example, or the word representing the object, such as the noun “box cutter." In addition, the robot has the freedom to substitute objects as long as the selected object satisfies the specified task. This is particularly useful in cases where the robot cannot locate the specific object the human asked for but found another object that can satisfy the given task, such as a knife instead of a box cutter to cut.

There has been much prior work on natural language object retrieval [50, 39, 16, 17] and similar areas such as image captioning and image retrieval [70, 57, 88, 99]. However, previous work primarily focuses on natural language commands that either specify the object class such as “scissors" or describe its visual attributes such as “red," “curved," “has handle,” and cannot handle unknown object classes or words. Our work, in contrast, anchors the desired object to its usage (which is specified by a verb) and reasons about the verb to handle unknown objects and nouns on-the-fly.

Our work demonstrates that an object’s appearance provides sufficient signals to predict whether the object is suitable for a specific task, without needing to explicitly classify the object class and visual attributes. Our model takes in RGB images of objects and a natural language command containing a verb, generates embeddings of the input language command

and images, and selects the image most similar to the given command in embedding space. The selected image should represent the object that best satisfies the task specified by the verb in the command. We train our model on natural language command-RGB image pairs. The evaluation task for the model is to retrieve the correct object from a set of five images, given a natural language command. We use ImageNet images and language commands generated from verb-object pairs extracted from Wikipedia to train our model. Our model achieves a mean reciprocal rank of 77.4% on a held-out test set of unseen ImageNet object classes and 69.1% on unseen object classes *and* unknown nouns. Our model also achieves a mean reciprocal rank of 71.8% on unseen YCB object classes, which have a different image distribution from ImageNet. In addition, we demonstrate our model on a KUKA LBR iiwa robot arm, enabling the robot to retrieve objects based on natural language commands[1], and present a new dataset of 655 verb-object pairs denoting object usage over 50 verbs and 216 object classes[2].

2.2 Related Work

Natural language object retrieval refers to the task of finding and recovering an object specified by a human user using natural language. The computer vision and natural language grounding communities attempt to solve object retrieval by locating or *grounding* the object specified in an image using natural language [50, 39, 78]. Krishnamurthy and Kollar [50] use a dataset of RGB images with segmented objects and their natural language descriptions to learn the grounding of words to objects in the image by exploiting repeated occurrences of segmented objects within images, along with their descriptions in natural language. Hu et al. [39] use a similar approach albeit using deep neural networks to avoid parsing and feature construction by hand. Schlangen et al. [78] learn individual classifiers for words and compose them to resolve object references. Similar to Krishnamurthy and Kollar [50] and Hu et al. [39], they focus on words that specify object categories or visual attributes such as “cup," “red," “right," and verbs that describe human activities such as “stand," “run," and cannot handle unknown object classes. Chen et al. [16] learn joint embeddings of language descriptions

[1]Video recordings of the robot demonstrations can be found at https://youtu.be/WMAdGhMmXEQ.

[2]The dataset and code for the project can be found at https://github.com/Thaonguyen3095/affordance-language.

and colored 3D objects for text-to-shape retrieval and generation of colored 3D shapes from natural language. Cohen et al. [17] learn joint embeddings of language descriptions and segmented depth images of objects for object retrieval within instances of the same object class. Our work, in contrast, learns an embedding across object classes based on their suitability for a given task specified using natural language. Our object embeddings are not conditioned on the output class of objects, but on the relevancy of the object for the specified task. This, therefore, allows us to retrieve objects based on descriptions of their usage and importantly allows handling of unknown nouns and unseen object classes.

Another relevant line of work is image captioning and image retrieval, which also aims to jointly model a natural language sequence and image content. The SUN scene attribute dataset [70] maps images to attributes such as “hills," “houses," “bicycle racks," “sun," etc. Such understanding of image attributes provides scene category predictions and high level scene descriptions, for example “human hiking in a rainy field." Methods based on recurrent neural networks (RNNs) [57, 88, 99] trained to directly model the probability distribution of generating a word given previous words and an image have shown to be effective in image caption generation, natural language image retrieval, and image caption retrieval. Our work is most similar to earlier attribute based image retrieval work. However, these models are trained on attributes that are directly specified and not inferred from indirect task based queries. Implicit task-based object attributes are harder to learn but are also more general than directly specifiable object visual attributes, and are useful for a natural language object retrieval system to have in its toolbox.

Object functionalities or *affordances* have been studied for a long time. Gibson [28] defines affordances as the “action possibilities" available to an agent, which covers a large range of actions. Chao et al. [14] mine the web for the knowledge of semantic affordance — given an object, determining whether an action can be performed on it. In contrast, our work focuses on actions that can be performed *with* the objects, which is similar to the affordances studied by Myers et al. [62], Do et al. [20], and Fulda et al. [26]. However, Myers et al. and Do et al. focus on detecting the affordances of object parts, and only work with a small number of object classes and affordances. Fulda et al. extract verb-noun pairs from word embeddings

trained on Wikipedia, but do not take into account objects' visual appearance and cannot handle unknown objects.

Also related to our work are methods on interactive object retrieval [93] and language grounding [79, 32]. These methods perform inference over dialogue to deduce the right object based on the specifications provided by the human user. However, they use known object corpora and directly specify the object attributes. Our work, in contrast, retrieves objects based on contextual information about the task specified by the language command. We are not performing inference over dialogue, but it is a natural next step for our work where our joint embedding can prove useful in the case of novel objects.

Similar to previous work, our work aims to learn joint object representations from visual and language information using RNNs. However, previous work primarily focuses on language commands specifying the object type or visual attributes such as "scissors," "red," "has handle." In contrast, our work focuses on fulfilling commands requesting an object for a task specified by a verb, for example "Hand me something to **cut**." To handle such commands, a possible approach is to rely on accurate classification of the object type and visual attributes and an external knowledge base to query for valid verb-object or verb-attribute pairings. However, that approach would be limited to known objects and words. Our work, on the other hand, bypasses classification of object type and attributes, and directly maps object use that is specified by the verb to object appearance that is captured by the image. Our work uses the context of the verb to implicitly infer the necessary object attributes for the task, and can generalize to unseen object classes and unknown nouns in the language commands.

2.3 Approach

To fulfill natural language commands requesting an object for a task specified by a verb, our model generates embeddings for the language command and candidate objects and selects the object that is closest to the command in embedding space. Our model is trained using pairs of natural language object requests containing verbs and ground truth objects that satisfy the requests. We describe our model and data collection process in detail in Sections 2.3.1 and 2.3.2.

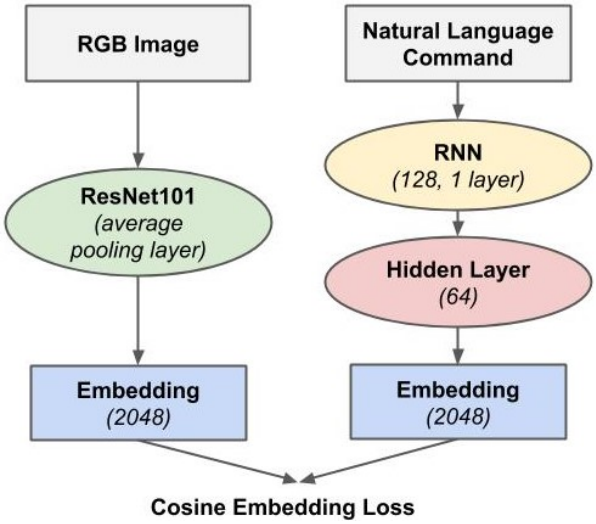


FIGURE 2.2: Diagram of the language-vision embedding model. The model encodes given natural language commands and RGB images, and minimizes the cosine embedding loss between the language and image embeddings during training. At inference time, the model calculates the cosine similarities between the embeddings of the language command and candidate images, and selects the image most similar to the command in embedding space.

2.3.1 Model

Given a natural language command and images of candidate objects, we want our model to correctly select the object that best satisfies the command. Our model does this by generating embeddings for the input natural language command and images, calculating the cosine similarities between the image embeddings and the language embedding, and selecting the image most similar to the command in embedding space. A diagram of our model is shown in Figure 2.2. Our model consists of separate image and language encoders, each in charge of generating embeddings for the input natural language commands and RGB images, respectively. During training, our model minimizes the cosine embedding loss between the embeddings of language command-RGB image pairs, thus maximizing the likelihood of the target image given the command. We describe the component image and language encoders and our model training process below.

2.3.1.1 Image Encoder

To encode each RGB image, we use the average pooling layer of a pretrained ResNet101 [33]. We chose ResNet101 due to ResNet's good performance on RGB images captured by common robot sensors [56]. The use of deep pretrained representations enables our model to leverage prior information of complex image features to allow for better encoding of the visual information from the images. We get an embedding of size 2048 from the pretrained ResNet model for each image.

2.3.1.2 Language Encoder

To encode each natural language command, our language encoder consists of a recurrent neural network (RNN) [22] followed by a fully connected layer. We randomly initialize word embeddings for each language command that are then trained from scratch. The language encoder produces an embedding vector that is the same size as the embedding produced by the image encoder.

2.3.1.3 Training Process

We train our model to optimize an objective function that attempts to bring the corresponding language and image embeddings closer to each other in embedding space. We achieve this by reducing the cosine embedding loss between the embedding produced by the image encoder, which takes in an RGB image of the object, and the embedding produced by the language encoder, which takes in the referring natural language command.

We describe the training data in Section 2.3.2.3. Positive training samples consist of pairs of natural language commands, each containing one verb, and RGB images of objects that can be paired with that verb. We obtain negative samples by randomly sampling an image of a different object that does not correspond with the verb and pairing the image with the language command containing the verb, resulting in a dataset of equally balanced positive and negative samples. We use Adam [46] as an optimiser with a learning rate of 0.0001 and train for 50 epochs until convergence.

2.3.2 Data Collection

To train and evaluate our model, we require pairs of natural language commands containing verbs and RGB images of objects. To obtain these command-image pairs, we need verb-object pairs denoting valid object usage such as “cut" for a “knife." We also require RGB images for the objects. Since we are interested in testing our model’s generalization capability on unseen object classes and nouns, we require a large number of object classes that can be paired with the verbs for a sufficient number of held-out object classes.

To the best of our knowledge, no existing datasets of verb-object pairs met our requirements. Chao et al. [14] collected a semantic affordance dataset of verb-noun combinations. However, their dataset only has 20 object classes, and focuses on verbs denoting actions that can be performed on the objects, such as “hunt" a “bird," rather than the objects’ usage. Other works on semantic affordances [62, 20] also provide datasets of objects labeled with their affordances. However, these datasets contain fewer than 20 object classes and fewer than 10 affordances. Fulda et al. [26] extracted verb-noun pairs from Wikipedia to enable an agent to play text-based adventure games, but do not consider the objects’ visual appearances. We, therefore, decided to collect our own dataset of valid verb-object pairs and use it to generate natural language commands paired with RGB images. We describe our data below.

2.3.2.1 Vision Data

We use RGB images and object classes from the ImageNet Large Scale Visual Recognition Challenge (ILSVRC2012) validation set [77]. We chose this dataset as it has 1000 object classes and a variety of images per object class, and we want our model to work on many different object classes and object instances. The ImageNet object classes such as “violin,” “suit,” and “quill” are usually nouns that occur frequently in textual data in correspondence with other verbs such as “play,” “wear,” “write.”

2.3.2.2 Language Data

We extracted sentences from Wikipedia containing the ImageNet object classes and used spaCy [37] to parse the sentences and extract corresponding verb-object pairs. We originally

TABLE 2.1: Example verb-object pairs from our dataset

contain – bucket	hit – hammer	wear – necklace
contain – wardrobe	hit – racket	wear – suit
cut – cleaver	play – baseball	wrap – cloak
cut – hatchet	play – violin	wrap – handkerchief
eat – banana	serve – plate	write – notebook
eat – pizza	serve – tray	write – quill

sought out to extract verb-object pairs from the common-sense knowledge base ConceptNet [83], which ended up being too small and was missing many valid verb-object pairings. We then decided to use Wikipedia instead for its large text corpus. However, the resulting dataset was highly noisy with 20,198 verb-object pairs, containing many abstract verbs such as “name,” “feature,” “use” that can be paired with any object, or nouns in the wrong word sense such as “suit” in “follow suit” and “file suit” that did not refer to physical objects and were not relevant for the natural language object retrieval task we were interested in. Therefore, we manually annotated the verb-object pairs to retain only concrete verbs paired with nouns in the correct sense such as “wear" and “suit," where “suit" refers to the clothing item. This resulted in a dataset with 655 verb-object pairs over 50 verbs and 216 object classes. Example verb-object pairs from our dataset are shown in Table 2.1.

2.3.2.3 Training and Testing Data

We use 80% of the 216 object classes and their corresponding 535 verb-object pairs to generate our training data. The training data consist of natural language commands paired with RGB images. For each verb-object pair, we randomly select sentence templates to generate language commands and pair them with different image instances of that object class. Table 2.2 lists the templates we use. Examples of the training data are shown in Table 2.3. Rather than using only the verbs and/or nouns from the verb-object pairs as language data for our model, we generate and give our model natural language sentences so that it can handle more realistic language commands, as most people do not ask for objects with one or two-word commands such as “knife” or “knife cut.”

TABLE 2.2: Templates for Language Command Generation

Verb only	An item that can <verb>. An object that can <verb>. Give me something that can <verb>. Give me an item that can <verb>. Hand me something with which I can <verb>. Give me something with which I can <verb>. Hand me something to <verb>. Give me something to <verb>. I want something to <verb>. I need something to <verb>.
Verb and object	Give me the <object> to <verb>. Hand me the <object> to <verb>. Pass me the <object> to <verb>. Fetch the <object> to <verb>. Get the <object> to <verb>. Bring the <object> to <verb>. Bring me the <object> to <verb>. I need the <object> to <verb>. I want the <object> to <verb>. I need a(n) <object> to <verb>. I want a(n) <object> to <verb>.

TABLE 2.3: Example training data (command-image pairs)

Verb-Object	Language Command	Image
contain – cup	*Give me an item that can contain*	
	I need something to contain	
play – drum	*Hand me something to play*	
	I want an object to play	
wear – kimono	*An item to wear*	
	Give me something to wear	

We hold out 20% of the object classes for testing. Test examples consisting of language command-set of five images pairs are randomly generated from objects in the test set and their corresponding verb-object pairs. The evaluation task is to select the correct object from a set of five images given a natural language command.

2.4 Experiments and Results

2.4.1 Qualitative Evaluation of Learned Task Embeddings

We rank all test objects based on their embeddings' similarity to the model's output embedding for each task (verb) and analyze the ranked lists to qualitatively evaluate the learned task embeddings. We use cosine similarity as the similarity metric. The test set contains 43 object classes and 118 object instances, which belong to the held-out 20% of the object classes. We report the top-10 and bottom-10 ranked objects for 6 verbs: "wear," "perform," "play," "serve," "contain," and "cut" in Tables 2.6–2.9. For each object, we show its image, class label, and embedding similarity score.

Table 2.4 shows the results for "wear." The top-9 objects are all items that can be worn. And while the 10th object is labeled as a puck, its image mostly depicts people in hockey uniforms, which are wearable objects. Our model was able to learn good embeddings for both tasks.

Results for "perform" can be found in Table 2.5. The majority of the top-10 objects are either musical instruments or stages, which are both compatible with "perform." Cleavers and shields are most likely incorrectly included in the top-10 due to their metallic appearances, which is a characteristic shared by several musical instruments in the training set (such as flute, gong, and trombone). In addition, when comparing the top-10 and bottom-10 ranked objects, we see that the images for the top-10 objects have warmer tones, whereas those for the bottom-10 have cool tones. This contrast likely led to the violin with a black-and-white image being misranked and included in the bottom-10.

We report the results for "play" in Table 2.6. Similar to "perform," most of the top-10 objects are musical instruments or stages, which are good matches for "play." The 8th and 9th objects' labels are library and castle, respectively, which do not seem to match the task. However, the

TABLE 2.4: Object rankings based on their embeddings' similarity score to the model's learned embedding for the verb "wear"

Top-10			**Bottom-10**		
Image	Label	Score	Image	Label	Score
	apron	0.3966		stage	-0.0158
	apron	0.3429		monastery	-0.0198
	cardigan	0.3168		castle	-0.0230
	wool	0.3057		mosque	-0.0245
	gown	0.2947		castle	-0.0289
	shield	0.2809		library	-0.0348
	gown	0.2584		library	-0.0374
	shield	0.2522		missile	-0.0460
	diaper	0.2459		library	-0.0477
	puck	0.2437		mosque	-0.0837

TABLE 2.5: Object rankings for the verb “perform"

Top-10			**Bottom-10**		
Image	Label	Score	Image	Label	Score
	sax	0.4795		palace	0.0741
	stage	0.4757		violin	0.0741
	sax	0.4039		castle	0.0733
	violin	0.3862		screen	0.0701
	violin	0.3583		palace	0.0636
	violin	0.3401		castle	0.0536
	cleaver	0.3359		missile	0.0500
	shield	0.3195		monastery	0.0454
	shield	0.2872		mosque	0.0287
	stage	0.2811		missile	-0.001

TABLE 2.6: Object rankings for the verb “play"

Top-10			**Bottom-10**		
Image	Label	Score	Image	Label	Score
	sax	0.5187		corn	0.0663
	stage	0.4542		castle	0.0657
	sax	0.4014		screw	0.0636
	violin	0.3841		burrito	0.0610
	stage	0.3668		library	0.0574
	stage	0.3624		missile	0.0569
	violin	0.3085		screen	0.0531
	library	0.2930		wool	0.0370
	castle	0.2831		cheeseburger	0.0361
	violin	0.2797		screw	0.0339

image for the library depicts a performance or play on an impromptu stage, while the castle's image with bright lights and a clear path resembles the appearance of a stage. In the training set, "play" is also paired with sports equipment such as basketball, baseball, and volleyball. Puck is the only sports equipment in the test set, and a puck is ranked at number 11 in its suitability for "play."

Table 2.7 shows the results for "serve." The top-10 objects include food, dishware, and restaurants, all of which are suitable for serving. Although mortar does not seem to be a good match for the task, its image resembles a cup with a spoon, which can be used to serve. The bubble, diaper, and syringe are incorrectly included in the top-10 objects. This is most likely due to the presence of people in their images, as people are often seen in the images of objects that are paired with "serve" in the training data.

Results for "contain" can be found in Table 2.8. Most of the top-10 objects are buildings, which can contain many things, but are not often what people have in mind when asking for objects to contain. However, the 4th and 9th objects, tub and restaurant, are reasonable choices for containers. The restaurant's image depicts a dining table with a glass bowl in the center, as well as multiple wine glasses and bottles, which are all suitable for containing. Goblets, another good match for "contain," are ranked relatively high at numbers 21, 31, and 39 (the majority of objects at number 11-20 are other buildings).

Table 2.9 shows the results for the verb "cut." The top choice, cleaver, has a significantly higher similarity score (0.4985) than other objects in the top-10 ($<= 0.3746$). It is also the only object instance in the test set that can be paired with "cut." While the 6th object is labeled as a tub, its image includes a pair of pliers, which can indeed be used to cut. The image for the 9th object, goblet, shows different goblets with decorative stems, the majority of which has sharp-looking parts that can be mistaken to be fit for cutting. The rest of the top-10 objects mostly have metallic appearances and/or slim bodies, which are characteristics shared by objects that are paired with "cut" in the training set.

Our model was able to learn good embeddings for a number of verbs to perform object retrieval based on descriptions of the object's usage. However, it was unable to learn better embeddings for more difficult verbs such as "serve" and "contain." As our model does not

TABLE 2.7: Object rankings for the verb “serve"

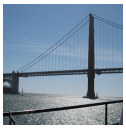
Top-10			**Bottom-10**		
Image	Label	Score	Image	Label	Score
	restaurant	0.5170		palace	0.0652
	burrito	0.4788		castle	0.0609
	goblet	0.4260		palace	0.0558
	goblet	0.4017		mosque	0.0536
	mortar	0.3703		screen	0.0483
	bubble	0.3374		palace	0.0443
	restaurant	0.3322		stage	0.0378
	cheeseburger	0.3296		castle	0.0376
	diaper	0.3252		pier	0.0360
	syringe	0.3163		pier	0.0157

TABLE 2.8: Object rankings for the verb “contain"

Top-10			**Bottom-10**		
Image	Label	Score	Image	Label	Score
	library	0.4577		swing	0.0977
	library	0.4530		screw	0.0969
	restaurant	0.4320		broom	0.0935
	tub	0.4205		fly	0.0928
	monastery	0.4196		violin	0.0866
	mosque	0.4185		cleaver	0.0830
	library	0.4068		screw	0.0495
	monastery	0.4005		stage	0.0477
	restaurant	0.3985		sax	0.0462
	library	0.3902		screw	0.0312

TABLE 2.9: Object rankings for the verb “cut"

Top-10			**Bottom-10**		
Image	Label	Score	Image	Label	Score
	cleaver	0.4985		pier	0.0493
	screw	0.3746		screen	0.0479
	broom	0.3494		stage	0.0435
	screw	0.2979		stage	0.0404
	missile	0.2944		restaurant	0.0336
	tub	0.2884		palace	0.0281
	violin	0.2807		mosque	0.0231
	screw	0.2773		library	0.0099
	goblet	0.2635		palace	0.0081
	swing	0.2517		castle	-0.0518

take into account the objects' labels for the retrieval task, it performs well even in cases where the label does not reflect what is shown in the object's image. The downside of this is that the model relies solely on the objects' appearances, which can be affected by extraneous properties such as the image tone and lighting conditions. Furthermore, our model learns to output embeddings that are close to the corresponding objects' embeddings, and thus is confined within the structure of the image encoder's output embedding space. Augmenting the data with depth information and image color augmentation techniques might help the model learn to pay less attention to extraneous image properties and achieve better generalization.

2.4.2 Quantitative Evaluation

The aim of our evaluation is to test our model's ability to accurately select objects based on natural language descriptions of their usage, given unseen object classes and unknown nouns in the language commands. Generalization to unseen object classes is much more difficult than to unseen instances of known object classes, as different instances of the same object class such as two bottles would usually look more alike than instances from different object classes such as a bottle and a bowl, even if those object classes can be used for similar tasks such as "contain."

The trained model is tested on natural language object retrieval tasks: given a language command containing a verb, along with five objects, it must output the correct object that can be paired with the verb. The evaluation task is modeling a typical in-the-wild retrieval task where there are a few objects on a table and the robot has to pick the correct one. Retrieval examples consisting of a language command paired with a set of five images are randomly generated from objects in the test set and their corresponding verb-object pairs. We test our model on several different test sets — the held-out ImageNet object classes and YCB object set — and report average top-1 and top-2 retrieval accuracies, as well as the mean reciprocal rank. Top-1 accuracy means that the model's top choice is the correct answer, and top-2 accuracy means that the correct answer is among the model's top-2 choices. Mean reciprocal rank (MRR) is the average of the multiplicative inverse of the correct answer's rank in the model's output ordered list of choices.

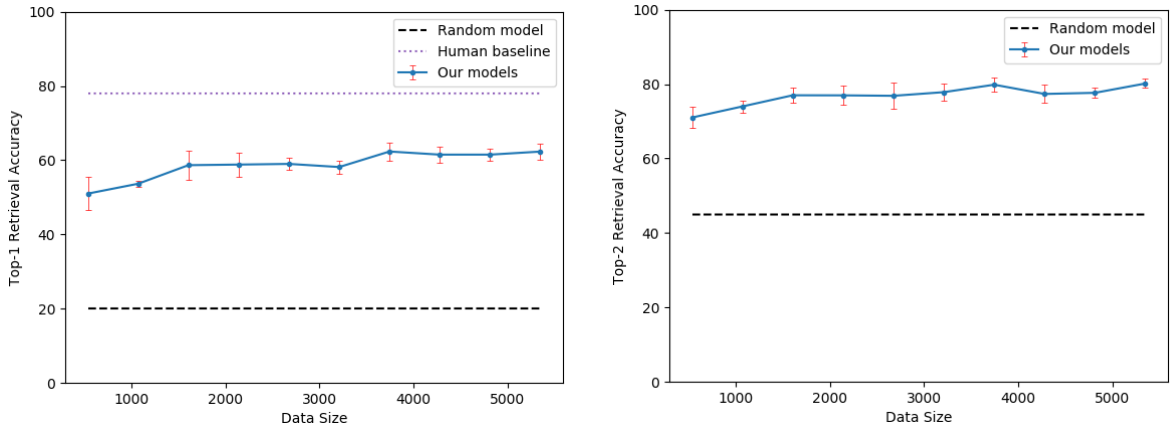
TABLE 2.10: Retrieval accuracies (%) over sets of 5 candidates from held-out ImageNet object classes

Model	Top-1 *(Std. Error)*	Top-2 *(Std. Error)*
Random	20.0	45.0
Data size 535	51.0 *(4.50)*	71.0 *(2.84)*
Data size 1070	53.7 *(0.86)*	74.0 *(1.60)*
Data size 1605	58.7 *(3.85)*	77.0 *(2.13)*
Data size 2140	58.8 *(3.31)*	77.0 *(2.50)*
Data size 2675	59.0 *(1.68)*	76.9 *(3.55)*
Data size 3210	58.2 *(1.70)*	77.8 *(2.30)*
Data size 3745	**62.3 *(2.48)***	79.8 *(1.94)*
Data size 4280	61.5 *(2.25)*	77.4 *(2.44)*
Data size 4815	61.5 *(1.71)*	77.7 *(1.43)*
Data size 5350	**62.3 *(2.18)***	**80.2 *(1.23)***
Human baseline	78.0 *(1.72)*	

2.4.2.1 Held-out ImageNet Object Set

We first test our model on object classes held out from our dataset, which have a similar image distribution to that of the training set, as the images all come from the ILSVRC2012 validation set. We evaluate our model under 2 different settings, representing increasingly difficult scenarios. We describe the test settings and our model's performance in each case below. For both cases, the test object classes are held-out, meaning our model has never seen any instances belonging to the test object classes during training.

2.4.2.1.1 Unseen Object Classes We first train and test our model on language commands containing only the verbs from the verb-object pairs, such as "I want something to `<verb>`." The templates for these commands are listed under the "Verb only" row in Table 2.2. This simplified setting where the test commands look like those in the training data, for example "Give me something to contain," helps us look at how well the model has learned to generalize the concepts associated with the verbs, such as "contain" requires objects with convexity. It removes the additional challenges that might arise with seeing unknown nouns in the commands, such as the fact that the embeddings for these nouns would be untrained. However, this is still a challenging problem because the model has never seen the object classes in the test set before.



(a) Top-1 retrieval accuracies (%)

(b) Top-2 retrieval accuracies (%)

FIGURE 2.3: Average retrieval accuracies over sets of 5 candidate objects (from unseen object classes from the held-out ImageNet object set) of models trained on different data sizes, represented by the solid lines, with vertical bars denoting standard errors. Models trained on larger data sizes usually perform better. All our models significantly outperform a random model, represented by the dashed lines. The dotted line represents the human object retrieval baseline.

We train separate models on increasing sizes of training data generated from the 535 verb-object pairs in the training set. Data was augmented by generating different natural language commands containing the verbs, and pairing the commands with different images of the objects from the verb-object pairs. Examples of the training data are shown in Table 2.3. The test set with 43 held-out object classes and 120 corresponding verb-object pairs and retrieval examples are fixed for all models.

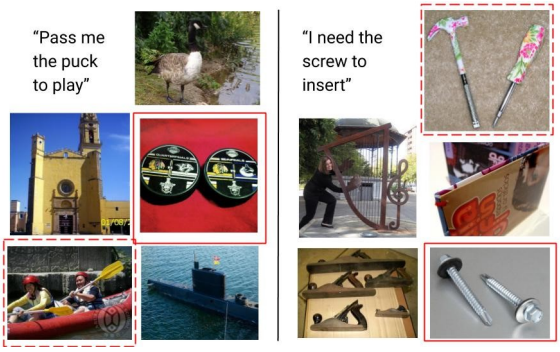
We test each model trained on different data sizes 5 times and report their average top-1 and top-2 retrieval accuracies and standard errors in Table 2.10 and Figure 2.3. Model performance generally increases with larger training size. All our models significantly outperform a random model (which has 20% top-1 and 45% top-2 retrieval accuracy) and achieve accuracies in the 50% – 62% range for top-1, and 70% – 80% for top-2. Our best average retrieval accuracy is 62.3% for top-1 and 80.2% for top-2 with standard errors of 2.18% and 1.23%, respectively. The best mean reciprocal rank is 77.4% with a standard error of 1.68%, as shown in Table 2.11. Our models were able to generalize to unseen object classes.

Our models were able to select the correct object to satisfy the task specified by most verbs in our dataset such as “contain," “write," “don," “rotate," “hit," etc. The objects paired with

(a) Model success cases. The solid boxes denote the model's top choice for each task.

(b) Model failure cases. The solid boxes denote the ground truth image that satisfies each command, and the dashed-line boxes denote the images the model actually chose.

(c) Model success cases. The solid boxes denote the model's top choice for each task.



(d) Model failure cases. The solid boxes denote the ground truth image that satisfies each command, and the dashed-line boxes denote the images the model actually chose.

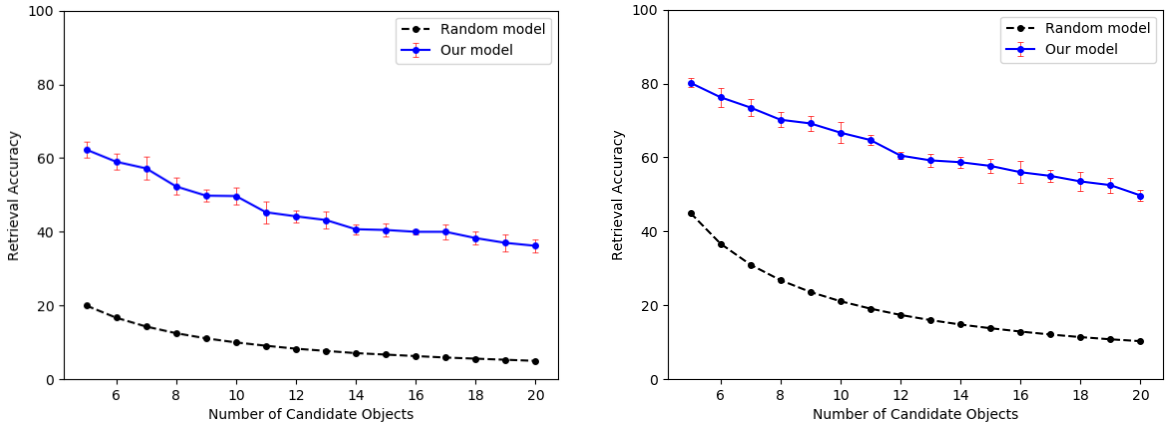
FIGURE 2.4: Example success and failure cases for our best models on object retrieval tasks with unseen object classes from the held-out ImageNet object set, shown in (a) and (b), and with both unseen object classes *and* unknown nouns, shown in (c) and (d). The models were given a natural language command and selected the object that they determine to best satisfy the command out of a set of five objects.

TABLE 2.11: Best retrieval accuracies and mean reciprocal ranks (%) over varying numbers of candidate objects from held-out ImageNet object classes

Candidate number	Top-1	Top-2	MRR
5 objects	62.3 *(2.18)*	80.2 *(1.23)*	77.4 *(1.68)*
6 objects	59.0 *(2.16)*	76.3 *(2.54)*	74.2 *(1.93)*
7 objects	57.2 *(3.13)*	73.5 *(2.23)*	72.0 *(2.71)*
8 objects	52.3 *(2.30)*	70.2 *(2.01)*	68.6 *(1.14)*
9 objects	49.8 *(1.67)*	69.2 *(2.01)*	66.5 *(0.95)*
10 objects	49.7 *(2.35)*	66.7 *(2.77)*	65.5 *(2.21)*
11 objects	45.3 *(3.03)*	64.7 *(1.39)*	62.5 *(2.00)*
12 objects	44.2 *(1.70)*	60.5 *(0.91)*	60.7 *(1.09)*
13 objects	43.2 *(2.35)*	59.2 *(1.86)*	59.3 *(2.02)*
14 objects	40.7 *(1.39)*	58.7 *(1.45)*	57.8 *(0.49)*
15 objects	40.5 *(1.78)*	57.7 *(1.94)*	57.0 *(1.10)*
16 objects	40.0 *(0.62)*	56.0 *(2.96)*	56.5 *(0.97)*
17 objects	40.0 *(2.04)*	55.0 *(1.72)*	56.2 *(1.40)*
18 objects	38.3 *(1.78)*	53.5 *(2.61)*	54.4 *(1.27)*
19 objects	37.0 *(2.27)*	52.5 *(2.01)*	53.3 *(1.17)*
20 objects	36.2 *(1.72)*	49.7 *(1.39)*	51.9 *(0.99)*

these verbs usually have similar visual appearances and attributes. For example, a gown and a suit can both be paired with “don" and are both made of fabric. However, our models performed not as well on more abstract verbs such as “play" and “protect," as it is less obvious which object attributes are required for the tasks specified by these verbs. The objects that can satisfy the tasks come in a larger variety of visual appearances, for example a harp, a volleyball, and a swing can all be paired with “play."

Example success and failure cases for our best model are shown in Figures 2.4(a) and 2.4(b), respectively. Our model correctly selected images of screws and goblets to satisfy natural language commands “Hand me something to rotate" and “Give me something with which I can serve." The model failed to select the swing given the command “I want something to play," and did not select the shield in response to “An object with which I can protect." However, the instance of a shield in the object retrieval task shown in the right image in Figure 2.4(b) does not look like a shield but more like a plate, and such object instance outliers can definitely throw the model off.



(a) Top-1 retrieval accuracies (%)

(b) Top-2 retrieval accuracies (%)

FIGURE 2.5: Average retrieval accuracies of our best model over varying number of candidate objects (from the held-out ImageNet object classes), represented by the solid lines, with vertical bars denoting standard errors. Our model significantly outperforms a random model, represented by the dashed lines.

We also evaluate our best model's retrieval performance over varying numbers of candidate objects. The model's average top-1 and top-2 retrieval accuracies, mean reciprocal ranks, and corresponding standard errors can be found in Table 2.11. Our model's performance is the best with five candidates, and decreases as the number of candidate objects increases. However, the average rates of decline in our model's retrieval accuracies are only 3.5% and 3.1% for top-1 and top-2, which are lower than those of a random model (8.8% and 9.3%, respectively), as visualized in Figure 2.5. Therefore, our model still significantly outperforms the random model and achieves reasonable performance levels with larger numbers of candidate objects for retrieval. For example, with 20 candidates, the random model would have a 5.0% top-1 and 10.3% top-2 retrieval accuracy, while our model achieved an average top-1 accuracy of 36.2% and 49.7% for top-2.

2.4.2.1.2 Unseen Object Classes and Unknown Nouns Next, we train and test our model on natural language commands containing both verbs and objects, for example "Give me the `<object>` to `<verb>`." The templates for these commands are listed under the "Verb and object" row in Table 2.2. The model is tested on object retrieval tasks with both unseen object classes *and* unknown nouns. Testing our model in this setting is necessary because when a deep net such as our model is dealing with an unknown word, it will map the word to

TABLE 2.12: Best retrieval accuracies and mean reciprocal ranks (%) over sets of 5 candidate objects for each experiment setting

Setting	Top-1	Top-2	MRR
Held-out ImageNet objects	62.3 *(2.18)*	80.2 *(1.23)*	77.4 *(1.68)*
Held-out objects & nouns	53.0 *(1.33)*	72.8 *(3.11)*	69.1 *(0.87)*
Unseen YCB objects	54.7 *(1.99)*	71.9 *(2.40)*	71.8 *(1.50)*

a random, untrained embedding. That random embedding could completely throw off the model's understanding, for example the model might pick the image for whatever noun the random embedding happens to be closest to. We need to know how our model would behave with truly unknown words in the input to get a sense of how it would work in the real world.

An example task in this setting is the model getting the command "Give me the dax to cut" with "dax" being an unknown word to the model, while also being shown object instances from classes it has never seen before. This task is more difficult than object retrieval with only unseen object classes, as the model has to figure out that unknown words such as "dax" adds no information and avoid being affected by the noise added by the unknown words.

Other than the inclusion of nouns in the language commands, the setup for this experiment is the same as that with only unseen object classes, as described in Section 2.4.2.1.1. Each model trained on different data sizes was tested 5 times. As shown in Table 2.12, our best average retrieval accuracy is 53.0% for top-1 and 72.8% for top-2, with standard errors of 1.33% and 3.11%, respectively. The best mean reciprocal rank is 69.1% with a standard error of 0.87%. Our models were negatively affected by the unknown words in the language commands. However, performance decline is to be expected as this is a more difficult task. Furthermore, our models' performance still demonstrates some generalization to unseen object classes *and* unknown nouns. A way to better handle unknown words and boost model performance in this setting would be to use pretrained word embeddings such as Word2vec [61] or GloVe [71] instead of random untrained embeddings.

Example success and failure cases for our best model are shown in Figures 2.4(c) and 2.4(d), respectively. Our model was able to correctly select the paintbrush when asked to "Bring me the paintbrush to write," and picked the hammer to satisfy the command "Get

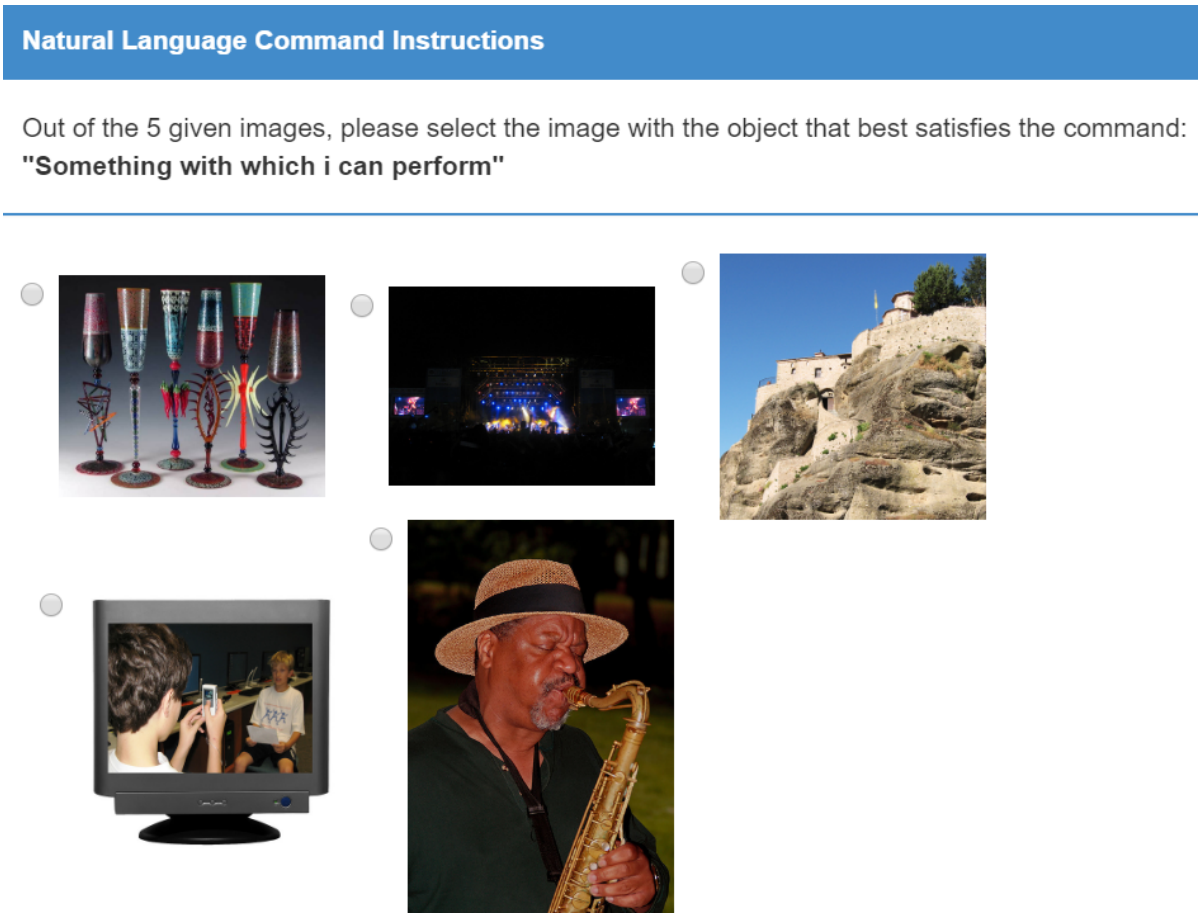


FIGURE 2.6: Amazon Mechanical Turk interface and example task for human baseline experiment.

me the hammer to hit." Unfortunately, the model incorrectly selected the canoe and hammer given the commands “Pass me the puck to play," and “I need the screw to insert," respectively. However, a canoe can also be paired with “play" as canoeing is a recreational activity. In addition, the image representing the hammer also includes a screwdriver, an object that can be used to “insert."

2.4.2.2 Human Retrieval Baseline

We also compare our models’ performance to a human baseline for the retrieval task. Humans are experts in natural language understanding and object grounding. The experiment was done on Amazon Mechanical Turk (AMT). We showed AMT workers five images and one language command such as “Give me something to `<verb>`," and asked them to select the

TABLE 2.13: Verbs annotated with YCB objects

construct	eat	open	serve
contain	grow	play	write
cut	hit	rotate	

image with the object that best satisfies the command. The AMT interface and an example task for the experiment is shown in Figure 2.6. We collected 5 answers for each of the 120 retrieval tasks. The average top-1 human retrieval accuracy is 78.0% with standard error of 1.72%, shown in Table 2.10 and Figure 2.3(a). The Fleiss' Kappa [25] is 0.66, which reflects a good level of inter-rater agreement. Even human users are not perfect at this task, as the given images of objects are not segmented and thus it can sometimes be confusing as to what object the image is supposed to be capturing, or the image only shows a partial/low-quality view of the object. In addition, the object usage being asked for in the language command can occasionally be unconventional such as using a "spoon" to "cut," and thus might not be obvious to the average AMT worker who is spending very little time on each task. Our models' performances are not as good as the human baseline but not far apart. Furthermore, the imperfect human performance proves how difficult of a task this is and how impressive our models' results are.

2.4.2.3 YCB Object Set

Finally, we run an evaluation to test whether the proposed model can perform natural language object retrieval on objects commonly seen and interacted with by real robots. For this evaluation, we test our best model on images of objects from the YCB Object and Model Set [12]. The YCB object set is designed for benchmarking robotic manipulation and consists of objects of daily life with different shapes, sizes, textures, etc. We did not use the YCB object set as our training image set because it has a much smaller number of object classes and only 1 instance per object class in comparison to ImageNet's 1000 object classes and 50 images per class.

Of the 65 object classes with corresponding RGB images in the YCB dataset, we select 33

object classes to test our model on, excluding classes our model has seen during training and picking only one class in the case of identical objects of differing sizes such as “S clamp," “M clamp," “L clamp," “XL clamp." Each object class in the YCB dataset is represented by one object instance, with corresponding RGB images of the object instance from multiple camera angles. We represent each selected object class by a single front-facing image of the object, taken from the YCB dataset. From the 50 verbs our model was trained on, we select 11 verbs (shown in Table 2.13) that are most compatible with the 33 YCB objects and annotate valid verb-object pairings among the selected objects and verbs. This resulted in 64 verb-object pairs. Examples of the annotated verb-object pairs are shown in Table 2.14. Natural language commands containing only the verbs were generated using templates (listed under the “Verb only" row in Table 2.2), and retrieval examples consisting of sets of five images paired with language commands were randomly generated from the annotated verb-object pairs.

Our best model, without being retrained on images from the YCB dataset, was tested 5 times and achieved average retrieval accuracies of 54.7% and 71.9% with standard errors of 1.99% and 2.40% for top-1 and top-2, respectively. The model also achieved a mean reciprocal rank of 71.8% with a standard error of 1.50%, as reported in Table 2.12. Although these results are far from perfect, they still demonstrate generalization on a dataset with a different distribution from the model’s training data. In addition, these results would enable the robot to significantly reduce its search space from all the candidate objects, and employ strategies such as question asking to further disambiguate and retrieve the correct object.

Notably, our model correctly identified a fork, a spoon, and scissors as objects that can be used to cut, while only having seen knife-like object classes such as hatchets paired with the verb “cut" in its training data. In addition, our model selected a chips can and mustard bottle when asked for something to “eat," which in retrospect are very reasonable pairings that were mistakenly left out of our verb-object pair annotations.

2.4.3 Robot Demonstrations

We implement our trained model on a KUKA LBR iiwa robot arm with a Robotiq 3-finger adaptive gripper. We pass a natural language command into our model along with manually

TABLE 2.14: Example annotated verb-object pairs for the YCB set

construct – power drill	eat – apple	play – tennis ball
contain – chips can	grow – pear	rotate – adjustable wrench
contain – windex bottle	hit – spoon	serve – bowl
cut – scissors	open – padlock	write – large marker

segmented RGB images of objects in the scene, captured by an Intel RealSense camera. The robot then grasps the observed object with the highest cosine similarity in embedding space to the language command. We use object classes that our model has not seen during training for all the demonstrations.

We test our robot on four object retrieval tasks. Images capturing the tasks are shown in Figure 2.7. The robot correctly selected the T-shirt for the task of "Give me something to wear." When asked to "Give me an item that can write," it was able to pick out the marker from other distracting objects that are also partly red and have slim bodies. Next, it accurately identified the pear and chips can as the top two items that would satisfy "Hand me something to eat." This is the only test case with more than one possible correct answer. Finally, when asked for "An object to contain," the robot selected the empty Minion-shaped bottle. Video recordings of the robot demonstrations can be found online[3].

We mostly use YCB objects for the demonstrations with the exception of the Minion-shaped bottle in the last case, which was to test our model on an odd-looking object. While these are common objects seen in our daily life, Mask R-CNN [34], the state-of-the-art method for object segmentation and classification, was unable to correctly segment and classify most of them. This is because the objects do not belong to MSCOCO [54], the dataset Mask R-CNN was trained on[4]. The objects that are not part of the 91 object types in COCO are: T-shirt, pear, chips can, marker, clamp, lock, and of course Minion bottle. In addition to these objects, Mask R-CNN was also unable to detect the soccer ball, despite having been trained on sports balls. With such results, relying on accurate classification of objects and querying of an external knowledge base for valid verb-object pairs to select the object that satisfies the language

[3] https://youtu.be/WMAdGhMmXEQ

[4] We did not use the COCO dataset as our training image set as it has a much smaller number of object classes in comparison to ImageNet's 1000 object classes.

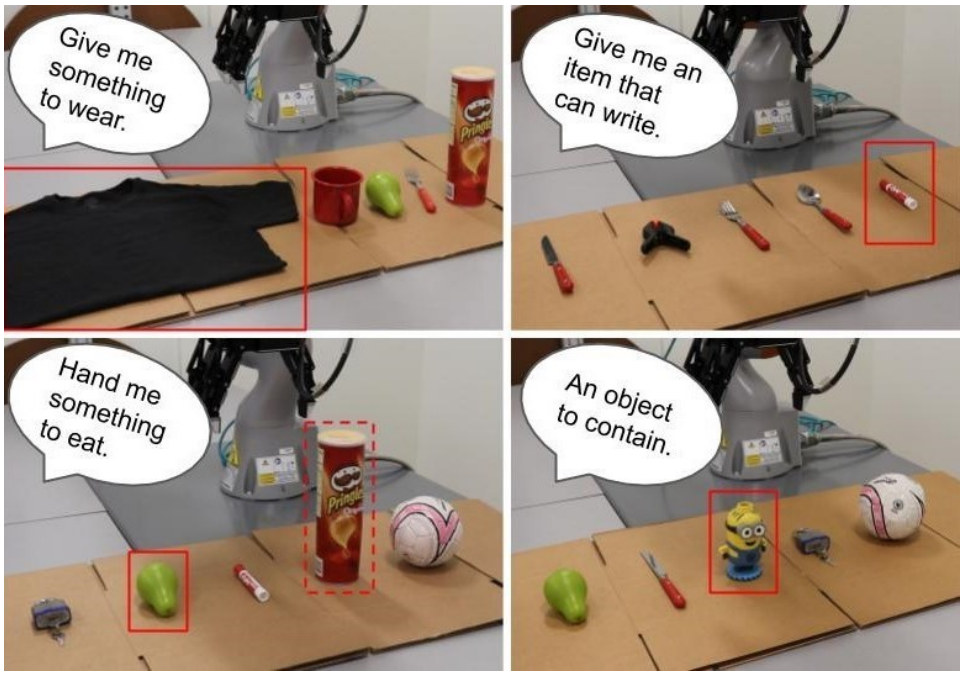


FIGURE 2.7: Natural language object retrieval tasks demonstrated on our robot. The given language commands are: "Give me something to wear" (top left), "Give me an item that can write" (top right), "Hand me something to eat" (bottom left), and "An object to contain" (bottom right). Solid boxes denote the robot's top choice for each task. The dashed-line box denotes the robot's top second choice. Our robot retrieved the correct object for each task.

command does not work in these cases. In contrast, our model was able to select the correct object based on the command without needing to explicitly classify the candidate objects or having seen their object classes.

2.5 Conclusion

Understanding open-ended natural language commands is a challenging but important problem. We address a sliver of the problem by focusing on object retrieval based on descriptions of the object's usage. We propose an object retrieval model that learns from contextual information from both language and vision to generalize to unseen object classes and unknown nouns. Given natural language commands, our model correctly selects objects out of sets of five candidates, and achieves a mean reciprocal rank of 77.4% on a held-out set of unseen ImageNet object classes and 69.1% on unseen object classes *and* unknown nouns. Our model also

achieves a mean reciprocal rank of 71.8% on unseen YCB object classes. We demonstrate our model on a KUKA LBR iiwa robot arm, enabling the robot to retrieve objects based on natural language descriptions of their usage. Along with our model, we also present a newly created dataset of 655 verb-object pairs denoting object usage over 50 verbs and 216 object classes, as well as the methods used to create this dataset. To the best of our knowledge, this is the first dataset built to perform this task, and could potentially be used for a range of object retrieval tasks.

Our model currently allows us to reduce the problem of task-based object retrieval to an attribute classification problem. There is room to improve our model's performance with a different image encoder such as EfficientNet [85] and pretrained word embeddings. Image color augmentation techniques and the use of depth information may also help the model learn to ignore extraneous image properties and achieve better generalization. Furthermore, a much richer model would perform explicit inference to determine the desired object from oblique language commands. Incorporating dialogue into this framework to perform inference can be a way to incorporate human preference more directly and provide a more intuitive interface.

Chapter 3

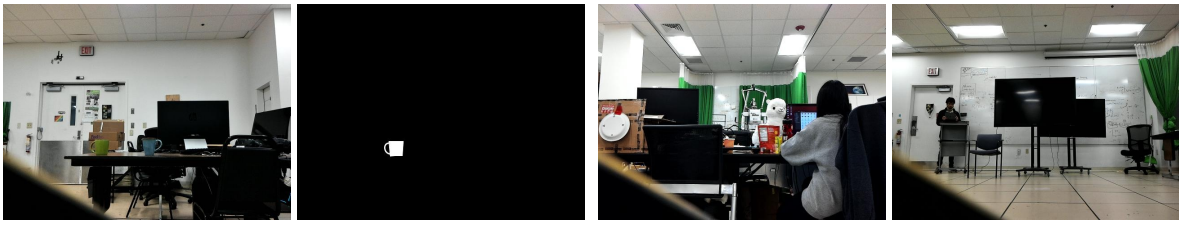
Language-Conditioned Observation Models for Visual Object Search

This work integrates language-conditioned visual models within a model-based decision-theoretic framework for effective robotic object search with complex natural language specifications.

3.1 Introduction

Object search is a challenging task because the robot has incomplete knowledge of the environment, limited field of view, and noisy sensors. When asked to find an object in an environment, the robot must first infer the desired object from the language instruction, then efficiently move around the space to look for the object. Most images captured by the robot in this process will not contain the target object. Furthermore, even when the object is in the robot's field of view, it might not be detected due to occlusion, sensor noise, the viewing angle, the object's distance from the robot, etc.

There have been many previous works on improving the efficiency and accuracy of robot object search by using prior semantic knowledge [47], active visual search [5, 6], object manipulation [52, 98, 19], and belief factorization for multi-object search [89, 103]. However, these works have often assumed that the target objects are specified with very simple language (such as "cup" for the object's class), and thus cannot fully utilize more complex language descriptions (such as "white cup") to avoid exhaustively searching over similar object instances



(a) A scene with the ground truth segmentation mask for the target object ("the green mug on the left").

(b) Most images captured by the robot do not contain the target object.

FIGURE 3.1: Our system takes as input a natural language description of the target object, and constructs a detector for that object based on the description. It addresses the problem that most images captured by a robot when searching for an object do not contain that object by incorporating a modified training process and using the confidence score of the detector in a POMDP model for object search.

in the environment. Furthermore, the robot is usually assumed to have a fixed-accuracy object detector [47, 6, 89, 103] or that detection noise only comes from occlusion [52, 98]. This is challenging to scale as new occlusion models and detectors have to be made for new objects. The computer vision community has developed deep learning models that can detect objects with high accuracy [34, 75], even given complex language descriptions of the desired object such as "white cup on the left" [38]. However, these models often assume that the object is somewhere in the input image and must be localized within that image. In contrast, when searching for objects, most images captured by the robot will not contain the object being searched for (Figure 3.1).

Our work addresses these problems by embedding a deep-learned object detector within a Partially Observable Markov Decision Process (POMDP) [13]. Our approach takes as input a language description of an object, and uses it to condition a camera-based observation model that is used to plan object-search actions and update the agent's belief about the object's pose. To achieve this, we modify the training process for the detector from Hu et al. [38] to handle the large number of images which do not contain the target object, and incorporate the detector's confidence scores into the POMDP belief updates. This allows us to handle complex language descriptions and search for objects more efficiently in household environments by reasoning dynamically. Our contributions are five-fold:

1) An experimental analysis on confidence scores outputted from language-conditioned visual segmentation models as a proxy for different sources of observation noise.

2) A novel class of visual observation models (Language-Conditioned Observation Models – LCOMs) whose detections and parameters are conditioned on natural language.

3) A novel decision making framework for object search that leverages LCOMs to use natural language to account for scene-dependent detection accuracy when estimating state uncertainty for planning.

4) A set of experiments on simulated robot hardware that compare the performance of planning models using LCOMs against fixed-noise sensor models on the object search task.

5) A demonstration of our method on a Boston Dynamics Spot robot [1], which enables Spot to handle complex natural language object descriptions and efficiently find objects in a room-scale environment, without using fiducial markers.

3.2 Related Work

Related work for robot object search generally falls into one of two categories: *model-based* and *end-to-end policy learning*. Model-based approaches separate state estimation and planning to leverage probabilistic inference, whereas model-free approaches leverage deep neural networks to learn a policy end-to-end.

There is a collection of works that employ deep learning for end-to-end visual and object search [24, 97, 68]. Our work differs from these in that we perform model-based planning to leverage our known dynamics models. Model-based planning has the potential to generalize better to new environments and systems with less training data because we encode a model of the robot's sensor and actuation capabilities, and only use deep learning for visual processing.

POMDPs [44] are a framework for sequential decision making under uncertainty frequently used for object search problems. Li et al. [52] and Xiao et al. [98] treat object search in clutter as a POMDP that can be efficiently solved by using approximate online planners and constraining planning samples based on spatial constraints and conditioning action selection on the current belief state, respectively. However, their observation models are only based on occlusion statistics calculated from object region overlap. Our proposed observation model can instead account for errors not solely derived from occlusion by conditioning on complex language. Danielczuk et al. [19] train a deep learning model to segment colored masks for

objects in a pile from RGB-D images and score each mask on whether it belongs to the target object. They, however, use a fixed object priority policy for action selection and assume a fixed sensor pose, while we focus on planning how to explore a space for the purpose of object search by leveraging an active sensor.

Aydemir et al. [6] frame the object search problem as active visual search and calculate candidate viewpoints based on a probability distribution over the search region, which is informed by prior knowledge of correspondences between objects and semantic room categories. However, they do not account for sensor error and assume the object to be detected if it is in the robot's field of view. Atanasov et al. [5] plan a sequence of sensor views to effectively estimate an object's orientation. These approaches are similar to our work in that they account for realistic sensor model errors, but unlike our work they do not use a general camera-based object detector.

Wandzel et al. [89] introduce Object-Oriented POMDP (OO-POMDP) to factorize the robot's belief into independent object distributions, enabling the size of the belief to scale linearly in the number of objects, and employ it for efficient multi-object search. Zheng et al. [103] extend OO-POMDP for efficient multi-object search in 3D space. Both of these works assume simple language inputs and fixed-accuracy object detectors. Our work builds on these frameworks but instead explores using a deep-learned detector that takes as input a natural language phrase and camera image to create an object detector that models varying levels of accuracy.

The computer vision community has developed deep learning models trained on object segmentation datasets that can detect objects with high accuracy [34, 75, 38]. The models self-supervisedly learned to output confidence scores along with their detection results. The output confidence scores do a good job of reflecting the models' detection accuracy, which dynamically changes depending on the input images. We build on the model developed by Hu et al. [38] for our object detector because it is trained to handle referring expressions—complex noun phrases to describe objects.

3.3 Preliminaries

POMDPs are a framework for modeling sequential decision making problems where the environment is not fully observable. Formally, a POMDP can be defined as a tuple $< S, A, \Omega, T, O, R >$, where S, A, Ω denote the state, action, and observation spaces of the problem, respectively. After the agent takes action $a \in A$, the environment state transitions from $s \in S$ to $s' \in S$ following the transitional probability distribution $T(s, a, s') = p(s'|s, a)$. As a result of its action and the state transition, the agent receives an observation $z \in \Omega$ following the observational probability distribution $O(s', a, z) = p(z|s', a)$, and a real-valued reward $R(s, a)$.

Because the environment is partially observable, the agent does not have full knowledge of the current state s and instead maintains a *belief state* b which is a probability distribution over the states in S. The agent starts with an initial belief b_0 and updates its belief after taking an action and receiving an observation (where η is the normalizing constant): $b'(s') = \eta O(s', a, z) \sum_{s \in S} T(s, a, s') b(s)$.

A policy π is a mapping from belief states to actions. The agent's task is to find a policy that maximizes the expected sum of discounted rewards given an initial belief:

$$V^{\pi}(b_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| a_t = \pi(b_t)\right]$$

where the discount factor γ determines the impact of future rewards on current decision making.

While many problems can be modeled by POMDPs, they are typically computationally intractable for exact planning in large domains [55]. To address the planning complexity, we use the sampling-based planner PO-UCT [80], which uses Monte-Carlo tree search with upper confidence bounds to select an action by estimating the best state-action values using rollouts conditioned on states sampled from the current belief state, and then performs an exact belief update each time step based on the incoming observation and performed action. PO-UCT has successfully been used in robotic object-search settings [89, 103]. However, we note that our contribution is not dependent on the specific method of planning and state estimation, and LCOMs would be useful in any approach that requires a model of the observation probability

distribution.

3.4 Object Search Formulation

We model object search as a POMDP with an observation model corresponding to a deep-learned object detector.

3.4.1 Planning Framework

To model the object search problem, we assume access to an occupancy-grid map, M, which is an $m \times n$ grid that marks locations as either empty or occupied, and is used for defining the space of positions and directions in the environment. We assume an object is completely contained within one of the grid cells in the map. Our main contribution is the novel language-conditioned observation model (LCOM), which modifies the observation model dynamically based on the results of the deep-learned object detector, and which we describe in detail in Section 3.4.2. Formally, we define the object search POMDP problem as a 10 tuple: $< o_d, L, S, A, T, R, \gamma, \Omega, h_L, O >$

1. o_d: is a desired object that exists in the environment (not including the robot). The desired object o_d has a 2D position attribute $(x_{o_d}, y_{o_d}) = o_d$, representing its discrete position in the occupancy-grid map M. The desired object is used to define the reward function.

2. L: is a string of words representing the natural language command given by the human, such as "The white cup on the table." L is only used to condition the visual observation model and transform raw images into our fan-shaped sensor model. We defer more details to Section 3.4.2. In this work we assume L to be given at the start and remain constant throughout the interaction, and defer handling dynamical language to future work.

3. S: is a set of states, where each state $s \in S$ is a 2 dimensional vector $(o_d, r) = s$, where $r = (r_x, r_y, r_o)$ is a 2D position and discrete orientation (NORTH, EAST, SOUTH, WEST)

for the robot in M. We assume r is fully observable and o_d is only partially observable, yielding a mixed-observable state. This assumption is equivalent to assuming our robot is equipped with a LIDAR sensor and has previously run SLAM [73] and localized itself within that map, but does not know where the desired object is currently located.

4. A: is a set of actions the robot can execute to move around the map, observe object locations, and declare the desired object as found. Specifically, we have three types of parameterized actions:

   (a) *Move*(*DIR*): points the robot in direction *DIR* and moves it one grid in that direction, with *DIR* being either NORTH, EAST, SOUTH, WEST.

   (b) *Look*: has the robot execute a look action from its current position and orientation (r_x, r_y, r_o).

   (c) $Find(X, Y)$: has the robot attempt to find the desired object o_d at grid cell (X, Y). If o_d is at (X, Y), the action will mark the object as found and terminate the episode.

5. T: is a deterministic transition function, where *Move* actions transition the robot to different states by changing its position and orientation (r_x, r_y, r_o). *Find* actions can transition the robot to a terminal state after finding the desired object.

6. R: is a reward function, where all *Move* actions receive -2 reward each, *Look* receives -1 reward, and $Find(X, Y)$ receives a 1000 reward when done at the location of the desired object $(X, Y) == (x_{o_d}, y_{o_d})$ and -1000 otherwise.

7. γ: is the discount factor, which we set to 0.9.

8. Ω: is the set of observations from our sensor, where each $\omega \in \Omega$ is a pair of RGB and depth images.

9. h_L: $\Omega \rightarrow z^s, c^s$ is a language-conditioned observation-mapping function that transforms raw images into observations from the same fan-shaped sensor model described in Wandzel et al. [89] and confidence scores for each object detection. If the desired object o_d is not detected by the sensor, $z^s = NULL$. Otherwise, z^s is the location (X, Y)

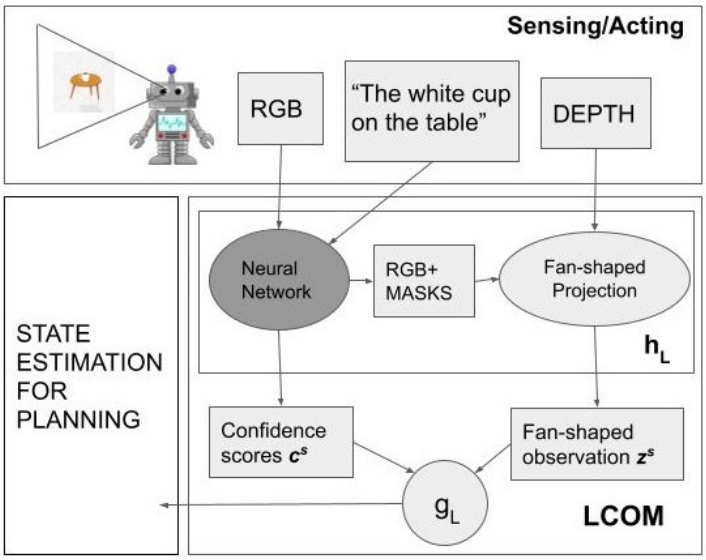


FIGURE 3.2: **LCOM Overview**: The robot receives an RGB-D image and language description of the object. RGB-D and language go into h_L, which produces language-conditioned confidence scores c^s for our fan-shaped detected observations z^s. The confidence scores are then transformed by g_L into a noise model for the detected observations, which is used to update the belief about the object's location via state estimation. Ovals are algorithms, and rectangles are data. The shaded oval is learned.

where o_d is detected in the discretized fan-shaped region V. c^s represents the object-specific observation confidence score. We describe how h_L is used for LCOMs in Section 3.4.2, and our particular instantiation of h_L for our experiments in Section 3.4.3.

10. O: is the Language-Conditioned Observation Model (LCOM), which assigns probabilities to observations z_t based on the current state s_t, action a_t, and natural language command L. The *Move* actions always produce the $NULL$ observation, the *Look* action produces noisy fan-shaped measurements conditioned on the language, and the $Find(X, Y)$ action always produces the $NULL$ observation except when $(X, Y) == (x_{o_d}, y_{o_d})$. We discuss the observation model in more detail in the following subsection.

3.4.2 Language-Conditioned Observation Model (LCOM)

Figure 3.2 presents an overview of LCOM. When we receive an RGB-D sensor observation, ω, we can transform it into our fan-shaped sensor observation z^s and associated confidence scores c^s by using the language-conditioned observation mapper $h_L(\omega) = z^s, c^s$. LCOMs

are independent of any particular instantiation of h_L as long as they satisfy the functional definition described in Section 3.4.1, and for the rest of this section's discussion we treat h_L as a black box function. In our experiments, we instantiate h_L using a deep neural network.

We treat z^s as having a probability of being drawn from three mutually exclusive and exhaustive events: a true positive (A), a false positive (B), or a true or false negative (C). More formally, let A be when z^s is from the desired object o_d and $z^s \in V$, B be when $z^s \in V$ but z^s comes from other sources besides o_d, and C be when $z^s = NULL$. We assume the $Find(X, Y)$ action always give perfect information about the potential object at location (X, Y) (*i.e.,* observations resulting from *Find* are not language-conditioned). In simulation this is reflected by knowing the ground truth state, and in real-life this can be reflected by asking a human to verify the selected location. For the *Look* action, we parameterize the probability of each of the events and the noise model for the observation conditioned on that event based on the associated confidence score c^s, and decompose the observation model $p(z^s|s, a)$ into:

$$p(z^s|s, a, c^s) = \sum_{e \in \{A,B,C\}} p(z^s|e, s, a, c^s) p(e|s, a, c^s) \tag{3.1}$$

If event A occurs, the observation is distributed normally with μ being the true object position: $p(z^s|A, s, a, c^s) = \eta' Norm(z^s|\mu, \Sigma)$. η' is the normalization factor, and the covariance matrix is defined by $\Sigma = \sigma \mathbf{I}^{2\times 2}$. If event B_i occurs, the observation is distributed uniformly within the sensor region: $p(z^s|B, s, a, c^s) = \frac{1}{|V|}$. If event C occurs, the null observation has nearly 1 probability while any other observation has nearly 0 probability, which we implement with additive smoothing.

Similar to Wandzel et al. [89], we define the probability of the events as $p(A|s, a) = \alpha$, $p(B|s, a) = \beta$, $p(C|s, a) = \gamma$, where $\alpha + \beta + \gamma = 1$. The probability of these events are conditioned on whether or not the desired object o_d is in the fan-shaped sensing region V, which is defined as:

$$(\alpha, \beta, \gamma) = \begin{cases} (\epsilon_{TPR}, 0, 1 - \epsilon_{TPR}) & \text{if } o_d \text{ is in } V \\ (0, 1 - \epsilon_{TNR}, \epsilon_{TNR}) & \text{if } o_d \text{ is not in } V \end{cases} \tag{3.2}$$

where ϵ_{TPR} represents the sensor's true positive rate, and ϵ_{TNR} represents its true negative rate.

Together σ, ϵ_{TPR}, and ϵ_{TNR} define the sensor's overall accuracy. To implement the function g_L, which transforms the confidence scores to the sensor noise in the observation model, we map the continuous value of c^s to a discrete range of hyper-parameter values that represent high-confidence and low-confidence for each setting, respectively. In our experiments, we map ϵ_{TPR} to 0.7 and σ to 0.6 when $c^s \geq 1$, and ϵ_{TPR} to 0.5 and σ to 1 otherwise. These numbers reflect that when the detector's confidence is high, the true positive rate should be high and the uncertainty over the observed position of the object should be low.

We note that LCOMs depend on visual input to detect potential objects in the image and report confidence scores that are used to define the sensor noise in the observation model. During state estimation with real-robot hardware, acquiring visual input is straightforwardly done by capturing images with the robot's camera. During planning, however, acquiring visual input may be challenging because it requires synthesizing novel images based on the pose of the robot and potential location of the target object. For computational efficiency, when performing visual object search in our experiments, we only use LCOMs for updating the agent's belief during state estimation, and use a fixed observation model similar to Wandzel et al. [89] during planning based on the 2D geometries of the known occupancy-grid map M. Integrating different 3D scene representations into the planning module is straightforward but orthogonal to our contribution, so we defer this investigation to future work.

3.4.3 Object Detector

We build upon the model developed by Hu et al. [38] for our object detector as it can handle complex noun phrases to describe objects. The model takes in a referring expression and RGB image and outputs scores for every pixel in the image, which are then binarized and returned as the predicted segmentation mask for the image region described by the language. The loss function used for training is the average pixelwise loss: $Loss = \frac{1}{WH}\sum_{i=1}^{W}\sum_{j=1}^{H} L(v_{ij}, M_{ij})$. W and H are image width and height, v_{ij} is the pixel's score, and M_{ij} is the binary ground-truth label at pixel (i, j).

The original model by Hu et al. [38] was trained on the ReferIt dataset [45] which mostly contains outdoor images, whereas we are interested in detecting indoor household objects. We, therefore, additionally trained the model on the RefCOCO dataset [45] which contains referring expression annotations for segmented objects from the COCO dataset of common objects in context [54]. Furthermore, the original model was primarily trained on positive examples such that most images contained the target object, and the model only had to learn to identify where the object was in the image. In contrast, when using a model like this for object search, most images fed to the model will not contain the referenced object. Thus filtering a large number of true negatives without missing the rare true positive is key to good performance in search tasks. We augmented the model's training data with negative examples where the object described by the referring expression does not appear in the image, and thus the model should return an empty segmentation mask. This is to adapt the model to our object search setting where the desired object is usually not in the robot's field of view. Our model, trained on the augmented data, achieved a true negative rate of **0.918**, a significant improvement over the original model's true negative rate of **0.124**.

We now describe our instantiation of h_L for our experiments based on the deep learning segmentation model. The model takes in the RGB image and language L and outputs a segmentation mask—a binary image which identifies pixels that are part of the target object described by L. If the mask is empty, the model did not detect the object and $z^s = NULL$. Otherwise, we take the average of the depth value at each pixel in the mask as well as the coordinates of the mask's center point and project it into a location (X, Y) in the robot's fan-shaped sensing region (*i.e.,* fan-shaped projection) and return (X, Y) as z^s. We also retain the model's original output score for each pixel, which we average over all pixels in the mask and use as the confidence value c^s for the detection. We note that the scores were not specifically trained for this task.

FIGURE 3.3: **Simulated Scenes**: example images of the AI2-THOR scenes used in our experiments. The scene categories are: kitchen *(top left)*, living room *(top right)*, bedroom *(bottom left)*, and bathroom *(bottom right)*.

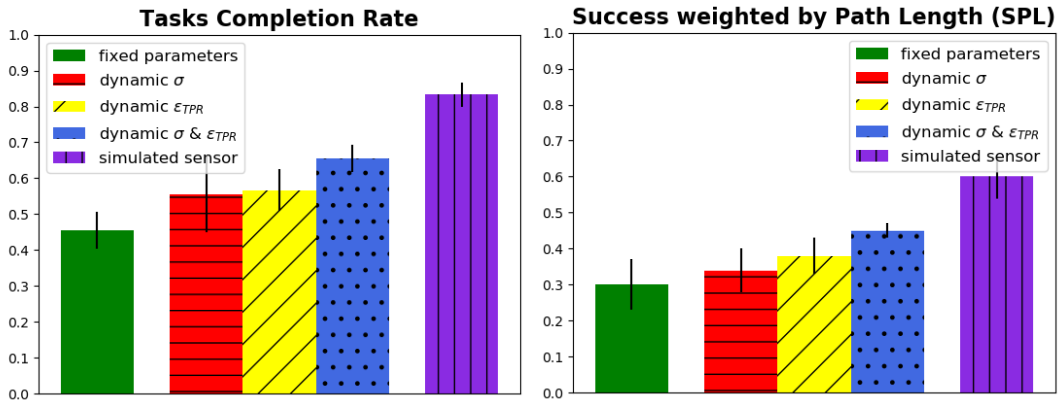
3.5 Experiments and Results

Our aim is to test the hypothesis that language-conditioned observation models combined with POMDPs can increase a robot's speed and accuracy in finding objects in complex environments. We evaluated our system both in a variety of simulation environments and on a real physical robot.

3.5.1 Simulation Results

We use scenes from the AI2-THOR simulator [48] to conduct our experiments. AI2-THOR consists of 120 near photo-realistic 3D scenes spanning four different categories: kitchen, living room, bedroom, bathroom. We select a subset of 15 scenes with 30 target objects (for an average of 2 objects/scene) for our experiments. Figure 3.3 shows images of the scenes used in our experiments. The average size of a scene is 4×4 meters, which we discretize into a 16×16 cell grid map with each cell being 0.25×0.25 meters.

We build upon the POMDP implementation by Zheng and Tellex [102] in the pomdp_py



(a) (b)

FIGURE 3.4: **Simulation Results**: Task completion rates and success weighted by path lengths for a simulated sensor and deep-learned sensor with static/dynamic observation models.

library. We modeled the POMDP as having no prior knowledge of the target object's location, thus it had a uniform initial belief state over all possible object locations. We used a planning depth of 3, exploration constant of 10000, planning time of 10 seconds for each action, and gave the agent a maximum time of 5 minutes and 10 *Find* actions to complete each object search task. We generated simple natural language descriptions of the objects in our experiments as input to the agent.

Results appear in Figure 3.4. We present both the task completion rate—the percentage of time the robot successfully finds the object, and success weighted by normalized inverse path length (SPL) [4]. SPL is calculated as: $\frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{max(p_i,l_i)}$ where N is the total number of tasks, l_i is the shortest path from the agent to the goal for task i, p_i is the path the agent actually took for the task, and S_i is a binary indicator of success in the task. For our experiments, p_i is the number of actions the agent actually took to search for the object, and l_i is the lowest number of actions needed to find the object. If the agent achieves a higher task completion rate but took more steps overall to find the objects, it will have a lower increase in its SPL. We collected l_i by performing planning with a perfect sensor with no noise. The perfect sensor was able to find all 30 objects at an average of 7.8 actions per object search task.

Each different version of our model was tested 3 times and we report the average and standard error in their performance. We present results for fixed optimal values of the sensor

parameters computed from the scenes in our dataset. Our deep learning model achieved a true positive rate (TPR) of 0.581, a true negative rate (TNR) of 0.918, and a covariance of 0.827 for the normal distribution over the desired object's position. We then show results with σ, ϵ_{TPR}, and both σ & ϵ_{TPR} values set dynamically based on the deep-learned detector's output confidence score As the sensor's TNR is already high, we decide to keep ϵ_{TNR} fixed. Lastly, we show the performance with a simulated sensor whose noise model perfectly matches the model used for planning by the POMDP.

As expected, the performance for the simulated sensor is the best. This is because the sensor observations are being generated from ground truth with exact noise models. This provides an upper bound on our system's performance, and also indicates that if we used a more realistic sensor model, our system has the potential to perform even better. In particular, the simulated sensor will sample multiple images with the same viewpoint independently, which is not true for the deep-learned detector. All versions of our system with a dynamic observation model outperform the static version. In addition, the version with dynamical σ & ϵ_{TPR} achieved a significantly higher average task completion rate and SPL than the static version (from **0.46** to **0.66**, and from **0.30** to **0.45**, respectively). On average, this version took **10.1** actions and **104** seconds to find each object, compared to the **11.5** actions and **118** seconds taken by the static version. Overall, these results demonstrate that using a dynamic observation model significantly improves the ability of our system to find objects quickly and efficiently in realistic environments.

3.5.2 Real-World Demonstration

We provide a real-world demonstration on the Boston Dynamics Spot robot. The robot takes as input an occupancy-grid map of the environment and a typed natural language phrase describing the desired object. RGB and Depth images are taken from two separate cameras in the robot's hand, and pixel correspondence between the two images is computed using both cameras' intrinsic and extrinsic matrices. Spot moves through the environment by taking steps that are 0.6 meters (one grid cell) in length, and all decisions are driven by the POMDP until it finds the object. Scenes from our demonstration and the corresponding belief updates

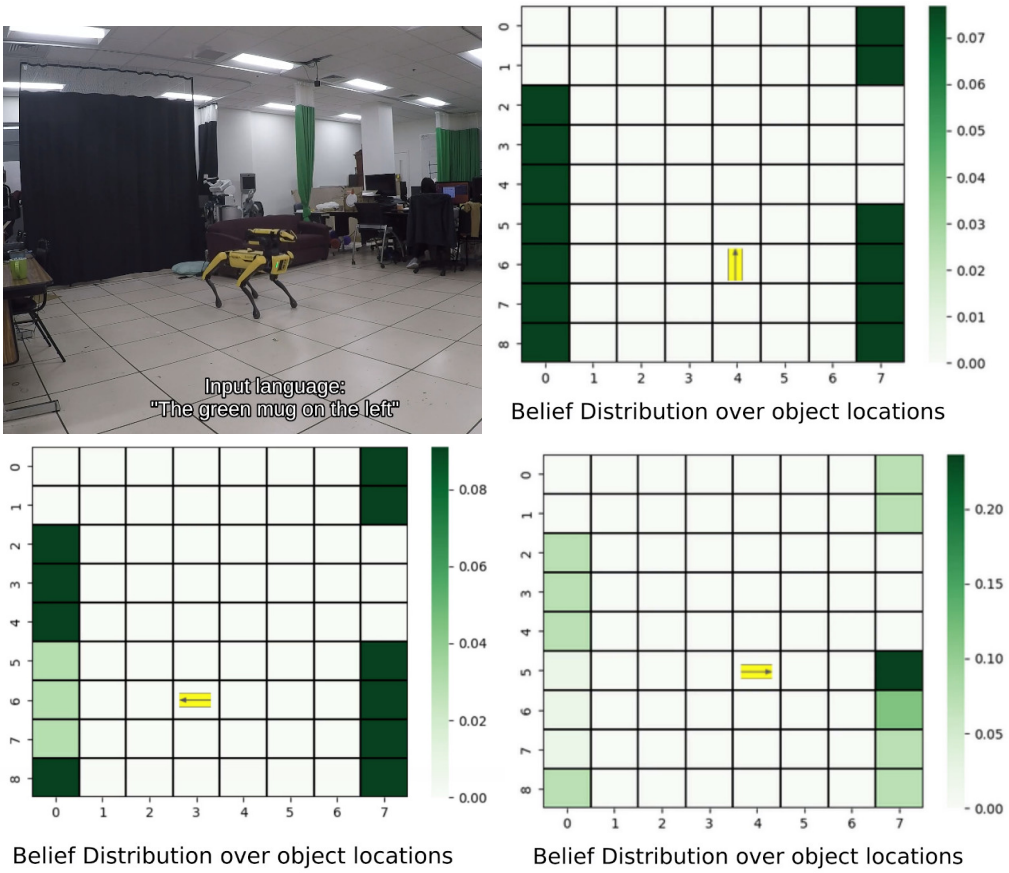


FIGURE 3.5: **Real Robot Demonstration**: sample images from our real robot experiments with the Spot using LCOMs to find an object. *top left:* the robot is turned on and tasked with finding "the green mug on the left." *top right:* the robot's uniform initial belief about the target object's location. *bottom left:* the robot moves and looks at a part of the room where the object is not located, and updates its belief that the object is most likely somewhere else. *bottom right:* the robot moves and looks where the object is actually located, and after updating its belief has maximum likelihood estimate at the target object's true location.

from using LCOMs with real robot hardware are shown in Figure 3.5. Full video footage of the robot executing the task, the incoming sensor data, and the LCOM outputs is available online[1]. The robot was asked to find "the green mug on the left" and successfully did so in 8 actions, where the planning and execution of each action took 10 seconds. This demonstration shows our system runs on a real-world platform in a realistically sized environment, computes a policy and observations efficiently, and enables a robot to efficiently search for and find objects.

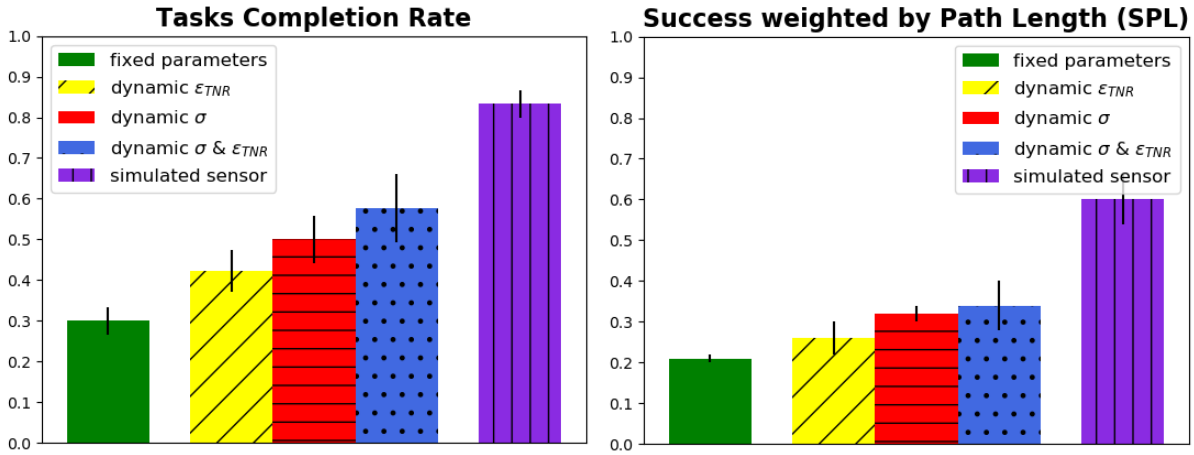
3.5.3 ViLD Experiments

Given the recent success in open-vocabulary image classification/object detection powered by CLIP [74], we swapped out our trained object detector with ViLD [29], an object detector trained via vision and language knowledge distillation, for our object search experiments. ViLD takes in natural language expressions and an RGB image, proposes regions of interest within the image, computes visual embeddings for the regions, and calculates the dot product (score) between the visual embeddings and text embeddings generated by CLIP. It then returns the highest scoring image regions that correspond to the input natural language.

For each object in our experiment, we pass a simple natural language description of the object into ViLD along with the RGB image taken by the robot, and take as output the segmentation mask associated with the highest scoring image region. The mask has the same size as the input RGB image, and value 1 for every pixel within the proposed region and 0 otherwise. If ViLD does not find an image region corresponding to the input language, the segmentation mask is empty and the observation $z^s = NULL$. Otherwise, we take the average of the depth value at each pixel in the mask as well as the coordinates of the mask's center point and project it into a location (X, Y) in the robot's fan-shaped sensing region and return (X, Y) as z^s. We also retain the image region's score as the confidence score c^s.

Without fine-tuning, ViLD achieved a true positive rate (TPR) of 0.976, a true negative rate (TNR) of 0.118, and a covariance of 1.825 for the normal distribution over the desired object's position on our AI2-THOR dataset. Similar to other object segmentation methods, ViLD tends

[1] https://youtu.be/3Z4XQUQXCsY



(a) (b)

FIGURE 3.6: **ViLD Simulation Results**: Task completion rates and success weighted by path lengths for a simulated sensor and ViLD with static/dynamic observation models.

to return a non-empty segmentation mask even when the queried object is not in the input image.

Experiment results are shown in Figure 3.6. The settings are the same as those described in Section 3.5.1. We present results for fixed values of the ViLD sensor parameters. Next, we show results with σ, ϵ_{TNR}, and both σ & ϵ_{TNR} values set dynamically based on the output confidence score c^s. We map ϵ_{TNR} to 0.1 and σ to 1.0 when $c^s \geq 0.25$, and ϵ_{TNR} to 0.3 and σ to 2.0 otherwise. As ViLD's TPR is already high, we decide to keep ϵ_{TPR} fixed. Each different version was tested 3 times and we report the average and standard error in their performance.

The simulated sensor's performance is still the upper bound on the object search task. ViLD's performance trails behind our object detector which is fine-tuned for the task. However, all versions of our system with a dynamic observation model still significantly outperform the static version. The version with dynamical σ & ϵ_{TNR} achieved an average task completion rate of **0.578** and SPL of **0.34**, compared to the **0.3** and **0.21** achieved by the static version. These results demonstrate that our system works seamlessly with different object detectors and using dynamic observation models improves object search performance.

3.6 Conclusion

Our contribution is a novel observation model that uses the detector's confidence score to better model the detection accuracy. This enables us to handle complex language descriptions of objects and perform object search with a real object detector in realistic environments. In addition, our method can be easily adapted to new environments without having to relearn the observation model's parameters.

Our model only considers 2D space. In future work, we plan to extend to 3D models, building on Zheng et al. [104] to model the 3D structure of objects. This extension will enable the robot to reason about different 3D viewpoints and predict the structure of a partially observed object to gather more views to identify and localize it. We also plan to specifically train the detector's output confidence scores to represent its detection accuracy.

Additionally, our model cannot reason about the likelihood of different views of the same object to improve its detection/localization of that object. Our current observation model assumes that each observation is independent, so if the robot observes the same scene from the same viewpoint, it will become more and more certain whether the object is present or not. However, in practice, when viewing an image from the same viewpoint, a deep-learned detector will give the same results; the observations are not independent samples. In the future, we could address this problem by creating a new observation model based on inverse graphics and an expected 3D model of the object appearance, enabling the robot to predict the next best view to maximally reduce its uncertainty about the object's location.

Furthermore, we focus on language descriptions of the desired object to generate the object detector and observations. More complex language instructions that provide information about the location of the object such as "look in the kitchen" or "the object is to your right" can be incorporated by directly updating the agent's belief about the object's pose.

Overall we see object search as a central problem for human-robot interaction, as finding, localizing, and then grasping an object is a first step for almost anything a person wants the robot to do in the physical world. Embedding object search as a sub-component of a more sophisticated dialog system can enable the robot to engage in collaborative dialog with a human partner to interpret complex natural language commands, find and manipulate objects

being referenced, and fluidly collaborate with a person to meet their needs.

Chapter 4

Find It Like a Dog: Using Gesture to Improve Robot Object Search

In this work, we seek to understand pointing gestures and enable robots to utilize both natural language and gesture information for object search.

4.1 Introduction

People need to communicate locations for a wide variety of tasks, and often use pointing gestures to do it. When pointing, a person uses their head, body, hand and arm to refer to an object or location in the environment. Using a deictic gesture such as pointing is intuitive for a person and directly communicates spatial information in the form of a 3D vector through space. Existing literature has shown that people can interpret points from others from infancy (e.g., [11]), and are highly accurate at interpreting the specific target of human pointing gestures [95, 9]. Point following is not limited to human beings, other species, in particular dogs, are able to follow human pointing gestures to locate hidden objects [3, 60, 31, 82] with little or no training, and from a very young age (e.g., [10, 76]).

Existing work on robotic following of human pointing gestures has used a variety of methods to obtain the 3D vector through space corresponding to the point. Previous works [91, 92, 69, 21, 18] have demonstrated effective human-robot collaboration on non-search tasks through the incorporation of pointing gestures along with speech to relay task-relevant information to a robot. Such existing approaches rely solely on social feedback and gestures to

FIGURE 4.1: Our system enables a robot to locate objects using information from a person's unscripted gestures. We compare our robot's performance to the performance of the domestic dog.

help identify the target object the human is pointing to, but without considering that objects can be hidden from view from the robot or the human's perspective, or be of different distances away from the person, so that the target of the point is ambiguous depending on how the pointing vector is identified. Prior work in both the robotics and cognitive science communities has used a range of vectors, such as the vector from the person's eyes to their hand [86, 2, 92, 7], the forearm vector [91, 35, 87, 40], as well as other non-pointing vectors such as eye gaze [67, 72, 59]. However, there has been no systematic study that measures which approach most accurately enables a robot to resolve pointing gestures to spatial object locations, or best corresponds to what vector other entities, such as dogs, use to follow points.

Our work addresses this gap by presenting a mathematical framework for incorporating human pointing gestures into robotic object search. We present five algorithms for resolving a pointing gesture to a 3D vector in space, and then transform that vector into a probability map in the physical world using a generative observation model. Using this probability map the robot can incorporate information from the pointing vector to efficiently find objects in the

environment in collaboration with the person. To our knowledge, no previous work has used pointing gestures for giving a robot information for object search.

We evaluate five different approaches for converting information from the human's body pose into a 3D vector in space, including the vector from eye-to-wrist, nose-to-wrist, elbow-to-wrist, shoulder-to-wrist, and the eye gaze vector. To our knowledge, we are the first work to systematically evaluate these different approaches; other works have used these approaches individually but have not directly compared them against each other to see what is most accurate. We compare our approach to the non-human baseline of the domestic dog, which is well-known to be able to follow human pointing gestures [3], and to engage in collaborative object search and other cooperative tasks with humans [31]. Our results demonstrate quantitatively that the vector from the gaze only performs the worst, while our other four candidate vectors display comparable levels of performance at identifying the object the person is pointing at. Finally, we demonstrate an end-to-end object search system running on a real robot, showing that the robot is capable of incorporating information from the person's gesture to find objects.

4.2 Related Work

Different humans may have unique ways of expressing pointing gestures, so it is imperative to understand the motor and perceptual processes used to generate pointing gestures. Past work with infants has found that from an early age, they understand that points are intended to direct another's attention towards an object or location in the environment, and that points are intentional, so that people will not point at things they do not know about and cannot see [81, 53, 96]. But how are points produced and interpreted by adults? Point production and following is ubiquitous in daily life. Under normal pointing conditions (meaning full visual access to the target), pointers tend to use an eye-to-hand vector. When blindfolded, however, pointers gesture with their arm alone [95]. There are also differences in how far the item being indicated is from the two vectors (eye to the hand vs. down the arm), with arm-only points consistently overshooting the target. This error in production is also mirrored with errors in comprehension. While in general, humans are quite accurate at producing points

for others, past work has revealed that there are minor but systemic errors in how the viewer perceives the targets of points [35, 36]. Much of this has to do with errors in perspective taking, with researchers suggesting that the pointer fails to account for the different viewing angle of the viewer. While there has been important work in how people understand and produce points, there has yet to be a systemic investigation of naturalistic point production. Further, we suspect that humans point differently when they point for other humans versus non-human entities such as dogs or robots, but this has not been explored.

Many studies [42, 49, 101, 63] in the field of Computer Vision focus on end-to-end automatic gesture detection. Jaiswal et al. [42] trained a deep convolutional neural network to estimate the direction of finger pointing gestures. The estimation only used the position of the person's elbow and wrist, disregarding many other relevant keypoints on the human body. Nakamura et al. [63] introduced a large-scale dataset and model for pointing recognition and direction estimation. However, their data always include the person's entire body, which a robot might not have access to from its point of view. An unsupervised learning approach [43] has been constructed to model the variation of pointing gestures without having to scale computationally with the number of gestures and objects being pointed at. Such research also distinguishes between arbitrary hand movements and meaningful gestures. Our work employs Google's MediaPipe Pose Landmarker [8], a deep-learned model for human pose estimation, to detect keypoints on the human user's body and explicitly compute 3D pointing vectors.

A number of human-computer and human-robot interaction (HRI) papers discuss how to interpret a pointing gesture. Human-computer interaction works [59, 58] usually require that people wear a headset and use a clicker to get visual feedback, which can be costly and difficult to use. We also want the interaction to be as natural and as comfortable as possible for human users. Nickel and Stiefelhagen [67] characterizes three approaches for estimating pointing direction: the line of sight between head and hand, forearm (elbow-to-wrist), and head orientation. In previous HRI work, one or another of these models was arbitrarily chosen as the "obvious" interpretation of pointing gestures. For example, Whitney et al. [92] and Azari et al. [7] use the eye-to-hand vector, Tölgyessy et al. [87] and Hu et al. [40] use

elbow-to-wrist, Mayer et al. [59] use eye gaze and Yoon et al. [100] use the shoulder-to-wrist vector. However, to our knowledge, there has not been a systematic evaluation of the different approaches to interpret a pointing gesture.

Other approaches study how to enable a robot to generate a pointing gesture. Fang et al. [23] described incorporating pointing gestures with language where the robot points to objects in order to specify them to a person. Williams et al. [94] studied how humans interpret robot pointing behavior, finding that the robot's head and neck were important in understanding pointing references. They studied three hypothesis: the arm vector, the line of sight from the robot's head to the end of the gripper, and the direction of the robot's head. They found that people interpret the pointing behavior of robots differently from that of people, using the robot's head gaze more than people do.

There have been many previous works on robot object search. Model-free approaches [24, 68, 15, 27] leverage deep neural networks to learn a policy end-to-end, and thus are data hungry and have limited generalization capabilities. Model-based works frequently employ the Partially Observable Markov Decision Processes (POMDPs) [44] framework to define and solve the problem of object search. Wandzel et al. [89] introduce Object-Oriented POMDP (OO-POMDP) to factorize the robot's belief into independent object distributions, enabling the size of the belief to scale linearly in the number of objects, and employ it for efficient multi-object search. Zheng et al. [105] extend OO-POMDP for efficient multi-object search in 3D space. We build off of their framework, termed GenMOS, for our robot object search demonstrations.

4.3 Technical Approach

Our approach enables a robot to interpret a person's pointing gesture in order to find objects in the environment. To perform this task, we need to estimate the person's body pose, and then use information from the pose to interpret pointing gestures. We explore different approaches to converting the body pose into a vector into the world. Finally, we define an observation model to convert this vector to a Gaussian expectation to enable the robot to use information from the pointing gesture to search for objects.

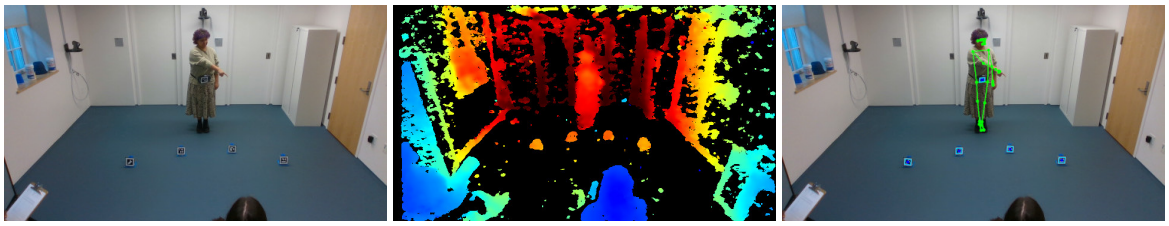


FIGURE 4.2: An example RBG images, depth image and MediaPipe's keypoint detection output.

4.3.1 Human Body Pose Estimation

We record the person's body pose with an Intel RealSense camera. For our experiments we used a camera on a tripod pointed at the scene. For the on-board experiments, we used an RGB-D camera that is part of the Spot robot's on-board sensing. Given a camera image, we need to estimate the human body pose. We use Google's MediaPipe Pose Landmarker [8] to process input RGB images and detect keypoints on the human body. We then employ the depth information to transform the relevant keypoints' coordinates from 2D into 3D space. Example input RGB-D images and MediaPipe's output are shown in Figure 4.2. We then ray-casted vectors from various key points with the user's wrist as the endpoint and calculate the vectors' intersection points with the environment as the pointing targets. Figure 4.3 presents visualizations of the five pointing vectors.

We assume the person is already in the robot's field of view. We also need to make sure the camera is calibrated relative to the position of the robot, in order to situate the pointing vector correctly in the robot's frame of reference. This requires a camera calibration step for an off-board camera, which we perform using April tags [90].

4.3.2 Converting Human Body Pose to Pointing Vectors

Given the body pose of a person, we need to extract a vector according to the pointing gesture. We explore five different algorithms for computing a vector from the person's body pose. Given this vector, we re-cast it into the environment and calculate the vectors' intersection points with the environment as the pointing targets. We explore two different high-level approaches: the vector from the head to the hand, and the vector from the arm. We use the

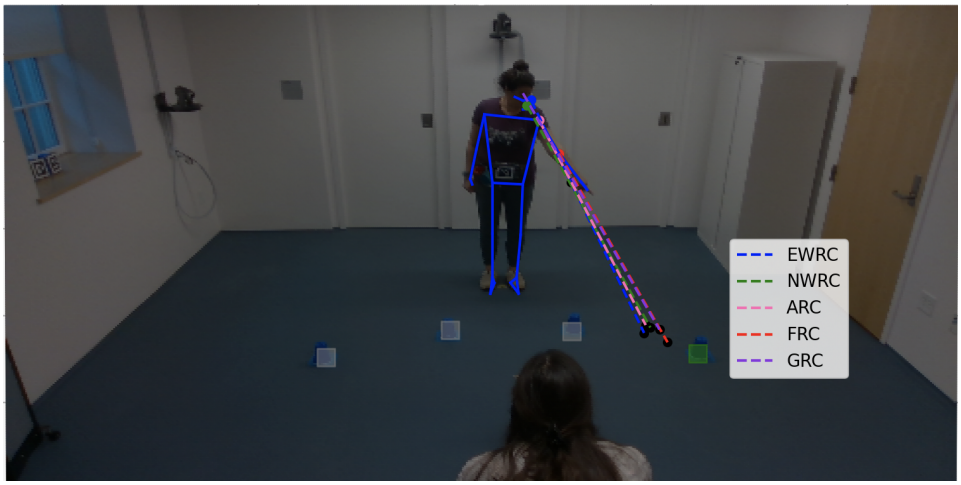


FIGURE 4.3: Five pointing vectors on a sample image: eye-to-wrist, nose-to-wrist, shoulder-to-wrist, elbow-to-wrist, and eye gaze. The left wrist is used as the frame of reference.

person's wrist position as a proxy for their hand, as fingers are much smaller and thus more difficult to detect.

To form a corresponding gaze vector that represents the general direction the user is looking at, we computed the normal vector to the plane passing through their left eye, right eye, and center of mouth. To distinguish meaningful gestures from arbitrary noise such as from crossed arms, we included an angle threshold so that vectors could be filtered out. Visualizations of the five pointing vectors are shown in Figure 4.3.

Our work uses five pointing vectors:

1) Eye-to-wrist ray-cast (EWRC): Defined by a vector connecting the eye and wrist of the pointing arm.

2) Nose-to-wrist ray-cast (NWRC): Defined by a vector connecting the nose and wrist of the pointing arm.

3) Arm ray-cast (ARC): A ray-cast defined by a vector connecting the shoulder and wrist of the pointing arm.

4) Forearm ray-cast (FRC): A ray-cast defined by a vector connecting the elbow and wrist of the pointing arm.

5) Gaze ray-cast (GRC): To establish a corresponding gaze vector representing the general direction the user is looking at, we computed the normal vector to the plane passing through

their left eye, right eye, and center of the mouth.

4.3.3 Observation Model for Pointing Vectors

A second technical contribution of this paper is a formal observation model for object search defined in terms of a pointing vector established from the person's body in Section 4.3.2. Following Zheng et al. [105], we formalize the object search problem as a OO-POMDP. The robot state consists of its pose; actions consist of moving through the environment, looking with its sensor at a particular location, and marking the object as found. After correctly marking the object as found, the robot receives a reward.

We define an observation model for pointing for object search as:

$$\Pr(o|s,a) \tag{4.1}$$

In previous work, the observation consisted of sensor input from the robot's sensor, as well as natural language input from the person. However, to our knowledge, no previous work has used pointing gestures for giving a robot information for object search. Our work assumes the vector can be parameterized by two angles, corresponding to the "pitch" and "yaw" of the pointing vector, which we term α and β, defined in Section 4.3.2. We assume the vector is parameterized by the person's location, which is known, height, which is directly observed, and two angles, α and β, giving us:

$$\Pr(o|s,a) = \Pr(\alpha,\beta|s,a) \tag{4.2}$$

We project this distribution forward to the ground as a vector as shown in Figure 4.4. The noise of this vector will result in a Gaussian shaped like an ellipse because moving the "pitch" of the arm will move the target farther away, creating more uncertainty about where the object is; whereas moving the "yaw" will keep the same distance. Thus uniform variance in both of these two angles will result in a different distribution on where the point intersects the plane.

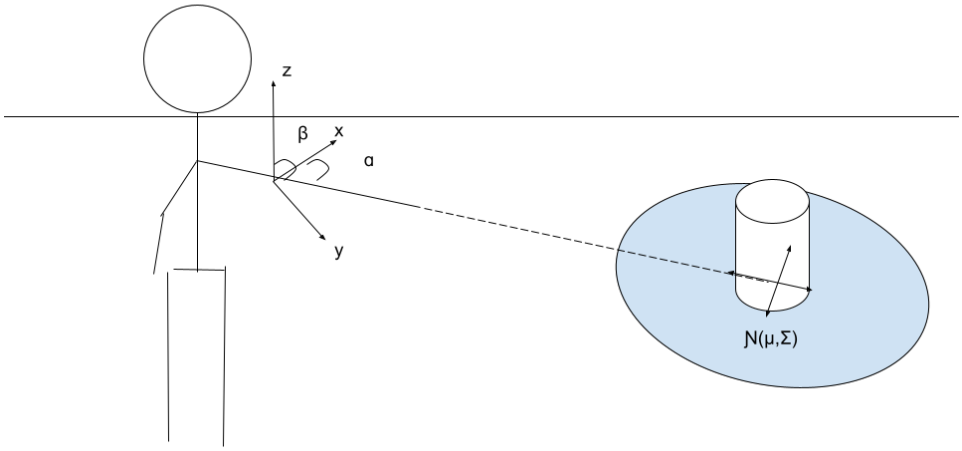


FIGURE 4.4: A diagram showing our observation model as a Gaussian projected onto the ground plane. For simplicity we show this vector as the arm vector (method ARC); in reality our evaluation assesses a number of different pointing vectors.

$$\Pr(\alpha, \beta | X) = \mathcal{N}(\mu, \Sigma) \tag{4.3}$$

In our evaluation we collect a dataset of people pointing out objects. For each item in the dataset, n, we have information about the true pose of the object, X, as well as the pointing ray α and β, observed from depth measurements. Given this information we can compute the perplexity of the model over the dataset N as follows:

$$Perplexity(N) = exp\left\{-\frac{1}{|N|}\sum_{n \in N} \log \Pr(\alpha, \beta | X)\right\} \tag{4.4}$$

In the special case where we know the object is in one of k predefined locations t_i and the distance from the pointing ray intersection location to each target d_j, we can compute the perplexity as a multinomial over the true location as follows:

$$\mathcal{L}(t_i|d_1,...,d_k) \propto \frac{d_i^{-1}}{\sum_{j=1}^{k} d_j^{-1}} \tag{4.5}$$

$$Perplexity(N) = exp\left\{-\frac{1}{|N|}\sum_{n\in N}\log \mathcal{L}(t_n|d_1,...,d_k)\right\} \tag{4.6}$$

Equation 4.6 is how we compute the perplexity scores in our experiments in Section 4.4.

4.4 Evaluation

The aim of our evaluation is to measure the effectiveness of different vectors for enabling a robot to accurately and efficiently resolve human pointing gestures to find objects. We collect a new dataset of humans pointing for a non-human partner, the domestic dog. We hypothesize that human-dog interaction is similar to human-robot interaction. We contrast this with humans pointing for humans to see if there are differences in behavior. The robot we use for interpreting the pointing gesture is a quadruped robot, the Boston Dynamics Spot robot. We use this dataset to evaluate the performance of our five different approaches for resolving pointing gestures based on human body pose, and also compare our algorithm's performance to that of the dogs. Finally, we perform an end-to-end demonstration on the real robot, demonstrating our algorithm's use at enabling a robot to resolve pointing gestures.

4.4.1 Experimental Setup

To assess the natural interaction between humans and dogs through deictic gestures, we brought dog-guardian pairs into the lab to observe both how guardians naturally point for their dogs, and how their dogs behave.

4.4.1.0.1 Participants Six human-dog pairs participated in the pointing tasks. Dog owners were all adults (over 18 years of age) who acted as the primary caretaker for their dog. The dogs were 5.2 years old on average, and three of the six dogs were female.

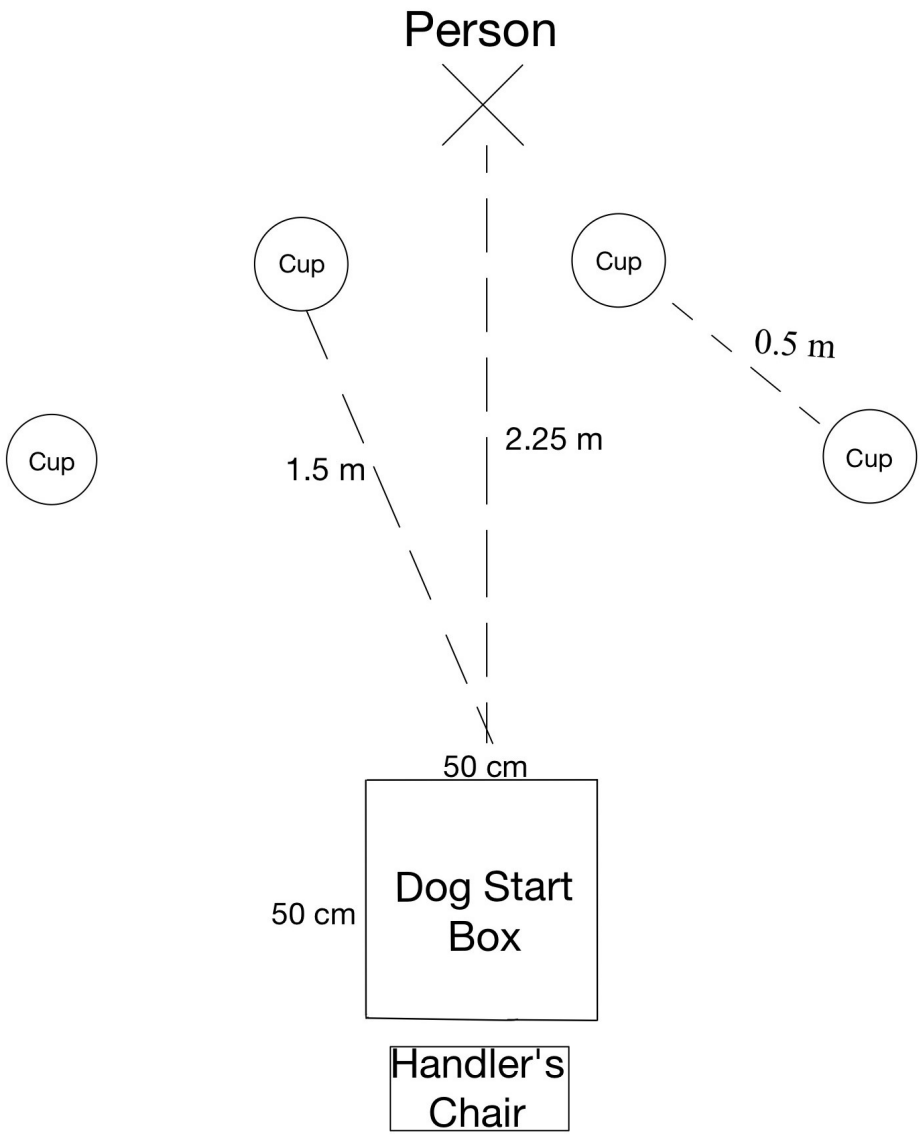


FIGURE 4.5: Sketch of the room setup for our experiments.

4.4.1.0.2 Materials The experimental setup, illustrated in Figure 4.5, comprises four cups placed equidistant in front of the dog. To minimize external device interference in the dogs' decision-making process, we utilized the Intel RealSense D435 camera to capture depth and RGB image. Dog treats were used to motivate dogs to search. An image of the room setup also appears in Figure 4.2

4.4.1.0.3 Procedure Before pointing, dog-human pairs completed two warm-up activities. First, in the initial familiarization phase, dogs observed their guardians place a treat under a cup and then were released to touch the cup, constituting a choice. This familiarized dogs with touching a target to reveal the hidden treat and was repeated four times. Next, during the hidden familiarization phase, dogs got to practice leaving the room and returning to locate a treat under the one hidden target. For our critical test trials, as in hidden familiarization at the start of each trial, the dog was led out of the room so the guardian can place a concealed treat beneath one of the targets as instructed by an experimenter (4 targets used, order semi-randomized). The dog was returned to the room, and guardians were instructed to point their dogs to the hidden treat. Dogs were then allowed to search exhaustively. This procedure was repeated for 12 trials, for a total of 72 recorded trials across dog-human pairs.

After pointing for their dogs, 3 of the 6 guardians were recorded pointing for the human experimenter to the cups in a semi-randomized order. At the start of each human pointing trial, dog guardians were asked to point to one of the four cups. The experimenter then waited approximately 2 seconds before following the point visually with their eyes and then instructing the dog guardian to point to the next cup. The room setup was the same as in Figure 4.2, and this was repeated for 12 trials, for a total of 36 recorded trials. We evaluated the 5 vectors' performance on all the data and report the results in Tables 4.1 and 4.2.

4.4.1.0.4 Evaluation Metrics We evaluate the performance of our algorithms at resolving the pointing gestures that people produced, as well as the dog's performance as measured by touching the object. We manually annotate the frame of the image used to evaluate human pointing results. We use three metrics to evaluate average object selection performance.

- Euclidean distance offset to the target object (lower is better). This metric cannot be used on dogs as we only evaluated success when they touched the correct cup.

- Weighted accuracy (higher is better), where A is 1 if the correct target was selected and 0 otherwise, n is the number of selections made until the target is selected, and w is the probability of a target being selected—calculated using the normalized inverse Euclidean distance.

$$acc = \frac{\sum_{i=1}^{n} w_i A_i}{\sum_{i=1}^{n} w_i}$$

- Perplexity(PP): Following equation 4.6 in Section 4.3.3, the target can be 1 of 4 cups, and we use the Euclidean distance from the pointing ray intersection to each cup to compute the perplexity score. The perplexity measures a model's surprise at the true location of the object, and a lower perplexity score is better.

4.4.2 Human-Dog Pointing Results

Even under naturalistic pointing conditions, dogs sometimes had difficulty following the human pointing gesture. Dogs were allowed to search exhaustively, and on their first choice dogs chose the pointed location on 37% of trials (chance being 1/4 or 25%, and on 42% of trials dogs chose correct location as their second choice. This is fairly consistent with past work with dogs when four search locations are used [51]. The two locations closer to the human pointer tend to be chosen more frequently than those on the periphery. In our sample, consistent with past work, dogs were highly accurate at choosing the correct side of the indicated cup, going to the correct side (to the pointer's Left or Right) on the first trial 76% of the time. Most errors made by dogs involved choosing the cup closer to the guardian, rather than the one further from the guardian on the same side. The proximity of the cup to the guardian may make it more attractive, as the proximity of a person is a cue that dogs can use to find hidden food [30]. It is also possible that dogs were seeking attention from their guardians, and were thus attracted to the closer locations, or that their past reward history with their guardian (meaning they have received lots of rewards directly from their guardian) causes dogs to prefer to search nearer to their guardian. We leave a full evaluation of these results

	Euclidean Distance (m) ↓	Accuracy (%) ↑	PP ↓
EWRC	0.516 *(0.071)*	**96.9 *(3.4)***	3.213 *(0.135)*
NWRC	**0.514 *(0.065)***	95.7 *(3.8)*	3.128 *(0.112)*
ARC	0.565 *(0.065)*	94.0 *(4.2)*	**3.111 *(0.135)***
FRC	0.868 *(0.272)*	92.5 *(4.2)*	3.372 *(0.131)*
GRC	2.711 *(0.158)*	51.8 *(8.4)*	3.581 *(0.120)*
Dog		64.9 *(6.7)*	4.00

TABLE 4.1: Performance on humans pointing for their dogs

	Euclidean Distance (m) ↓	Accuracy (%) ↑	PP ↓
EWRC	0.607 *(0.117)*	**100.0 *(0)***	3.228 *(0.162)*
NWRC	**0.591 *(0.108)***	**100.0 *(0)***	3.199 *(0.177)*
ARC	0.593 *(0.123)*	**100.0 *(0)***	**3.066 *(0.213)***
FRC	0.742 *(0.170)*	98.6 *(2.8)*	3.265 *(0.187)*
GRC	2.947 *(0.305)*	57.0 *(10.0)*	3.986 *(0.002)*

TABLE 4.2: Performance on data of humans pointing for other humans

to a future paper as the primary focus of this paper is the performance of our autonomous pointing algorithms.

4.4.3 Ray-cast Performance

Table 4.1 shows the performance of our five different vectors for resolving pointing gestures, along with the performance of domestic dogs, and the 95% confidence intervals. Our primary result is that most vectors perform similarly, with the lowest-performing vector using gaze alone, which performs significantly worse than the other vectors. All methods significantly outperform dogs as measured by accuracy and perplexity, probably because dogs preferred cups nearer to their guardian. It is interesting to see that the eye-to-wrist vector has higher accuracy, but the shoulder-to-wrist vector (arm ray-cast) has the lowest PP. Given the confidence intervals, there might not be much of a significant difference between which vector to use. The perplexity differs from accuracy as it is more resistant to noise in the data: a small change in the distance from the vector's intersection location to the cups can result in a large change in the accuracy but not the perplexity score.

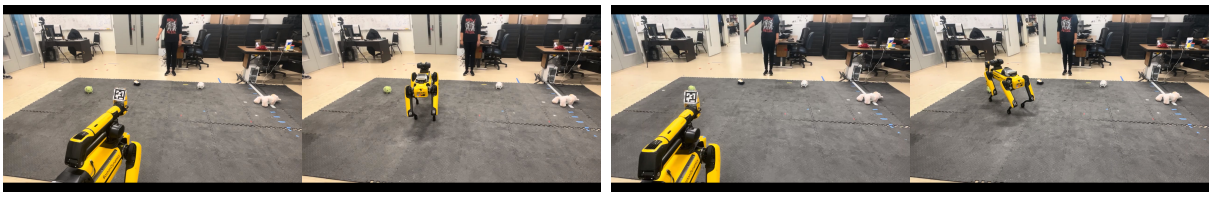


FIGURE 4.6: Our system enables the robot to correctly fetch the object the human user is pointing at, such as the penguin plush (left) and green cat (right).

4.4.4 Human-Human Pointing Experiment

Table 4.2 shows the vectors' performance on the humans pointing for other humans data. As the human experimenter told the participants which cups to point to, there is no human performance on pointing gesture resolution to report on this data. There appears to be consistent performance between nose, eye, and shoulder-to-wrist vectors. The weighted accuracy does not differ much in human-to-dog versus human-to-human, but PP is better in the human-to-human case. (PP_dog_baseline = 4, determined through a logarithmic base-2 approach and assuming a uniform distribution). While conclusions should be limited at this time given the reduced sample size, it is interesting that, as observed in the human-dog pointing data, the gaze-only vector is a much worse fit, while all other vectors perform exceptionally well. The fatigue effect could also contribute to the result, as the data was collected after the participants have completed the pointing trials with their dog.

4.4.5 Spot Demonstration

We demonstrate the effectiveness of resolving pointing vectors on the Spot robot in two ways. In both cases, we used the shoulder-to-wrist vector as it has the lowest perplexity score and comparable performance with the other vectors.

First, we assess the accuracy of pointing by directly using the vector to resolve the object reference to the object closest to the pointing vector intersection. We then used the Spot SDK[1] to direct the robot to pick up that object. The results are demonstrated in Figure 4.6 and our supplementary video. Spot was able to follow the human pointer to correctly approach and select the indicated object from a set of four candidate objects.

[1] https://github.com/boston-dynamics/spot-sdk

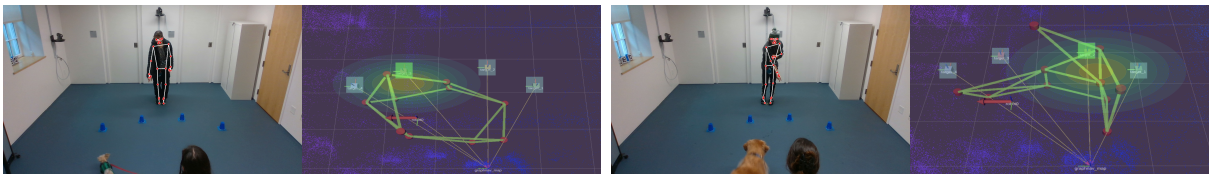


FIGURE 4.7: Examples of pointing gestures and their corresponding probability distributions derived by our observation model, which enabled the robot to constrain the search regions and effectively and accurately search for the objects. The four candidate objects are highlighted, with the robot's selected object highlighted in green.

Finally, we assess the performance of the observation model described in Section 4.3.3 for end-to-end object search using the GenMOS package [105]. Figure 4.7 illustrates results in simulation where we interpret the pointing gesture to a probability distribution, which Spot then used to constrain its search region to effectively and accurately search for the object. The green edges connect the candidate viewpoint positions for Spot, which are sampled based on the probability distribution and environment map. Results are further demonstrated in our supplementary video.

4.5 Conclusion

In this paper we presented an evaluation of different vectors to resolve human pointing gestures to locations in the environment. We present a probabilistic observation model for how this vector can be used for object search, the primary reason humans point for other entities. We evaluated our system on a new dataset of humans pointing for their domestic dogs, as well as humans pointing for other humans, and compared the performance of our autonomous algorithms to that of dogs.

We plan to revise our experimental design to further distinguish between the different vectors' performance on pointing gesture resolution. Future work can also consider using timecourse data and pointing information from videos rather than still images. Anecdotally, many pointers first aligned their gaze with the target, then moved their gaze back to the point viewer when initiating arm movement. This could help to explain why, at the moment of pointing, the gaze-only vector had such poor accuracy. In addition, we assumed that the person and their pointing gesture is within the robot's field of view. This assumption can be

relaxed by employing additional cameras in the environment following Sprute et al. [84] or human detection and tracking methods [106, 41].

Bibliography

[1] Spot® - The Agile Mobile Robot. URL https://www.bostondynamics.com/products/spot.

[2] Shaukat Abidi, MaryAnne Williams, and Benjamin Johnston. Human pointing as a robot directive. In *8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 67–68. IEEE, 2013.

[3] B. Agnetta, B. Hare, and M. Tomasello. Cues to food location that domestic dogs (Canis familiaris) of different ages do and do not use. *Animal Cognition*, 3(2):107–112, 2000. ISSN 1435-9448. doi: 10.1007/s100710000070.

[4] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On Evaluation of Embodied Navigation Agents. *arXiv preprint arXiv:1807.06757*, 2018.

[5] Nikolay Atanasov, Bharath Sankaran, Jerome Le Ny, George J Pappas, and Kostas Daniilidis. Nonmyopic View Planning for Active Object Detection. *arXiv preprint arXiv:1309.5401*, 2013.

[6] Alper Aydemir, Andrzej Pronobis, Moritz Göbelbecker, and Patric Jensfelt. Active Visual Object Search in Unknown Environments Using Uncertain Semantics. *IEEE Transactions on Robotics*, 29(4):986–1002, 2013.

[7] Bita Azari, Angelica Lim, and Richard Vaughan. Commodifying pointing in hri: simple and fast pointing gesture detection from rgb-d images. In *16th Conference on Computer and Robot Vision (CRV)*, pages 174–180. IEEE, 2019.

[8] Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. Blazepose: On-device real-time body pose tracking, 2020.

[9] Bennett I. Bertenthal, Ty W. Boyer, and Samuel Harding. When do infants begin to follow a point? *Developmental Psychology*, 50(8):2036–2048, 2014. ISSN 1939-0599. doi: 10.1037/a0037152.

[10] Emily E. Bray, Gitanjali E. Gnanadesikan, Daniel J. Horschler, Kerinne M. Levy, Brenda S. Kennedy, Thomas R. Famula, and Evan L. MacLean. Early-emerging and highly heritable sensitivity to human communication in dogs. *Current Biology*, 31(14): 3132–3136.e5, July 2021. ISSN 0960-9822. doi: 10.1016/j.cub.2021.04.055.

[11] George Butterworth. What is special about pointing in babies? In *The development of sensory, motor and cognitive capacities in early infancy: From perception to cognition*, pages 171–190. Psychology Press/Erlbaum (UK) Taylor & Francis, Hove, England, 1998. ISBN 978-0-86377-512-3.

[12] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M. Dollar. The YCB object and Model set: Towards common benchmarks for manipulation research. In *Proceedings of the IEEE International Conference on Advanced Robotics*, pages 510–517, 2015.

[13] Anthony R Cassandra. A survey of pomdp applications. In *Working notes of AAAI 1998 fall symposium on planning with partially observable Markov decision processes*, volume 1724, 1998.

[14] Yu-Wei Chao, Zhan Wang, Rada Mihalcea, and Jia Deng. Mining Semantic Affordances of Visual Object Categories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4259–4267, 2015.

[15] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object Goal Navigation using Goal-Oriented Semantic Exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[16] Kevin Chen, Christopher B. Choy, Manolis Savva, Angel X. Chang, Thomas Funkhouser, and Silvio Savarese. Text2Shape: Generating Shapes from Natural Language by Learning Joint Embeddings. In *Asian Conference on Computer Vision*, pages 100–116. Springer, 2018.

[17] Vanya Cohen, Benjamin Burchfiel, Thao Nguyen, Nakul Gopalan, Stefanie Tellex, and George Konidaris. Grounding Language Attributes to Objects using Bayesian Eigenobjects. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, 2019.

[18] Stefan Constantin, Fevziye Irem Eyiokur, Dogucan Yaman, Leonard Bärmann, and Alex Waibel. Interactive Multimodal Robot Dialog Using Pointing Gesture Recognition. In *European Conference on Computer Vision*, pages 640–657. Springer, 2022.

[19] Michael Danielczuk, Andrey Kurenkov, Ashwin Balakrishna, Matthew Matl, David Wang, Roberto Martín-Martín, Animesh Garg, Silvio Savarese, and Ken Goldberg. Mechanical Search: Multi-Step Retrieval of a Target Object Occluded by Clutter. In *Proceedings of the International Conference on Robotics and Automation*, pages 1614–1621. IEEE, 2019.

[20] Thanh-Toan Do, Anh Nguyen, and Ian Reid. AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–5, 2018.

[21] Akif Ekrekli, Alexandre Angleraud, G Sharma, and R Pieters. Co-speech gestures for human-robot collaboration. *arXiv preprint arXiv:2311.18285*, 2023.

[22] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990. doi: 10.1207/s15516709cog1402_1.

[23] Rui Fang, Malcolm Doering, and Joyce Y Chai. Embodied collaborative referring expression generation in situated human-robot interaction. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, pages 271–278, 2015.

[24] Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-based Planning. In *Proceedings of the International Conference on Robotics and Automation*, pages 5113–5120. IEEE, 2018.

[25] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.

[26] Nancy Fulda, Daniel Ricks, Ben Murdoch, and David Wingate. What Can You Do with a Rock? Affordance Extraction via Word Embeddings. *arXiv preprint arXiv:1703.03429*, 2017.

[27] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. CLIP on Wheels: Zero-Shot Object Navigation as Object Localization and Exploration. *arXiv preprint arXiv:2203.10421*, 2022.

[28] James J Gibson. The theory of affordances. *Hilldale, USA*, 1(2), 1977.

[29] Xiuye Gu, Tsung-Yi Lin, Weicheng Kuo, and Yin Cui. Open-vocabulary Object Detection via Vision and Language Knowledge Distillation. *arXiv preprint arXiv:2104.13921*, 2021.

[30] Brian Hare and Michael Tomasello. Domestic dogs (Canis familiaris) use human and conspecific social cues to locate hidden food. *Journal of Comparative Psychology*, 113(2): 173–177, 1999. ISSN 1939-2087. doi: 10.1037/0735-7036.113.2.173. Place: US Publisher: American Psychological Association.

[31] Brian Hare, Michelle Brown, Christina Williamson, and Michael Tomasello. The Domestication of Social Cognition in Dogs. *Science*, 298(5598):1634, 2002. doi: 10.1126/science.1072702.

[32] Jun Hatori, Yuta Kikuchi, Sosuke Kobayashi, Kuniyuki Takahashi, Yuta Tsuboi, Yuya Unno, Wilson Ko, and Jethro Tan. Interactively Picking Real-World Objects with Unconstrained Spoken Language Instructions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3774–3781, 2018.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[34] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2961–2969, 2017.

[35] Oliver Herbort and Wilfried Kunde. Spatial (mis-)interpretation of pointing gestures to distal referents. *Journal of Experimental Psychology: Human Perception and Performance*, 42 (1):78–89, 2016. ISSN 1939-1277. doi: 10.1037/xhp0000126.

[36] Oliver Herbort, Lisa-Marie Krause, and Wilfried Kunde. Perspective determines the production and interpretation of pointing gestures. *Psychonomic Bulletin & Review*, 28 (2):641–648, April 2021. ISSN 1531-5320. doi: 10.3758/s13423-020-01823-7.

[37] Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.

[38] Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. Segmentation from natural language expressions. In *European Conference on Computer Vision*, pages 108–124. Springer, 2016.

[39] Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. Natural Language Object Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4555–4564, 2016.

[40] Zhixian Hu, Yingtian Xu, Waner Lin, Ziya Wang, and Zhenglong Sun. Augmented Pointing Gesture Estimation for Human-Robot Interaction. In *IEEE International Conference on Robotics and Automation*, pages 6416–6422. ICRA, 2022.

[41] Md Jahidul Islam, Jungseok Hong, and Junaed Sattar. Person-following by autonomous robots: A categorical overview. *The International Journal of Robotics Research*, 38(14):1581–1618, 2019.

[42] Shruti Jaiswal, Pratyush Mishra, and GC Nandi. Deep learning based command pointing direction estimation using a single rgb camera. In *5th IEEE Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, pages 1–6. IEEE, 2018.

[43] Doreen Jirak, David Biertimpel, Matthias Kerzel, and Stefan Wermter. Solving visual object ambiguities when pointing: an unsupervised learning approach. *Neural Computing and Applications*, 33:2297–2319, 2021.

[44] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[45] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 787–798, 2014.

[46] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[47] Thomas Kollar and Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. In *Proceedings of the International Conference on Robotics and Automation*, pages 2168–2173. IEEE, 2009.

[48] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017.

[49] Okan Köpüklü, Ahmet Gunduz, Neslihan Kose, and Gerhard Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. In *14th IEEE international conference on automatic face & gesture recognition (FG 2019)*, pages 1–8. IEEE, 2019.

[50] Jayant Krishnamurthy and Thomas Kollar. Jointly Learning to Parse and Perceive: Connecting Natural Language to the Physical World. *Transactions of the Association for Computational Linguistics*, 1:193–206, 2013.

[51] Gabriella Lakatos, Márta Gácsi, József Topál, and Ádám Miklósi. Comprehension and utilisation of pointing gestures and gazing in dog–human communication in relatively complex situations. *Animal Cognition*, 15(2):201–213, March 2012. ISSN 1435-9448, 1435-9456. doi: 10.1007/s10071-011-0446-x.

[52] Jue Kun Li, David Hsu, and Wee Sun Lee. Act to See and See to Act: POMDP Planning for Objects Search in Clutter. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5701–5707. IEEE, 2016.

[53] Kristin Liebal, Tanya Behne, Malinda Carpenter, and Michael Tomasello. Infants use shared experience to interpret pointing gestures. *Developmental Science*, 12(2):264–271, March 2009. ISSN 1467-7687. doi: 10.1111/j.1467-7687.2008.00758.x.

[54] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[55] Omid Madani, Steve Hanks, and Anne Condon. On the Undecidability of Probabilistic Planning and Infinite-Horizon Partially Observable Markov Decision Problems. In *AAAI/IAAI*, pages 541–548, 1999.

[56] Arijit Mallick, Angel P. Del Pobil, and Enric Cervera. Deep Learning based Object Recognition for Robot picking task. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, pages 1–9, 2018.

[57] Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. Deep Captioning with Multimodal Recurrent Neural Networks (m-RNN). *arXiv preprint arXiv:1412.6632*, 2014.

[58] Sven Mayer, Katrin Wolf, Stefan Schneegass, and Niels Henze. Modeling distant pointing for compensating systematic displacements. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 4165–4168, 2015.

[59] Sven Mayer, Valentin Schwind, Robin Schweigert, and Niels Henze. The Effect of Offset Correction and Cursor on Mid-Air Pointing in Real and Virtual Environments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2018.

[60] A. Miklösi, R. Polgárdi, J. Topál, and V. Csányi. Use of experimenter-given cues in dogs. *Animal Cognition*, 1(2):113–121, 1998. ISSN 1435-9448. doi: 10.1007/s100710050016.

[61] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv preprint arXiv:1301.3781*, 2013.

[62] Austin Myers, Ching L. Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance Detection of Tool Parts from Geometric Features. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1374–1381, 2015.

[63] Shu Nakamura, Yasutomo Kawanishi, Shohei Nobuhara, and Ko Nishino. DeePoint: Visual Pointing Recognition and Direction Estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 20577–20587, 2023.

[64] Thao Nguyen, Nakul Gopalan, Roma Patel, Matthew Corsaro, Ellie Pavlick, and Stefanie Tellex. Robot Object Retrieval with Contextual Natural Language Queries. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.080.

[65] Thao Nguyen, Nakul Gopalan, Roma Patel, Matt Corsaro, Ellie Pavlick, and Stefanie Tellex. Affordance-based robot object retrieval. *Autonomous Robots*, 46(1):83–98, 2022.

[66] Thao Nguyen, Vladislav Hrosinkov, Eric Rosen, and Stefanie Tellex. Language-Conditioned Observation Models for Visual Object Search. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10894–10901. IEEE, 2023.

[67] Kai Nickel and Rainer Stiefelhagen. Pointing gesture recognition based on 3d-tracking of face, hands and head orientation. In *Proceedings of the 5th international conference on Multimodal interfaces*, pages 140–146, 2003.

[68] Farzad Niroui, Kaicheng Zhang, Zendai Kashino, and Goldie Nejat. Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments. *IEEE Robotics and Automation Letters*, 4(2):610–617, 2019.

[69] Takenori Obo, Ryosuke Kawabata, and Naoyuki Kubota. Cooperative human-robot interaction based on pointing gesture in informationally structured space. In *World Automation Congress (WAC)*, pages 1–5. IEEE, 2018.

[70] Genevieve Patterson and James Hays. SUN Attribute Database: Discovering, Annotating, and Recognizing Scene Attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2751–2758, 2012.

[71] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[72] Jairo Perez-Osorio, Hermann J. Müller, Eva Wiese, and Agnieszka Wykowska. Gaze Following Is Modulated by Expectations Regarding Others' Action Goals. *PLOS ONE*, 10:e0143614, 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0143614.

[73] A Alan B Pritsker. *Introduction to Simulation and SLAM II*. Halsted Press, 1984.

[74] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[75] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[76] Julia Riedel, Katrin Schumann, Juliane Kaminski, Josep Call, and Michael Tomasello. The early ontogeny of human-dog communication. *Animal Behaviour*, 75(3):1003–1014, 2008. ISSN 1095-8282. doi: 10.1016/j.anbehav.2007.08.010. Place: Netherlands Publisher: Elsevier Science.

[77] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[78] David Schlangen, Sina Zarriess, and Casey Kennington. Resolving References to Objects in Photographs using the Words-As-Classifiers Model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1213–1223, 2016. ISBN 9781510827585. URL http://arxiv.org/abs/1510.02125.

[79] Mohit Shridhar and David Hsu. Interactive Visual Grounding of Referring Expressions for Human-Robot Interaction. *arXiv preprint arXiv:1806.03831*, 2018.

[80] David Silver and Joel Veness. Monte-Carlo Planning in Large POMDPs. In *Advances in Neural Information Processing Systems*, pages 2164–2172, 2010.

[81] Beate Sodian and Claudia Thoermer. Infants' Understanding of Looking, Pointing, and Reaching as Cues to Goal-Directed Action. *Journal of Cognition and Development*, 5(3): 289–316, 2004. ISSN 1532-7647. doi: 10.1207/s15327647jcd0503_1.

[82] K. Soproni, A. Miklósi, J. Topál, and V. Csányi. Comprehension of human communicative signs in pet dogs (Canis familiaris). *Journal of Comparative Psychology*, 115(2): 122–126, June 2001. ISSN 0735-7036. doi: 10.1037/0735-7036.115.2.122.

[83] Robyn Speer, Joshua Chin, and Catherine Havasi. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[84] Dennis Sprute, Robin Rasch, Aljoscha Pörtner, Sven Battermann, and Matthias König. Gesture-based object localization for robot applications in intelligent environments. In

2018 14th International Conference on Intelligent Environments (IE), pages 48–55. IEEE, 2018.

[85] Mingxing Tan and Quoc Le. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[86] Janet L. Taylor and D. I. McCloskey. Pointing. *Behavioural Brain Research*, 29:1–5, 1988. ISSN 0166-4328. doi: 10.1016/0166-4328(88)90046-0.

[87] Michal Tölgyessy, Martin Dekan, František Duchoň, Jozef Rodina, Peter Hubinskỳ, and L'uboš Chovanec. Foundations of visual linear human–robot interaction via pointing gesture navigation. *International Journal of Social Robotics*, 9:509–523, 2017.

[88] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164, 2015.

[89] Arthur Wandzel, Yoonseon Oh, Michael Fishman, Nishanth Kumar, Wong Lawson LS, and Stefanie Tellex. Multi-Object Search using Object-Oriented POMDPs. In *Proceedings of the International Conference on Robotics and Automation*, pages 7194–7200. IEEE, 2019.

[90] John Wang and Edwin Olson. AprilTag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4193–4198. IEEE, 2016.

[91] David Whitney, Miles Eldon, John Oberlin, and Stefanie Tellex. Interpreting Multimodal Referring Expressions in Real Time. In *IEEE International Conference on Robotics and Automation*. ICRA, 2016.

[92] David Whitney, Eric Rosen, James MacGlashan, Lawson L.S. Wong, and Stefanie Tellex. Reducing Errors in Object-Fetching Interactions through Social Feedback. In *IEEE International Conference on Robotics and Automation*, pages 1006–1013. ICRA, 2017.

[93] David Whitney, Eric Rosen, James MacGlashan, Lawson L.S. Wong, and Stefanie Tellex. Reducing Errors in Object-Fetching Interactions through Social Feedback. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1006–1013, 2017.

[94] Mary-Anne Williams, Shaukat Abidi, Peter Gärdenfors, Xun Wang, Benjamin Kuipers, and Benjamin Johnston. Interpreting robot pointing behavior. In *Social Robotics: 5th International Conference, ICSR 2013, Bristol, UK, October 27-29, 2013, Proceedings 5*, pages 148–159. Springer, 2013.

[95] Marta Wnuczko and John M Kennedy. Pivots for pointing: Visually-monitored pointing has higher arm elevations than pointing blindfolded. *Journal of Experimental Psychology: Human Perception and Performance*, 37(5):1485, 2011.

[96] Amanda L. Woodward and Jose J. Guajardo. Infants' understanding of the point gesture as an object-directed action. *Cognitive Development*, 17(1):1061–1084, 2002. ISSN 1879-226X. doi: 10.1016/S0885-2014(02)00074-6.

[97] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to Learn How to Learn: Self-Adaptive Visual Navigation Using Meta-Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[98] Yuchen Xiao, Sammie Katt, Andreas ten Pas, Shengjian Chen, and Christopher Amato. Online Planning for Target Object Search in Clutter under Partial Observability. In *Proceedings of the International Conference on Robotics and Automation*, pages 8241–8247. IEEE, 2019.

[99] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[100] Sungboo Yoon, Yeseul Kim, Changbum R Ahn, and Moonseo Park. Challenges in Deictic Gesture-Based Spatial Referencing for Human-Robot Interaction in Construction. In

ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, volume 38, pages 491–497. IAARC Publications, 2021.

[101] Jimin Yu, Maowei Qin, and Shangbo Zhou. Dynamic gesture recognition based on 2D convolutional neural network and feature fusion. *Scientific Reports*, 12(1):4345, 2022.

[102] Kaiyu Zheng and Stefanie Tellex. pomdp_py: A Framework to Build and Solve POMDP Problems. *arXiv preprint arXiv:2004.10099*, 2020.

[103] Kaiyu Zheng, Yoonchang Sung, George Konidaris, and Stefanie Tellex. Multi-Resolution POMDP Planning for Multi-Object Search in 3D. *arXiv preprint arXiv:2005.02878*, 2020.

[104] Kaiyu Zheng, Yoonchang Sung, George Konidaris, and Stefanie Tellex. Multi-resolution POMDP planning for multi-object search in 3D. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021. Robocup Best Paper.

[105] Kaiyu Zheng, Anirudha Paul, and Stefanie Tellex. A System for Generalized 3D Multi-Object Search. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[106] Jianpeng Zhou and Jack Hoang. Real time robust human detection and tracking system. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops*, pages 149–149. IEEE, 2005.