LEARNING FROM LARGE-SCALE VISUAL DATA FOR ROBOTS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Ozan Şener

August 2016

LEARNING FROM LARGE-SCALE VISUAL DATA FOR ROBOTS

Ozan Şener, Ph.D.

Cornell University 2016

Humans created a tremendous value by collecting and organizing all their knowledge in publicly accessible forms, as in Wikipedia and YouTube. The availability of such large knowledge bases not only changed the way we learn, it also changed how we design artificial intelligence algorithms. Recently, propelled by the available data and expressive models, many successful computer vision and natural language processing algorithms have emerged. However, we did not see a similar shift in robotics. Our robots are still having trouble recognizing basic objects, detecting humans, and even performing simple tasks like *how to make an omelet.*

In this thesis, we study the type of knowledge robots need. Our initial analysis suggests that robots need a very unique type of knowledge base with many requirements like multi-modal data and physical grounding of concepts. We further design such a large-scale knowledge base and show how can it be used in many robotics tasks. Given this knowledge base, robots need to handle many challenges like scarcity of the supervision and the shift between different modalities and domains. We show that the common solution to all these problems is understanding the latent structure of the data. We also show that the key to discover and learn the latent structure is using large scale data. We propose machine learning algorithms, which can learn latent semantic structure with no supervision over multiple domains and modalities. Our algorithms show state-of-the-art performance in many robotics and computer vision benchmarks.

BIOGRAPHICAL SKETCH

Ozan Şener was born and raised in Sinop, a beautiful northern city in Turkey. His childhood memories mostly include writing various science-fiction stories full of robots and aliens. He learned programming during his high school in order to solve one nasty mathematical puzzle and his life changed when he realized he could build robots with that skill. He has built various robots since that day while competing in various robotics competitions. His next revelation was when he wanted to put a camera in one of his robots. He realized how challenging and fun the problem of robot perception/computer vision is and he lost his interest in mechanical and electrical engineering concepts.

He obtained his Bachelor of Science degree from Middle East Technical University, Ankara, Turkey and joined a computer vision research lab in the same university to complete his masters. During his studies he collaborated closely with Nokia Research Center, Tampere, Finland developing mobile computer vision algorithms. His algorithms were deployed in production as part of multimedia engine in Nokia N9 series phones, and he still brags about it whenever he sees someone using a Nokia phone- unfortunately, it does not happen often anymore-.

During his PhD studies at Cornell, he worked in the area of machine learning with applications to computer vision and robotics. He believes this is the perfect sweet spot for him, as he likes to work on theory, which can be applied on real systems and engineering systems, which has strong theoretical inspirations. In his free time, he likes to juggle, hike, and write. He re-activated the Cornell Juggling Club during his PhD.

To the musicians of 70s and all the kids who listened them[1]

[1]In particular, two little kids from whom my parents grew

ACKNOWLEDGEMENTS

1 - Tale of the White Rabbit [1]

— **Fox:** "What are you working on?"

— **Rabbit:** "My thesis."

— **Fox:** "Hmmm. What's it about?"

— **Rabbit:** "Oh, I'm writing about how rabbits eat foxes"

— **Fox:** "That's ridiculous! Any fool knows that rabbits don't eat foxes"

— **Rabbit:** "Sure they do, and I can prove it. Come with me."

They both disappear into the rabbit's burrow. After a few minutes, the rabbit returns, alone, to his typewriter and resumes typing.

Scene inside the rabbit's burrow: In one corner, there is a pile of fox bones. On the other side of the room, a huge lion is belching and picking his teeth. It doesn't matter what you choose for a thesis subject. It doesn't matter what you use for data. What does matter is who you have for a thesis advisor.

Following the tale of the white rabbit, I was advised by Ashutosh Saxena and he was flanked by Silvio Savarese. I am grateful to both of them for their scientific contribution, as well as allowing me to pursue my own ideas and patiently supporting me during the process. I am also thankful to my thesis committee members Emin Gün Sirer and David Mimno for their useful comments and suggestions.

2 - Down the Rabbit Hole @ Cornell First of all, I am grateful to Cornell Robot Learning Lab for creating a productive and fun environment both in Cornell and Stanford. I would like to thank Ashesh Jain, Dipendra Misra and Jae Sung for all the discussions and fun we had, Hema Koppula for all the guidance and help, and Ian Lenz and Chenxia Wu for their support and friendship.

I am thankful to everyone who made me feel like at home in New York State. Especially, members of red tigers trivia team Ken Yancey, Breana Branham and

Sean Rice, my juggling and magic partner Tayyar Rezayev, the Cornell Tech crew Stephanie Hyland, Neel Madhukar, Faisal Alquaddoomi and Natalie Davidson, and last but not least Turks of Cornell: Sinem Unal, Sevi Baltaoglu and Melik Turker.

3 - A Mad Tea-Party @ Stanford I am grateful to Amir Zamir for countless impromptu discussions about life and science, and his guidance and help, Hyun Oh Song for his help in formalizing my ideas and Iro Armeni for her patience and understanding during our collaboration. I also had a chance to mentor two amazing undergrads during my PhD Hope Casey Allen and Jay Hack. I learned a lot from our discussions and hope to same for them. I am also grateful to all members of CVGL for their friendship, Alex, Amir, Yu, Chris, Kevin and Kuang. I am also grateful to Hakan Inan for all the discussions we had about random math topics.

Figure 1: The White Rabbit [36]

I am also thankful to everyone who converted this mad tea-party of silicon valley into a joyful experience. Ilbey, Saumitro, Jonathan Ho, Shane and Rebecca; thanks a lot for your friendship. Finally, I am also grateful to one of the most beautiful places in the world -Dolores Park-

4 - Queen of Hearts ♥ It is rather tautological to say that I would not be the person who I am without my family. I thank my parents Omer and Gullizar as well as my brother Umut for their years of love and faith in me. Finally, I would like to thank Zehra Hayrici with all my heart, for the unconditional love and the infinite patience she has had for me. She has always been there to multiply my joy in good times and lift my spirits in bad times. Thank you, mavourneen.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

"Begin at the beginning," the King said, very gravely, "and go on till you come to the end: then stop."

Lewis Carroll

Alice's Adventures in Wonderland

In the last two decades, we not only collected and structured human knowledge, but we also made it accessible to the public. We, humans, collectively created many knowledge bases of millions of entries such as Wikipedia and Youtube. We currently rely on them to share information, show our skills and learn new skills. It is even possible to learn very advanced skills, for example, there are thousands of videos for "How to create a 3D CGI Character."

The availability of large-scale, crowd-generated knowledge opens up an exciting opportunity for robots, which we envision to be part of our daily lives soon. However, this opportunity also brings its difficulties. One of the most important difficulties is the fact that these knowledge bases are designed for humans, using a language only humans understand. Understanding human communication, both verbal and visual, has always been one of the fundamental problems in artificial intelligence, and the availability of large-scale knowledge makes it more important than ever.

Recently we have seen many success stories in the quest of computers understanding humans. Among all of the successful applications, common theme is the necessity of very large-scale data. Following the Zipf law, the amount of data

needed to learn a reliable model scales exponentially with the scale of the problem, and, considering the massive data-requirement for small scale applications like choosing an object class from a small dictionary; data requirement for robot perception systems can be considered intractable since robot perception requires much richer modeling. Robot perception requires a joint understanding of various concepts resulting in a space of dimension significantly larger, making the manual data collection and human supervision intractable. Hence, we need to address the fact that full supervision is not tractable for most of the robotics applications.

Robots need powerful perception algorithms, which can convert raw sensory readings into semantically meaningful representations. Given these representations, planner algorithms needs to come up with a physically plausible plan for a robot to execute. Although robot perception, in principle, aims to convert sensory input (*e.g.* videos) into semantic concepts (*e.g.* humans, objects, activities), developing it in isolation is not a scalable approach. We want our robots to robustly perform in various environments of different complexities. We even want them to perform in environments we have never seen in our knowledge bases. Hence, we want our *representations* to handle and express uncertainty. Moreover, we also want these representations to be understandable by planning systems to be useful in overall robotics scenarios. Hence, we believe the problem of robot perception learning is, indeed, a problem of learning *actionable representations* linking different domains.

Following the aforementioned major challenges of scalable data collection, handling lack of supervision and learning actionable representations, we try to answer the following questions in this thesis;

- How can we tractably collect a large knowledge base of actionable physical

information?

- How can we model uncertainty in perception algorithms?
- How can we learn actionable representations from a dataset with no/limited supervision?
- How can we overcome the shift between representations designed for planning and perception systems?

Fortunately all of these problems are somewhat linked with each other. Moreover, our key insight to all these questions lies in exploiting the *structure* in the data. Using sparsity of the structured spaces, we develop algorithms which can handle uncertainty, using low-dimensional substructure of the structured spaces, we develop algorithms which can handle lack of supervision even at the extreme case of unsupervised learning.

1.1 Robotics at Scale:

Large-scale actionable knowledge bases

Robotics systems composed of many subsystems and they are related in various ways creating a necessity of knowledge bases spanning different modalities and domains. For example, consider a very basic example of concept of a kettle. We need to relate the kettle with a physical entity which can boil water for planning purpose, similarly we need a very detailed visual description of the kettle for perception subsystem to recognize and locate them, moreover we also need to understand kettle in full geometric form for grasping and manipulation algorithms to work. Hence, most of the existing knowledge bases[181, 50, 18] are very little

use; since, they do not jointly consider modalities and they are limited to a single context.

In order to solve this problem, we introduce a knowledge engine, which learns and shares structured knowledge representations, for robots to carry out a variety of tasks. Our knowledge engine is designed to solve the unique challenges due to dealing with multiple modalities including symbols, natural language, haptic senses, robot trajectories, visual features and many others. The *knowledge* we curated through crowdsourcing comes from multiple sources including physical interactions that robots have while performing tasks (perception, planning and control), knowledge bases from the Internet and learned representations from several robotics research groups.

In Chapter 2, we discuss various technical aspects and associated challenges such as modeling the correctness of knowledge, inferring latent information and formulating different robotic tasks as queries to the knowledge engine. We describe the system architecture and how it supports different mechanisms for users and robots to interact with the engine. We give very detailed treatment about how can we design a multi modal knowledge base which can scale to thousands of videos as we further use in Chapter 4. This knowledge engine is developed in collaboration with many researchers as a part of a bigger project [1], and our contribution mostly lies in designing a large-scale multi-modal system architecture and integrating existing knowledge bases.

[1] In collaboration with Saxena, Jain, Jami, Misra, Koppula as part of http://robobrain.me

1.2 Perception through Time:

Handling unknowns of robot perception

Context is a key to many robot perception tasks. For example, it is crucial to model relationships between objects if the final aim is parsing the visual scene into humans and objects [5]. Similarly, a context of humans is critical to understand activities [102, 83]. One of the very successful ways of modeling context is using graphical models.

Although graphical models are very powerful tools to model context, inference in such a system typically follows finding the Maximum-a-Posterior solution via solving a combinatorial optimization problem . Hence, such system generates a single prediction with possible errors. The errors typically come from both known and unknown unknowns of the systems. The error can be because of a behavior never seen in the dataset or because of a behavior that very rarely happens in the data and considered to be a noise by learning algorithms. The solution this problem is designing representations, which can handle uncertainty caused by known and unknown unknowns.

In Chapter 3, we present a new recursive algorithm that we call Recursive Conditional Random Field (rCRF) which can compute a belief over a temporal CRF model accurately representing uncertainty. We extend the graphical models to temporal structures that can model the uncertainty by estimating a full believe over predicted variables. We only use a structured diversity in our formulation and present a computationally tractable inference and learning algorithm based on Bayesian filtering and combinatorial diverse-M-best construction. We further apply our model on the problem of efficiently computing beliefs over future hu-

man activities from RGB-D videos. We present our extensive experimentation on human activity anticipation setup and also briefly describe another successful application of our algorithm on the problem of parsing large point clouds.

1.3 Learning at Scale:

Learning actionable representations with no-supervision

After we develop our large-scale actionable knowledge base *RoboBrain*, one observation we had was scarcity of supervision. The powerful knowledge we collected from YouTube in terms of instructional videos had no supervision on them, moreover, existing supervision was very sparse covering only the subset of the videos. Moreover, most of the videos also has human speech in it which requires a multimodal approach.

In order to handle the challenge of learning with limited/no supervision over multiple modalities, we rely on the underlying structure of the data as well. For example, consider the YouTube videos. They are typically generated by humans to communicate their knowledge. Moreover, human communication typically has an underlying structure, with a starting point, ending, and certain objective steps between them. In order to exploit this structure, we propose a method for parsing a video into such semantic steps in an unsupervised way in Chapter 4. We accomplish this using both visual and language cues in a joint generative model. Our method can also provide a textual description for each of the identified semantic steps and video segments.

1.4 Sharing at Scale:

Learning to close the gap between domains

Main motivation behind our large-scale actionable knowledge base *RoboBrain* is letting robots share knowledge with each other. This is critical since robotics system includes many sub-systems and most of the research is contained within one of the sub-domains. Hence, scaling is only possible when we close the gap between domains. For example, most of the available robot perception systems are trained using real camera/sensor recordings. On the other hand, planning algorithms mostly uses symbolic representations or very simple visual images like computer generated images in video game logs as in [136].

In order to handle multiple domains, we also rely on structure. Our construction was simply using the available supervised domains to supervise rest of the domains. In other words, we approach the problem from a transductive perspective. We incorporate the domain shift and the transductive target inference into our framework by jointly solving for an asymmetric similarity metric and the optimal transductive target label assignment. Key insight to solve this metric learning with limited supervision was enforcing structural consistency in unsupervised domains. We also show that our model can easily be extended for deep feature learning in order to learn features that are discriminative in the target domain. We show that our method is capable of linking artificial computer generated images planner algorithms use with actual images from robot perception datasets in Chapter 5

1.5 Designing Robotics Systems using Our Methods

We study each of the aforementioned problems separately in a full abstraction; however, it is crucial to understand how each of our proposed algorithms can be used in end-to-end robotics systems. In order to answer this question, let's consider a simple example of a household robot being asked, "Make me a pancake, please!" At this point, robot needs to answer many questions;

1. How can one make a pancake?
2. What are the objects in this kitchen?
3. What are the relationships between the objects and the pancake?

Our algorithms about large-scale knowledge bases in Chapter 2 and parsing large-scale video collections in Chapter 4 are useful to answer the first question. Our proposed algorithm converts the unstructured "Make me a pancake" query into a structured recipe. Our uncertainty aware activity and object understanding method from Chapter 3 is used to not only describe the objects in the kitchen but also model the uncertainty of our knowledge about them. Its output is every object and their plausible affordances like "object X can be used to carry water with probability 0.6". After we discover the structured recipes and understand the environment, we need to answer the following question;

- How can robot apply each action in the structured recipe to the environment described?

In order to apply each action, we need to solve the domain difference between perception and planning algorithm since the planners we use is based on CGI

(computer generated imagery) simulations and our recipes are based on real images. We use our domain adaptation algorithm from Chapter 5 to align domains.

1.6 First published appearances of the contributions

Most of the contributions described in this dissertation have first appeared as the following publications.

- Chapter 2: Saxena, Jain, Sener, Jami, Misra, and Koppula [166]
- Chapter 3: Sener and Saxena [170]; Armeni, Sener, Zamir, Jiang and Savarese [8]
- Chapter 4: Sener, Zamir, Savarese and Saxena [171]; Sener, Wu, Zamir, Savarese and Saxena [169]
- Chapter 5: Sener, Song, Saxena, Saverese [168].

The following research works are applications or are related to my thesis, but are not fully presented in this thesis:

- Semantic parsing of large-scale building clouds. This is an application of our approach in Chapter 2 and is described completely in [8].
- Parsing RGB-D Activity Videos. This is a similar approach to our multi-modal parsing method in Chapter 3 that we specifically develop for RGB-D activity videos. It is described fully in [202].
- Efficient Inference on CRF for Videos. This is spatial extension of our rCRF algorithm from Chapter 2. We develop it for mobile devices considering constrained computational resources and fully explain in [45].

CHAPTER 2

ROBOTICS AT SCALE - ROBOBRAIN

You can have data without information, but you cannot have information without data.

Daniel Keys Moran

Robotics systems are typically composed of many sub-systems and these systems are developed in isolation. Moreover, robots learn continuously after deployment using human feedback is isolation as well. Hence, one of the most critical problems in robotics is sharing knowledge among different robotic systems as well as robotics sub-systems in deployment. In this chapter, we introduce the knowledge-engine we developed which enables robotics systems to share knowledge among robotic platforms as well as robotics sub-systems. The knowledge-engine we developed can handle variety of modalities and domains. More importantly, the knowledge comes from a variety of sources including online knowledge bases, learned concepts from robotics research groups from all over the world as well as physical robotic experiments.

Contributions of this chapter. We developed this knowledge engine in collaboration with Saxena, Jain, Jami, Misra and Koppula, and we call it RoboBrain. Within the scope of this dissertation, I contributed to a large-scale distributed system architecture which can scale to thousands of videos, images and text files. For completeness, in this chapter we also describe the formal definition of knowledge-engine, the query language for accessing the knowledge and a various applications of RoboBrain. RoboBrain is available at: http://www.robobrain.me

2.1 Why do we need robot knowledge bases?

Over the last decade, we have seen many successful applications of large-scale knowledge systems. Examples include Google knowledge graph [37], IBM Watson [49], Wikipedia, and many others. These systems know answers to many of our day-to-day questions, and not crafted for a specific task, which makes them valuable for humans. Inspired by them, researchers have aggregated domain specific knowledge by mining data [9, 18], and processing natural language [25], images [34] and speech [137]. These sources of knowledge are specifically designed for humans, and their human centric design makes them of limited use for robots—for example, imagine a robot querying a search engine for how to "bring sweet tea from the kitchen" (Figure 2.1).

In order to perform a task, robots require access to a large variety of information with finer details for performing perception, planning, control and natural language understanding. When asked to bring sweet tea, as shown in Figure 2.1, the robot would need access to the knowledge for grounding the language symbols into physical entities, the knowledge that sweet tea can either be on a table or in a fridge, and the knowledge for inferring the appropriate plans for grasping and manipulating objects. Efficiently handling this joint knowledge representation across different tasks and modalities is still an open problem.

In this chapter we present RoboBrain that allows robots to learn and share such representations of knowledge. We learn these knowledge representations from a variety of sources, including interactions that robots have while performing perception, planning and control, as well as natural language and visual data from the Internet. Our representation considers several modalities including symbols, natural language, visual or shape features, haptic properties, and so on. RoboBrain

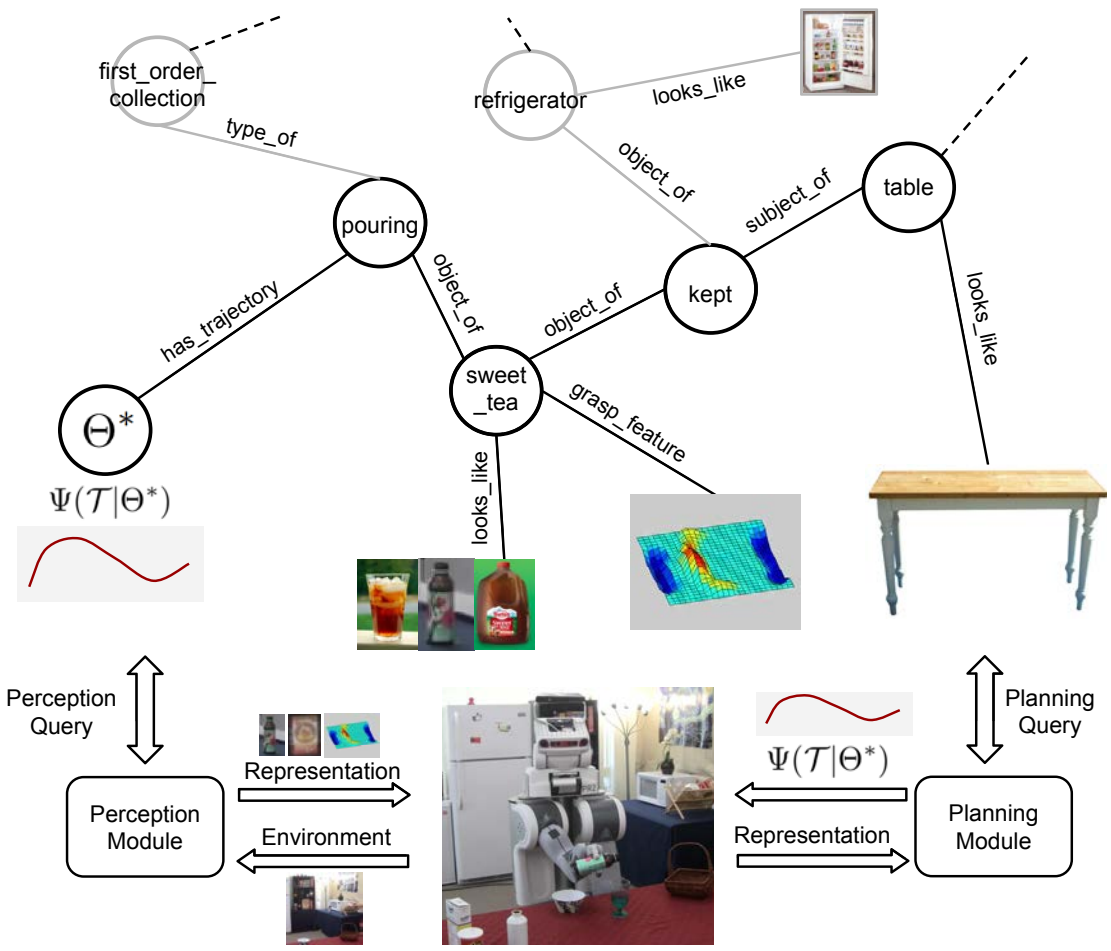


Figure 2.1: **An example showing a robot using RoboBrain for performing tasks.** The robot is asked "Bring me sweet tea from the kitchen", where it needs to translate the instruction into the perceived state of the environment. RoboBrain provides useful knowledge to the robot for performing the task: (a) sweet tea can be kept on a table or inside a refrigerator, (b) bottle can be grasped in certain ways, (c) opened sweet tea bottle needs to be kept upright, (d) the pouring trajectory should obey user preferences of moving slowly to pour, and so on.

connects this knowledge from various sources and allow robots to perform diverse tasks by jointly reasoning over multiple data modalities.

RoboBrain enables sharing from multiple sources by representing the knowledge

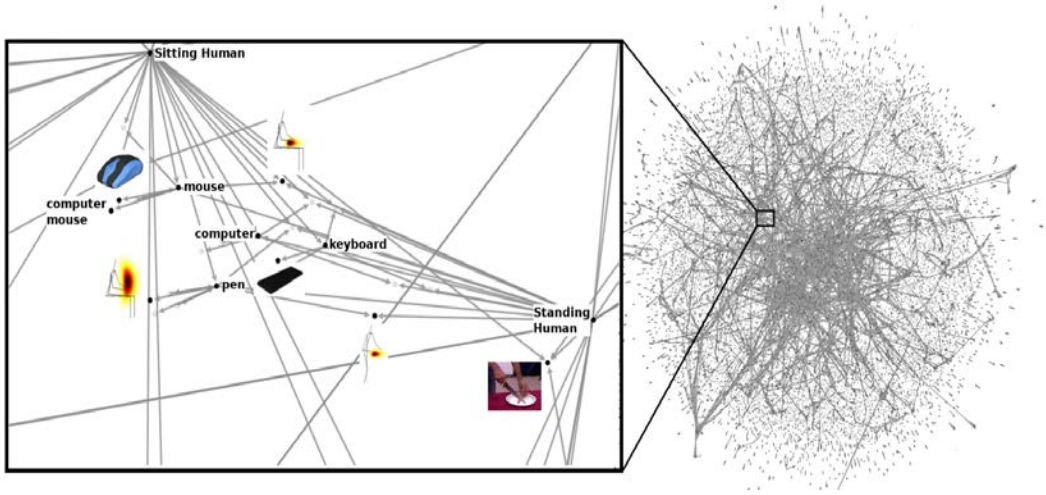


Figure 2.2: **A visualization of the RoboBrain graph** on Nov 2014, showing about 45K nodes and 100K directed edges. The left inset shows a zoomed-in view of a small region of the graph with rendered media. This illustrates the relations between multiple modalities namely images, heatmaps, words and human poses. *For high-definition graph visualization, see:* `https://sites.google.com/site/robotknowledgeengine/`

in a graph structure. Traversals on the RoboBrain graph allow robots to gather the specific information they need for a task. This includes the semantic information, such as different grasps of the same object, as well as the functional knowledge, such as spatial constraints (e.g., a bottle is kept on the table and not the other way around). The key challenge lies in building this graph from a variety of knowledge sources while ensuring dense connectivity across nodes. Furthermore, there are several challenges in building a system that allows concurrent and distributed update, and retrieval operations.

We present use of RoboBrain on three robotics applications in the area of grounding natural language, perception and planning. For each application we show usage of RoboBrain *as-a-service*, which allow researchers to effortlessly use the state-of-the-art algorithms. We also present experiments to show that sharing

knowledge representations through RoboBrain improves existing language grounding and path planning algorithms.

RoboBrain is a collaborative project that we support by designing a large-scale cloud architecture. In the current state, RoboBrain stores and shares knowledge across several research projects [188, 136, 78, 81, 84, 103, 201, 122] and Internet knowledge sources [121, 47]. We believe as more research projects contribute knowledge to RoboBrain, it will not only improve the concerned project but will also be beneficial for the robotics community at large.

The goal of the chapter is to present an overall view of the RoboBrain, its architecture, functionalities, and demonstrate its application to robotics. In Section 2.4 we formally define the RoboBrain graph and describe its system architecture in Section 2.5. In order for robots to use RoboBrain we propose the Robot Query Library in Section 2.6. In Section 2.7 we present different robotic applications using RoboBrain.

2.2 Related Work on Robot Knowledge Bases

We now describe some works related to RoboBrain. We first give an overview of the existing knowledge bases and describe how RoboBrain differs from them. We then describe some works in robotics that can benefit from RoboBrain, and also discuss some of the related on-going efforts.

Knowledge bases. Collecting and representing a large amount of information in a knowledge base (KB) has been widely studied in the areas of data mining, natural language processing and machine learning. Early seminal works have manually

created KBs for the study of common sense knowledge (Cyc [121]) and lexical knowledge (WordNet [47]). With the growth of Wikipedia, KBs started to use crowd-sourcing (DBPedia [9], Freebase [18]) and automatic information extraction (Yago [181, 74], NELL [25]) for mining knowledge.

One of the limitations of these KBs is their strong dependence on a single modality that is the text modality. There have been few successful attempts to combine multiple modalities. ImageNet [34] and NEIL [28] enriched text with images obtained from Internet search. They used crowd-sourcing and unsupervised learning to get the object labels. These object labels were further extended to object affordances [210].

We have seen successful applications of the existing KBs within the modalities they covered, such as IBM Watson Jeopardy Challenge [50]. However, the existing KBs are human centric and do not directly apply to robotics. The robots need finer details about the physical world, e.g., how to manipulate objects, how to move in an environment, etc. In RoboBrain we combine knowledge from the Internet sources with finer details about the physical world, from RoboBrain project partners, to get an overall rich graph representation.

Robot Learning. For robots to operate autonomously they should perceive their environments, plan paths, manipulate objects and interact with humans. We describe previous work in each of these areas and how RoboBrain complements them.

Perceiving the environment. Perception is a key element of many robotic tasks. It has been applied to object labeling [109, 5, 201], scene understanding [95, 68], robot localization [133, 139], path planning [89], and object affordances [33, 104]. RoboBrain stores perception related knowledge in the form of 3D point clouds,

grasping features, images and videos. It also connects this knowledge to human understandable concepts from the Internet knowledge sources.

Path planning and manipulation. Planning algorithms formulate action plans which are used by robots to move around and modify its environment. Planning algorithms have been proposed for the problems of motion planning [215, 167], task planning [4, 19] and symbolic planning [43, 158]. Some planning applications include robots baking cookies [19], folding towels [173], assembling furniture [97], and preparing pancakes [13]. The previous works have also learned planning parameters using methods such as Inverse Optimal Control [3, 156, 211, 78]. RoboBrain stores the planning parameters learned by previous works and allow the robots to query for the parameters.

Interacting with humans. Human-robot interaction includes collaborative tasks between humans and robots [143, 142], generating safe and human-like robot motion [127, 115, 58, 39, 24], interaction through natural language [187, 136], etc. These applications require joint treatment of perception, manipulation and natural language understanding. RoboBrain stores different data modalities required by these applications.

Previous efforts on connecting robots range from creating a common operating system (ROS) for robots [152] to sharing data acquired by various robots via cloud [197, 6]. For example, the RoboEarth [197] provides a platform for the robots to store and off-load computation to the cloud and communicate with other robots; and the KIVA systems [6] use the cloud to coordinate motion for hundreds of mobile platforms. On the other hand, RoboBrain provides a knowledge representation layer on top of data storing, sharing and communication.

Open-Ease [14] is a related on-going effort towards building a knowledge engine for robots. Open-Ease and RoboBrain differ in the way they learn and represent knowledge. In Open-Ease the knowledge is represented as formal statements using pre-defined templates. On the other hand, the knowledge in RoboBrain is represented as a graph. The nodes of the RoboBrain graph have no pre-defined templates and they can be any robotic concept like grasping features, trajectory parameters, and visual data. This graph representation allows partner projects to easily integrate their learned concepts in RoboBrain. The semantic meaning of concepts in the RoboBrain graph are represented by their connectivity patterns in the graph.

2.3 Overview of RoboBrain

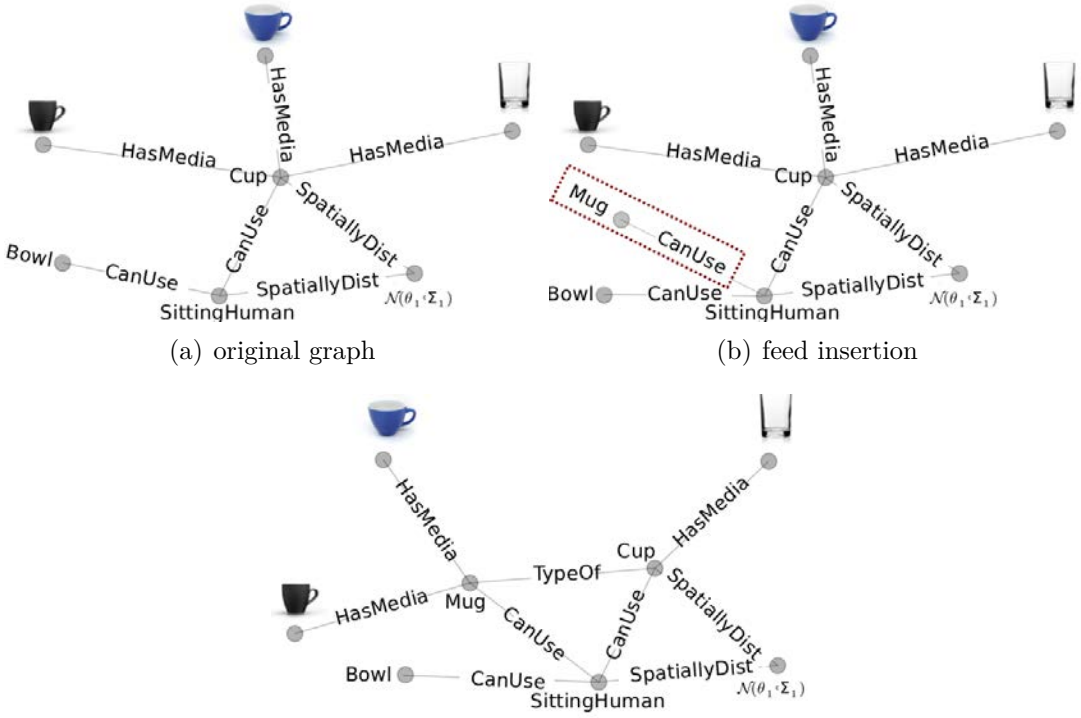
RoboBrain is a never ending learning system that continuously incorporates new knowledge from its partner projects and from different Internet sources. One of the functions of RoboBrain is to represent the knowledge from various sources as a graph, as shown in Figure 2.2. The nodes of the graph represent concepts and edges represent the relations between them. The connectivity of the graph is increased through a set of graph operations that allow additions, deletions and updates to the graph. As of the date of this submission, RoboBrain has successfully connected knowledge from sources like WordNet, ImageNet, Freebase, OpenCyc, parts of Wikipedia and other partner projects. These knowledge sources provide lexical knowledge, grounding of concepts into images and common sense facts about the world.

The knowledge from the partner projects and Internet sources can sometimes

be erroneous. RoboBrain handles inaccuracies in knowledge by maintaining beliefs over the correctness of the concepts and relations. These beliefs depend on how much RoboBrain trusts a given source of knowledge, and also the feedback it receives from crowd-sourcing (described below). For every incoming knowledge, RoboBrain also makes a sequence of decisions on whether to form new nodes, or edges, or both. Since the knowledge carries semantic meaning RoboBrain makes many of these decisions based on the contextual information that it gathers from nearby nodes and edges. For example, RoboBrain resolves polysemy using the context associated with nodes. Resolving polysemy is important because a 'plant' could mean a 'tree' or an 'industrial plant' and merging the nodes together will create errors in the graph.

RoboBrain incorporates supervisory signals from humans in the form of crowd-sourcing feedback. This feedback allows RoboBrain to update its beliefs over the correctness of the knowledge, and to modify the graph structure if required. While crowd-sourcing feedback was used in some previous works as means for data collection (e.g., [34, 161]), in RoboBrain they serve as supervisory signals that improve the knowledge engine. RoboBrain allows user interactions at multiple levels: (i) Coarse feedback: these are binary feedback where a user can "Approve" or "Disapprove" a concept in RoboBrain through its online web interface; (ii) Graph feedback: these feedback are elicited on RoboBrain *graph visualizer*, where a user modifies the graph by adding/deleting nodes or edges; (iii) Robot feedback: these are the physical feedback given by users directly on the robot.

In this chapter we discuss different aspects of RoboBrain, and show how RoboBrain serves as a knowledge layer for the robots. In order to support knowledge sharing, learning, and crowd-sourcing feedback we develop a large-scale distributed



(a) original graph

(b) feed insertion

(c) after $merge(Mug, Mug') \rightarrow Mug \circ split(Cup) \rightarrow (Cup, Mug')$

Figure 2.3: **Visualization of inserting new information.** We insert '*Sitting human can use a mug*' and RoboBrain infers the necessary split and merge operations on the graph. In (a) we show the original sub-graph, In (b) information about a *Mug* is seen for the first time and the corresponding node and edge are inserted, In (c) inference algorithm infers that previously connected cup node and cup images are not valid any more, and it splits the *Cup* node into two nodes as *Cup* and Mug' and then merges Mug' and *Mug* nodes.

system. We describe the architecture of our system in Section 2.5. In Section 2.6 we describe the robot query library, which allow robots to interact with RoboBrain. Through experiments we show that robots can use RoboBrain *as-a-service* and that knowledge sharing through RoboBrain improves existing robotic applications. We now present a formal definition of our Robot Knowledge Engine and the graph.

2.4 Knowledge Engine: Formal Definition

In this section we present the formal definition of RoboBrain. RoboBrain represents knowledge as a directed graph $\mathcal{G} = (V, E)$. The vertices V of the graph stores concepts that can be of a variety of types such as images, text, videos, haptic data, or learned entities such as affordances, deep learning features, parameters, etc. The edges $E \subseteq V \times V \times C$ are directed and represents the relations between concepts. Each edge has an edge-type from a set C of possible edge-types.

An edge (v_1, v_2, ℓ) is an ordered set of two nodes v_1 and v_2 and an edge-type ℓ. Few examples of such edges are: (StandingHuman, Shoe, *CanUse*), (StandingHuman, $\mathcal{N}(\mu, \Sigma)$, *SpatiallyDistributedAs*) and (Grasping, DeepFeature23, *UsesFeature*). We do not impose any constraints on the type of data that nodes can represent. However, we require the edges to be consistent with RoboBrain edge set C. We further associate each node and edge in the graph with *a feature vector representation* and a *belief*. The feature vector representation of nodes and edges depend on their local connections in the graph, and their belief is a scalar probability over the accuracy of the information that the node or an edge represents. Tables 2.1 and 2.2 show few examples of nodes and edge-types. A snapshot of the graph is shown in Figure 2.2.

2.4.1 Creating the Graph

Graph creation consists of never ending cycle of two stages namely, knowledge acquisition and inference. Within the knowledge acquisition stage, we collect data from various sources and during the inference stage we apply statistical techniques to update the graph structure based on the aggregated data. We explain these two

stages below.

Knowledge acquisition: RoboBrain accepts new information in the form of set of edges, which we call a *feed*. A *feed* can either be from an automated algorithm crawling the Internet sources or from one of RoboBrain's partner projects. We add a new *feed* to the existing graph through a sequence of union operations performed on the graph. These union operations are then followed by an inference algorithm. More specifically, given a new *feed* consisting of a set of N edges $\{(v_1^1, v_2^1, \ell^1) \dots (v_1^N, v_2^N, \ell^N)\}$, and the existing graph $G = (V, E)$. The graph union operations give a graph $G' = (V', E')$ as follows:

$$\begin{aligned} V' &= v_1^1 \cup v_2^1 \cup \dots \cup v_1^N \cup v_2^N \cup V \\ E' &= (v_1^1, v_2^1, \ell^1) \cup \dots \cup (v_1^N, v_2^N, \ell^N) \cup E \end{aligned} \tag{2.1}$$

Inference on the Graph: After adding the *feed* to the graph using equation (2.1), we perform inference to update the graph based on this new knowledge. The inference outputs a sequence of graph operations which are then performed on the graph. These graph operations modify the graph by adding new nodes or edges to the graph, deleting nodes or edges from the graph, merging or splitting nodes, etc.

We mention two graph operations here: *split* and *merge*. The split operation is defined as splitting a node into a set of two nodes. The edges having end points in the split node are connected to one of the resultant nodes using the inference algorithm. A merge operation is defined as merging two nodes into a single node, while updating the edges connected to the merged nodes. An example of such an update is shown in Figure 2.3. When a new information *"sitting human can use a mug"* is added to the graph, it causes the *split* of the *Cup* node into two nodes: a *Cup* and a *Mug* node. These two are then connected by an edge-type *TypeOf*.

Table 2.1: Some examples of different node types in our RoboBrain graph. For full-list, please see the code documentation.

Word	an English word represented as an ASCII string
DeepFeature	feature function trained with a Deep Neural Network
Image	2D RGB Image
PointCloud	3D point cloud
Heatmap	heatmap parameter vector

The graph update can be expressed through the following equation:

$$G^{\star} = split_{v_{s_1}} \circ merge_{v_{m_1}, v_{m_2}} \circ \ldots \circ split_{v_{s_M}} \circ G'$$

In the above equation $G^{\star}$ is the graph obtained after the inference. The goal of the inference steps is to modify the graph G' in a way that best explains the physical world. However, the graph that captures the real physical world is a *latent graph*, i.e., it is not directly observable. For example, the latent information that *"coffee is typically in a container"* is partially observed through many edges between the *coffee* node and the nodes with *container* images. Our graph construction can also be explained in a generative setting of having a latent graph with all the knowledge about physical word, and we only observe noisy measurements in form of *feeds*. In this chapter, we abstract the algorithmic details of inference and focus on the overall ideas involved in RoboBrain, its architecture, and its application to robotics.

Table 2.2: Some examples of different edge types in our RoboBrain graph. For full-list, please see the code documentation.

IsTypeOf	human *IsTypeOf* a mammal
HasAppearance	floor *HasAppearance* as follows (this image)
CanPerformAction	human *CanPerformAction* cutting
SpatiallyDistributedAs	location of human is *SpatiallyDistributedAs*
IsHolonym	tree *IsHolonym* of leaf

2.5 System Architecture

We now describe the system architecture of RoboBrain, shown in Figure 2.4. The system consists of four interconnected layers: (a) knowledge acquisition, (b) knowledge parser, (c) knowledge storage, and (d) knowledge inference. The principle behind our design is to efficiently process large amount of unstructured multi-modal knowledge and represent it using the structured RoboBrain graph. In addition, our design also supports various mechanisms for users and robots to interact with RoboBrain. Below we discuss each of the components.

Knowledge acquisition layer is the interface between RoboBrain and the different sources of multi-modal data. Through this layer RoboBrain gets access to new information which the other layers process. RoboBrain primarily collects knowledge through its partner projects and by crawling the existing knowledge bases such as Freebase, ImageNet and WordNet, etc., as well as unstructured sources such as Wikipedia.

Knowledge parser layer of RoboBrain processes the data acquired by the acquisition layer and converts it to a consistent format for the storage layer. It also marks the incoming data with appropriate meta- data such as timestamps, source

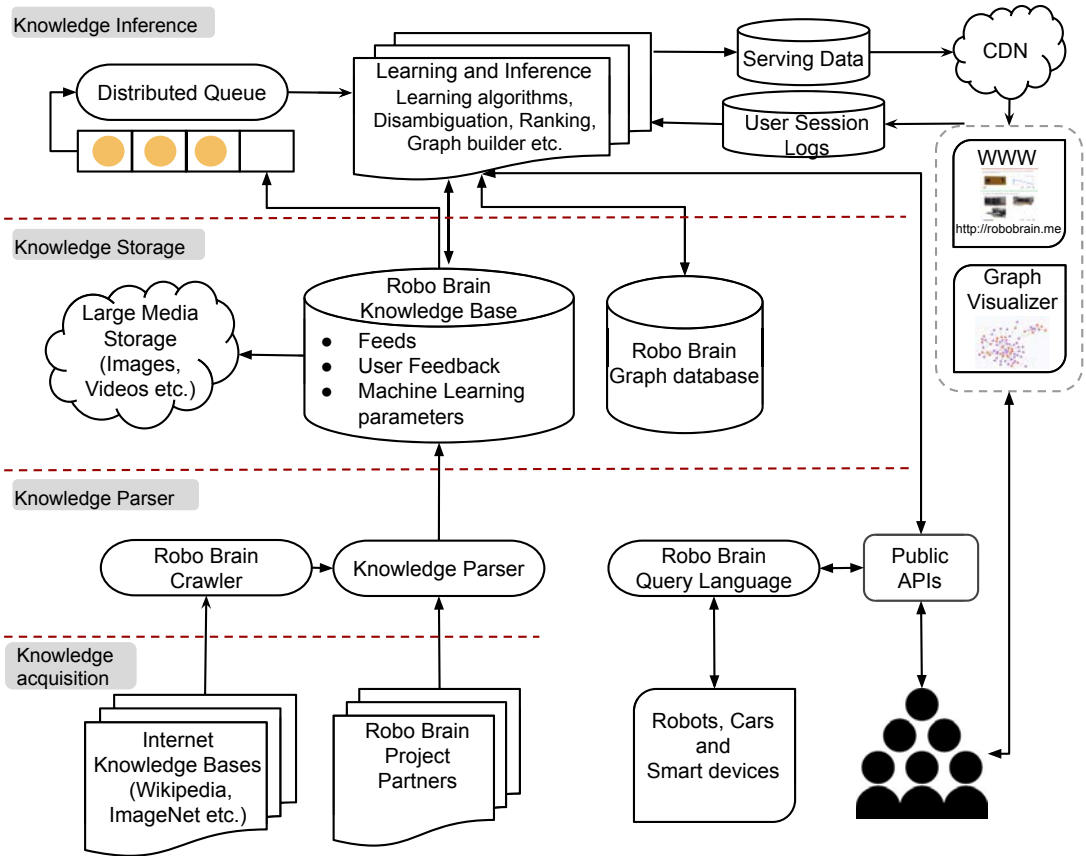


Figure 2.4: **RoboBrain system architecture.** It consists of four interconnected knowledge layers and supports various mechanisms for users and robots to interact with RoboBrain.

version number etc., for scheduling and managing future data processing. Moreover, since the knowledge bases might change with time, it adds a back pointer to the original source.

Knowledge storage layer of RoboBrain is responsible for storing different representations of the data. In particular it consists of a NoSQL document storage database cluster – RoboBrain Knowledge Base (RoboBrain-KB) – to store "feeds" parsed by the knowledge parser, crowd-sourcing feedback from users, and parameters of different machine learning algorithms provided by RoboBrain project part-

ners. RoboBrain-KB offloads large media content such as images, videos and 3D point clouds to a distributed object storage system built using Amazon Simple Storage Service (S3). The real power of RoboBrain comes through its graph database (RoboBrain-GD) which stores the structured knowledge. The data from RoboBrain-KB is refined through multiple learning algorithms and its graph representation is stored in RoboBrain-GD. The purpose behind this design is to keep RoboBrain-KB as the RoboBrain's single source of truth (SSOT). SSOT centric design allows us to re-build RoboBrain-GD in case of failures or malicious knowledge sources.

Knowledge inference layer contains the key processing and machine learning components of RoboBrain. All the new and recently updated feeds go through a persistent replicated distributed queuing system (Amazon SQS), which are then consumed by some of our machine learning plugins (inference algorithm, graph builder, etc.) and populates the graph database. These plugins along with other learning algorithms (operating on the entire graph) constitute our learning and inference framework.

RoboBrain supports various *interaction mechanisms* to enable robots and users to communicate with the knowledge engine. We develop a Robot Query Library as a primary method for robots to interact with RoboBrain. We also make available a set of public APIs to allow information to be presented on the WWW for online learning mechanisms (eg., crowd-sourcing). RoboBrain serves all its data using a commercial content delivery network (CDN) to reduce the end user latency.

2.6 Robot Query Library (RQL)

In this section we present the RQL query language, through which the robots use RoboBrain for various robotic applications. The RQL provides a rich set of *retrieval functions* and *programming constructs* to perform complex traversals on the RoboBrain graph. An example of such a query is finding the possible ways for humans to use a cup. This query requires traversing paths from the human node to the cup node in the RoboBrain graph.

The RQL allows expressing both the *pattern* of sub-graphs to match and the *operations* to perform on the retrieved information. An example of such an operation is *ranking* the paths from the human to the cup node in the order of relevance. The RQL admits following two types of functions: (i) graph retrieval functions; and (ii) programming construct functions.

2.6.1 Graph retrieval function

The graph retrieval function is used to find sub-graphs matching a given *template* of the form:

$$\text{Template: } (u) \rightarrow [e] \rightarrow (v)$$

In the template above, the variables u and v are nodes in the graph and the variable e is a directed edge from u to v. We represent the graph retrieval function with the keyword `fetch` and the corresponding RQL query takes the following form:

`fetch`(Template)

The above RQL query finds the sub-graphs matching the template. It instantiates the variables in the template to match the sub-graph and returns the list of

instantiated variables. We now give a few use cases of the retrieval function for RoboBrain.

Example 1 *The RQL query to retrieve all the objects that a human can use*

$$\textit{Template: } (\{\texttt{name} : \text{`}\texttt{Human}'\}) \rightarrow [\text{`}\texttt{CanUse}'] \rightarrow (\texttt{v})$$

$$\textit{Query: } \texttt{fetch}(\textit{Template})$$

The above query returns a list of nodes that are connected to the node with name `Human` *and with an edge of type* `CanUse`*.*

Using the RQL we can also express several *operations* to perform on the retrieved results. The operations can be of type `SortBy`, `Len`, `Belief` and `ArgMax`. We now explain some of these operations with an example.

Example 2 *The RQL query to retrieve and sort all possible paths from the* `Human` *node to the* `Cup` *node.*

$$\texttt{paths} := \texttt{fetch}(\{\texttt{name} : \text{`}\texttt{Human}'\}) \rightarrow [\texttt{r}*] \rightarrow (\{\texttt{name} : \text{`}\texttt{Cup}'\})$$

$$\texttt{SortBy}(\lambda \texttt{P} \rightarrow \texttt{Belief P})\,\texttt{paths}$$

In the example above, we first define a function `paths` *which returns all the paths from the node* `Human` *to the node* `Cup` *in the form of a list. The* `SortBy` *query first runs the* `paths` *function and then sorts, in decreasing order, all paths in the returned list using their beliefs.*

2.6.2 Programming construct functions

The programming construct functions serve to process the sub-graphs retrieved by the graph retrieval function `fetch`. In order to define these functions we make use of functional programming constructs like `map`, `filter` and `find`. We now explain the use of some of these constructs in RQL.

Example 3 *The RQL query to retrieve affordances of all the objects that* `Human` *can use.*

```
objects := fetch({name : 'Human'}) → ['CanUse'] → (v)
affordances n := fetch({name : n}) → ['HasAffordance'] → (v)
map(λu → affordances u) objects
```

In this example, we illustrate the use of `map` *construct. The* `map` *takes as input a function and a list, and then applies the function to every element of the list. More specifically, in the example above, the function* `objects` *retrieves the list of objects that the human can use. The* `affordances` *function takes as input an object and returns its affordances. In the last RQL query, the* `map` *applies the function* `affordances` *to the list returned by the function* `objects`*.*

We now conclude this section with an expressive RQL query for retrieving *joint parameters* shared among nodes. Parameters are one of the many concepts we store in RoboBrain and they represent learned knowledge about nodes. The algorithms use joint parameters to relate multiple concepts and here we show how to retrieve

joint parameters shared by multiple nodes. In the example below, we describe the queries for parameter of a single node and parameter shared by two nodes.

Example 4 *The RQL query to retrieve the joint parameters shared between a set of nodes.*

```
parents n := fetch (v) → ['HasParameters'] → ({handle : n})

parameters n := fetch ({name : n}) → ['HasParameters'] →(v)

ind_parameters a :=

        filter(λu →Len parents u = 1)parameters a

joint_parameters a1 a2 :=

        filter(λu → Len parents u = 2 and

        u in parameters a2) parameters a1
```

The query above uses the `filter` *construct function and* `Len` *operation. The* `filter` *takes as input a list and a check condition, and returns only those items from the list that satisfies the input condition. The* `Len` *takes as input a list and returns the number of items in the list. In the query above, we first define a function* `parents` *which for a given input node returns its parent nodes. Then we define a function* `parameters` *which for a given input node returns its parameters. The third and the fourth queries are functions accepting one and two input nodes, respectively, and return the (joint) parameters that share an edge with every input node and not with any other node.*

2.7 Applications

In this section we first show how RoboBrain can be used *as-a-service* by the robots for several robotics problems. Specifically, we explain the usage of RoboBrain in

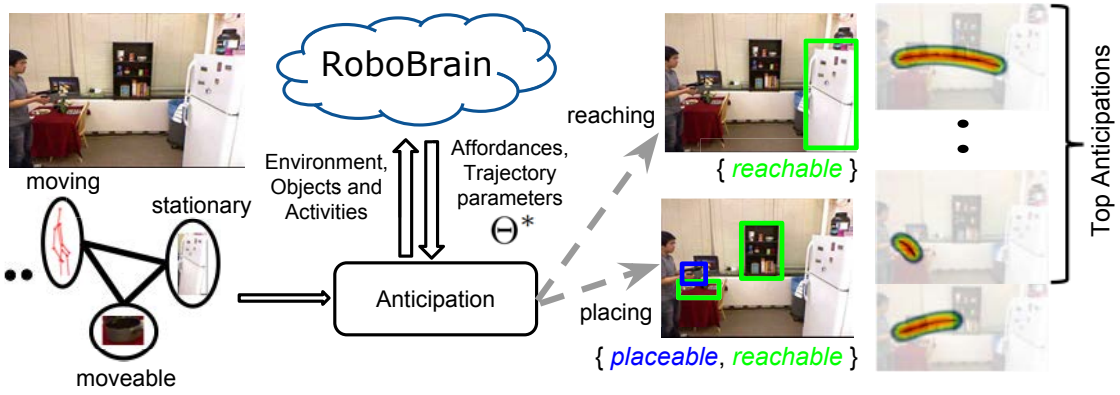


Figure 2.5: **RoboBrain for anticipating human activities.** Robot using anticipation algorithm of [103] queries RoboBrain, for the activity, affordance and trajectory parameters in order to generate and rank the possible future activities in a given environment.

anticipating human activities, grounding of natural language sentences, and path planning. We then show how RoboBrain can help robotics projects by sharing knowledge within the projects and throughout the Internet.

2.7.1 RoboBrain *as-a-service*

Our goal with providing RoboBrain *as-a-service* is to allow robots to use the representations learned by different partner projects. This allows RoboBrain to effortlessly address many robotics applications. In the following we demonstrate RoboBrain as-a-service feature for three robotics applications that deal with different data modalities of perception, natural language and trajectories.

Anticipating human actions

The assistive robots working with humans should be able to understand human activities and also anticipate the future actions that the human can perform. In order to anticipate, the robot should reason over the action possibilities in the environment, i.e., *object affordances*, and how the actions can be performed, i.e., *trajectories*. Several works in robotics have addressed the problem of anticipation [95, 103, 106].

We now show how robots can query RoboBrain and use the previous work by Koppula et al. [103] for anticipating human actions. In order to anticipate the future human actions, the authors [103] learn parameters using their anticipatory algorithm, and using the learned parameters they anticipate the most likely *future* object affordances and human trajectories. RoboBrain serves anticipation *as-a-service* by storing those learned parameters, object affordances and trajectories as concepts in its graph. Figure 2.5 illustrates a robot retrieving relevant information for anticipation. The robot first uses the following queries to retrieve the possible trajectories of an object:

$$
\begin{aligned}
&\texttt{affordances n} := \texttt{fetch}\ (\{\texttt{name} : \texttt{n}\}) \rightarrow [\text{`}\texttt{HasAffordance}'] \rightarrow \\
&\qquad\qquad (\texttt{v}\{\texttt{src} : \text{`}\texttt{Affordance}'\}) \\
&\texttt{trajectories a} := \texttt{fetch}\ (\{\texttt{handle} : \texttt{a}\}) \rightarrow [\text{`}\texttt{HasParameters}'] \rightarrow \\
&\qquad\qquad (\texttt{v}\{\texttt{src} : \text{`}\texttt{Affordance}', \texttt{type} : \text{`}\texttt{Trajectory}'\}) \\
&\texttt{trajectory\_parameters o} := \\
&\qquad\qquad \texttt{map}(\lambda\texttt{a} \rightarrow \texttt{trajectories a})\ \texttt{affordances o}
\end{aligned}
$$

In the queries above, the robot first queries for the affordances of the object and then for each affordance it queries RoboBrain for the trajectory parameters. Having retrieved all possible trajectories, the robot uses the learned parameters [103] to anticipate the future human actions. Since the learned parameters are also stored in the RoboBrain graph, the robot retrieves them using the following RQL queries:

```
parents n := fetch (v) → ['HasParameters'] → ({handle : n})

parameters n := fetch ({name : n}) → ['HasParameters'] →
                (v{src : 'Activity'})

find_parameters a :=
                filter(λu →Len parents u = 1)parameters a

joint_parameters a1 a2  := filter(λu → Len parents u = 2
                and u in parameters a2) parameters a1
```

The queries above retrieves both independent and joint parameters for anticipating the object affordances and human activities. Detailed explanation of the query is given in Example 4 of Section 2.6

Grounding natural language

The problem of grounding a natural language instruction in an environment requires the robot to formulate an action sequence that accomplish the semantics of the instruction [188, 136, 65, 132]. In order to do this, the robot needs a variety of information. Starting with finding action verbs and objects in the instruction, the robot has to discover those objects and their affordances in the environment.

We now show the previous work by Misra et al. [136] using RoboBrain *as-a-service* in their algorithm. In order to ground a natural language instruction the

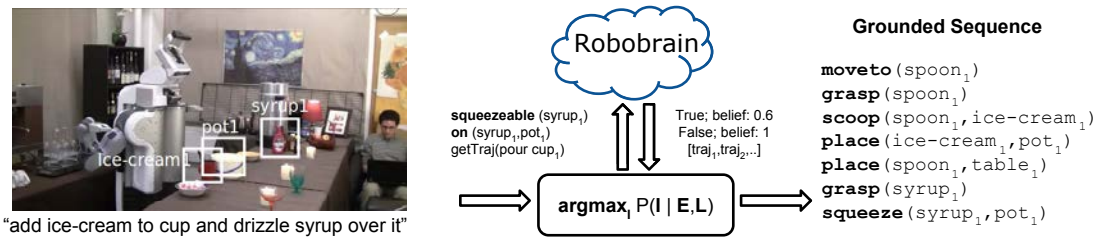


Figure 2.6: **Grounding natural language sentence.** The robot grounds natural language by using the algorithm by Misra et al. [136] and querying RoboBrain to check for the satisfiability of actions.

robot has to check for the satisfiability of the actions it generates in the given environment. For example, an action which pours water on a book should be deemed unsatisfiable. In the previous work [136], the authors manually define many preconditions to check the satisfiability of actions. For example, they define manually that a *syrup bottle* is *squeezable*. Such satisfiability depends on the object's affordances in the given environment, which can be retrieved from RoboBrain.

Figure 2.6 illustrates a robot querying RoboBrain to check the satisfiability of actions that it can perform in the given environment. Below is the RQL query for retrieving the satisfiability of *squeezable* action:

$$\texttt{squeezable syrup} := \texttt{Len fetch}\ (\texttt{u}\{\texttt{name} : \texttt{`syrup'}\}) \rightarrow$$
$$[\texttt{`HasAffordance'}] \rightarrow (\texttt{v}\{\texttt{name} : \texttt{`squeezable'}\}) > 0$$

2.7.2 RoboBrain for sharing knowledge

RoboBrain allows sharing the knowledge learned by different research groups as well as knowledge obtained from various internet sources. In this section we show

with experiments how sharing knowledge improves existing robotic applications:

New algorithms are commonly proposed for a problem to address the shortcomings of previous methods. These algorithms have their own learned representations. For example, different representations have been learned for grounding natural language [188, 136, 65, 132]. However, it is usually hard for practitioners to choose a single representation since there are always inputs where one representation fails but some others work. In this experiment we show that a robot can query RoboBrain for the best representation while being agnostic to the algorithmic details of the learned representations.

Simulating the above setting, we present an experiment for sharing multiple learned representations on a natural language grounding problem. Here the goal is to output a sequence of instructions for the robot to follow, given an input natural language command and an environment. Following the work by Misra et al. [136], we train a baseline algorithm for the task of *making ramen* (Algorithm A), and train their full algorithm for the task of *making affogato* (Algorithm B). These algorithms assign a confidence score (i.e., probability) to the output sequence of instructions. We store these learned representations as concepts in the RoboBrain graph, along with a prior belief over the correctness of the algorithms. The robot queries RoboBrain for a representation as follows:

$$
\begin{aligned}
&\texttt{algParam} := \texttt{fetch}(\texttt{u}\{\texttt{type} :' \texttt{GroundingAlgorithm}'\}) \rightarrow \\
&\qquad\qquad [\text{`}\texttt{HasParameters}'] \rightarrow (\texttt{v}) \\
&\texttt{prior n} := \texttt{fetch}(\{\texttt{name} : \texttt{n}\}) \rightarrow [\text{`}\texttt{HasPriorProb}'] \rightarrow (\texttt{v}) \\
&\texttt{groundings L, E} := \texttt{argMaxBy}(\lambda(\texttt{u}, \texttt{v}) \rightarrow \texttt{v}) \\
&\qquad\qquad \texttt{map}(\lambda(\texttt{u}, \texttt{v}) \rightarrow \texttt{u}(\texttt{L}, \texttt{E}, \texttt{v}) * \texttt{prior u})\ \ \texttt{algParam}
\end{aligned}
$$

Table 2.3: RoboBrain allows sharing learned representations. It allows the robot to query RoboBrain for a representation given an input natural language command. In this table the Algorithm A is a greedy algorithm based on Misra et al. [136], and Algorithm B is their full model. The IED metric measures the string-edit distance and the EED metric measures the semantic distance between the ground-truth and the inferred output instruction sequences. The metrics are normalized to 100 such that higher numbers are better.

Algorithm	**IED**	**EED**
Algorithm A	31.7	16.3
Algorithm B	23.7	**27.0**
RoboBrain (A+B)	**34.2**	24.2

In the `algParam` function, we retrieve all natural language grounding algorithms from the RoboBrain graph with their parameters. This returns a list in which each element is a tuple of algorithm u and its parameters v. The `prior` function retrieves the prior belief over the correctness of an algorithm. In order to ground a given natural language command `L` in environment `E`, the `grounding` function evaluates the likelihood score for each algorithm using their parameters as $\mathtt{u}(\mathtt{L}, \mathtt{E}, \mathtt{v})$. It further incorporates the prior belief over the algorithms, and returns the representation with the highest likelihood score. These set of queries corresponds to the following likelihood maximization equation:

$$\mathcal{I}^* = \underset{\mathcal{I}, m' \in \{A,B\}}{\arg\max} \; P(\mathcal{I}|E, L, w^*_{m'}, m')P(m')$$

As shown in the Table 2.3, choosing a representation by querying the RoboBrain achieves better performance than the individual algorithms.

2.8 Discussion and Conclusion

The RoboBrain graph currently has 44347 nodes (concepts) and 98465 edges (relations). The knowledge in the graph is obtained from the Internet sources and through the RoboBrain project partners. For the success of many robotics application it is important to relate and connect the concepts from these different knowledge sources. In order to empirically evaluate the connectivity of concepts in RoboBrain, we plot the degree distribution of the RoboBrain graph and compare it with the degree distribution of independent knowledge sources (Figure 2.7). The graph of independent knowledge sources is the union of each knowledge source, which have nodes from all the projects and the edges only between the nodes from the same project. As shown in the Figure 2.7, RoboBrain successfully connects projects and increases the average degree per-node by 0.8. The RoboBrain graph

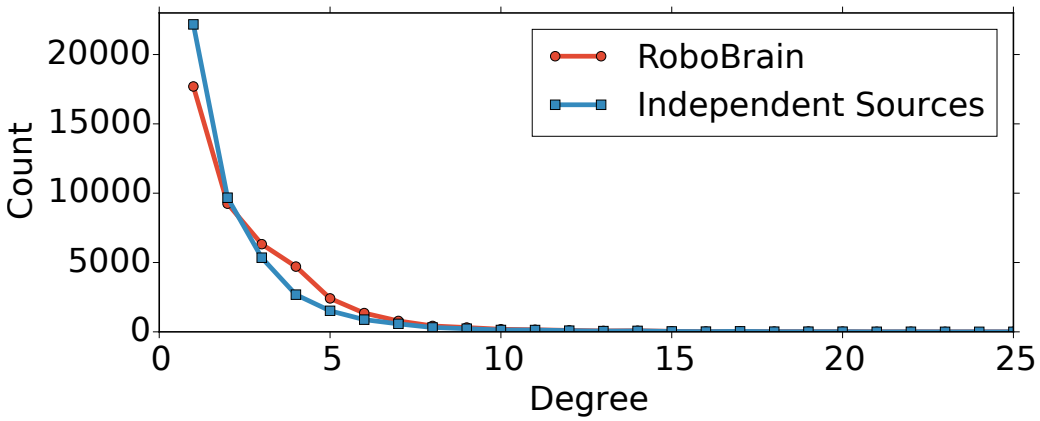


Figure 2.7: Degree distribution of RoboBrain and the union of independent knowledge sources. For the case of independent sources, we only consider the edges between nodes from the same source. RoboBrain connects different projects successfully: number of nodes with degree 1 and 2 decrease and nodes with degree 3 and more increase.

has fifteen thousand nodes with degree one. Most of the nodes with a single de-

gree come from the Internet sources such as Wikipedia and WordNet. These nodes are not directly related to the physical world and represent abstract concepts like political ideas, categories of art, etc.

In this chapter we described different aspects and technical challenges in building RoboBrain knowledge engine. RoboBrain represents multiple data modalities from various sources, and connects them to get an overall rich graph representation. We presented an overview of the RoboBrain large-scale system architecture and developed the Robot Query Library (RQL) for robots to use RoboBrain. We illustrated robotics applications of anticipation, natural language grounding, and path planning as simple RQL queries to RoboBrain. We also showed in experiments that sharing knowledge through RoboBrain improves existing path planning and natural language grounding algorithms. RoboBrain is an ongoing effort where we are collaborating with different research groups. We are working on improving different aspects such as learning from crowd-sourcing feedback, inference methods over the graph for discovering new relations between concepts, and expanding RoboBrain to new robotics applications.

CHAPTER 3

PERCEPTION THROUGH TIME - RECURSIVE CONDITIONAL RANDOM FIELDS

Time moves in one direction,
memory in another.

William Gibson

Understanding humans is an important skill for robots working with humans. Robots not only need to detect the activity and pose of the human but also need to anticipate *what activities can a human possibly perform in the near future in which poses* in order to choose the right actions. Anticipation ability is especially important for assistive robots, and we have recently seen many successful collaborative robotics applications [199, 128, 99] using the most likely action(s) humans might take in near future. Although existing methods can successfully detect most likely activity and/or human pose, robots need to model uncertainty as well. In this chapter, we focus on estimating the set of all possible future states with their likelihoods.

Anticipation is a challenging task, and it requires us to model the relationships between several objects and the human(s) in the scene, as well as their temporal evolution. Although the modeling assumptions and model parameterization varies, the common approach [100, 86, 101, 111] is using Conditional Random Field (CRF) to represent the rich relations in the scene, and anticipating a single or a few most likely future states. Since the future is ambiguous, the most likely state might not be sufficient enough to assess the risk of each action. For example, consider a collaborative cooking scenario, the object that human is reaching is typically a

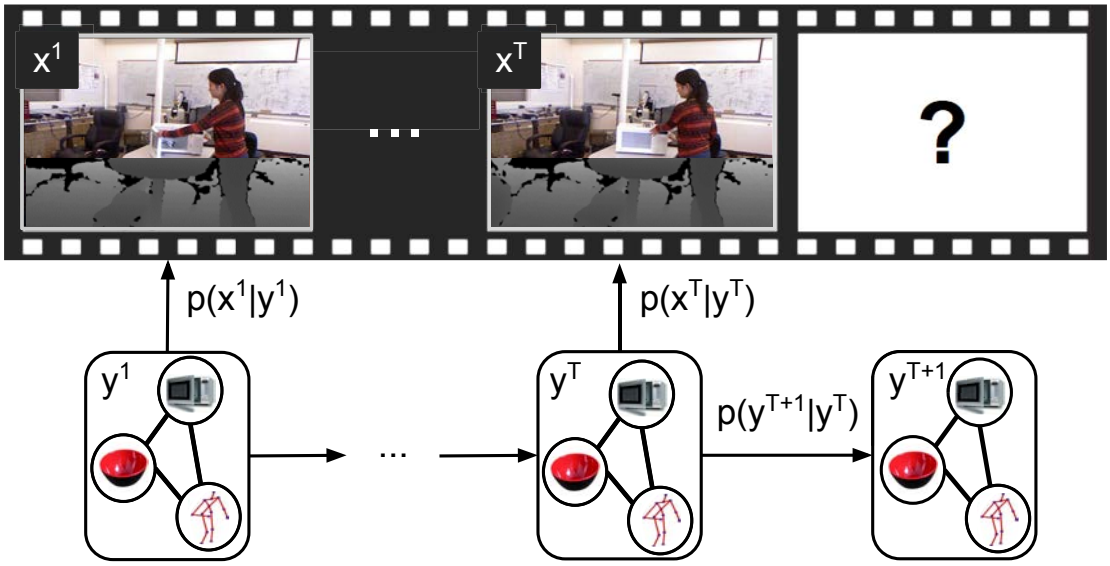


Figure 3.1: Figure is showing the state and measurements at each time represented by a CRF. Our algorithm, rCRF, enables the application of recursive Bayesian estimation to CRF-based scene models. rCRF computes the full belief over human activity and object affordances $(y^1, \ldots, y^{T+M})$ by using RGB-D Video $(x^1, \ldots, x^T)$.

distribution over many objects. Computing the trajectory, that is least likely to conflict with the human, is only possible via consideration of all future possibilities. The question, we address in this chapter, is: *How can we estimate all plausible future activities and their probabilities in a scene modeled by a CRF?*

Bayesian filtering methods can accurately estimate a belief (set of probabilities) over variables of interest from sequential data. However, it is still very challenging to estimate a belief over a CRF for two reasons. Firstly, it is not tractable to enumerate the labels over a CRF model since the output space has a dimension exponential in the number of objects, labels, and the temporal length[1]. Secondly, there is a modeling difference between CRFs and Bayesian filtering framework.

[1]Typically with 10 objects, 10 min. length (with 1 sec. long segments), 10 activity and 10 object labels, dimension is $(10^{10} \times 10)^{10 \times 60} = 10^{6600}$.

CRF is based on a discriminative setting whereas the Bayesian filtering mostly relies on the generative formulation.

In this chapter, we present a recursive algorithm – Recursive CRF (rCRF) – that can efficiently estimate a full belief over a CRF-based temporal scene model. rCRF can be seen as an efficient belief estimation method which enables us to use CRF-based scene model in Bayesian filtering. It models the temporal evolution via Bayesian updates and models the measurements in the scene via CRF. In order to use CRFs in such a scenario, we solve two problems. First, we present an approximation to convert the discriminative likelihood of the CRF into a generative measurement equation. Second, we use structured diversity for tractable computation. To the best of our knowledge, rCRF is the only tractable method that can use a CRF-based scene model in a recursive Bayesian filtering.

We apply the rCRF to the problem of activity detection and anticipation from RGB-D data. As a CRF-based scene model, we use the model from [102], which represents the scene as a CRF over human activity and object affordances. We then use the RGB-D video to detect and anticipate activities via rCRF.

Our experiments show that we outperform the state-of-the-art methods for detection and anticipation, and the improvement in the anticipation accuracy is significant. In addition to the improvements in accuracy, we show that our anticipation also improves the computation time and runs near real-time.

In summary, the contributions of this work are:

- We present Recursive-CRF (rCRF) method that uses the rich modeling power of CRF in Bayesian filtering setting.
- We present a structured-diversity based approach to enable tractable com-

putation of the belief.

- We apply our rCRF method to the problem of activity detection and anticipation in RGB-D videos.

3.1 Related Work on Graphical Models for Robot Perception

Bayesian Recursive Filtering: Estimating a belief over variables of interest from partial observations is a widely studied problem [190]. Sequential Monte Carlo (SMC) —aka *particle filter*— is typically used to estimate beliefs in high-dimensional cases. SMC methods represent the belief as a set of samples and we refer the reader to [32] for rigorous analysis.

SMC methods are not directly applicable to spaces like CRF since the number of samples required is intractably high. One solution to this problem is the Rao-Blackwellised particle filter [38]. It uses a partition of the state variables $\mathbf{y}$ into two set of variables $\mathbf{y_1}$ and $\mathbf{y_2}$ such that the variables in one partition $\mathbf{y_2}$ can be estimated using the partition $\mathbf{y_1}$. Then Rao-Blackwellised particle filter [38] estimates the $\mathbf{y_1}$ via SMC and directly estimates $\mathbf{y_2}$ using $\mathbf{y_1}$. However, for our problem, we are not aware of any state decomposition, which enables Rao-Blackwellised particle filter. Although there are discriminative extensions of Bayesian models like recursive least squares [165], in this chapter we only consider the states represented by CRFs. Moreover, we are not aware of any Bayesian smoothing formulation applied over CRFs.

One tractable application of the SMC framework to the CRF based scene anal-

ysis problems is the ATCRF [100] model. ATCRF [100] uses a set of heuristics to sample the particles. However, ATCRF faces the problem of computational limitations and requires computationally intractable number of samples for anticipation. We follow the Bayesian filtering theory and efficiently estimate the belief.

Structured Diversity and Variants of CRFs: CRFs are widely used to solve activity analysis problems [177, 151] in a discriminative setting. CRF models the conditional likelihood of the state given the observations, and the MAP solution can be found. Although this setting is powerful, it does not give any information about the belief other than the MAP state.

Other than the MAP solution, it is also tractable to compute the modes of the CRF [12, 116, 27]. These modes can be considered as an approximate state space, and the belief can be computed only for them. Indeed, this claim is empirically validated in many problems like parameter learning [134], empirical MBR [150] and discriminative re-ranking [203].

Among the aforementioned approaches, Div-M-Best [12] is a method applicable to the sequential information. [12] starts by dividing the video into a set of frames and computes the diverse-most-likely solutions of each frame independently. Then, it combines the results via the temporal relations. On the contrary, we formulate the problem as recursive Bayesian smoothing and compute the samples based on temporal relations. Formally, given state variables $\mathbf{y}^1, \ldots, \mathbf{y}^T$ and observations $\mathbf{x}^1, \ldots, \mathbf{x}^T$, we directly sample $p(\mathbf{y}^t|\mathbf{x}^1, \ldots, \mathbf{x}^T)$, whereas, [12] samples $p(\mathbf{y}^t|\mathbf{x}^t)$. Since our sampling procedure uses the entire video, our samples are more accurate.

There are variants of CRFs that rely on sequential models as well such as, Dynamic CRF (dCRF) [185], Infinite Hidden CRF [21], Gaussian Process Latent

CRF [86] and Hierarchical Semi-Markov CRF (HSCRF). Although they are applicable to videos, we are not aware of any tractable method to compute a belief over any of the aforementioned graphical model.

DCRF [198] learns the observation likelihood –$p(\mathbf{x^t}|\mathbf{y^t})$– by using the low-dimensional nature of the features and follows Bayesian filtering. Since our features have very high-dimension (for N objects, we have $58N + 20N^2 + 103$ dimensional features), DCRF [198] is not directly applicable. However, it is possible to learn $p(\mathbf{y^t}|\mathbf{x^t})$ and *approximately* use the DCRF formulation by assuming observation and label likelihoods are equal. Moreover, This approach can be shown equivalent to finding local maximum of energy function defined by [102] following the formulation of Fox et al [52].

It is also common to compute a belief over latent nodes as in the case of infinite hidden CRF [21] and Gaussian Process Latent CRF[86]. However, they are not directly applicable to our problem since they can compute a belief only over the latent node. CRF-Filter [124] is a closely related approach which uses CRFs in a particle filtering scenario. However, it is based on sampling of a low dimensional state space and it is not applicable to our rich model either.

Human Activity Detection and Anticipation: Early works relied solely on human poses. These works range from jointly segmenting and recognizing sub-activities [72, 174] to choosing a relevant model out of activity models [131]. Main limitation of these methods is that they do not use the object information. Some methods successfully model and use the relations of the human-poses and objects in the scene [66, 204, 85, 83]. However, a significant drawback of these works is missing the fact that object affordance is more important than object types for activities [57]. Indeed, object-affordance based models had higher performance

(*e.g.* [102]). A recent work modeled human activities with latent models [75] and also handled the disagreements among the activity annotations [76].

Another drawback of these methods is the requirement of the entire activity. Detecting the activity in its early stages is especially crucial for assistive robotics and surveillance systems. Although a few recent works address the problem of activity detection with partial/early information [71, 163], these works do not perform anticipation. There are a few recent works addressing *what human will perform next* by using trajectory prediction. It is possible to predict the trajectory of the human using inverse reinforcement learning in 2D [212, 107, 96] or 3D [40]. However, these models rely on the low-dimensional structure of the 2D/3D coordinate space and therefore they do not apply to rich models like CRF.

Recent work on anticipatory temporal CRF [100] considers anticipation with a CRF model. It anticipates the future via augmenting set of possible future observations to the CRF. It is also extended with an improved human motion model based on a Gaussian process [86]. However, their accuracy significantly drops for a long anticipation horizon since they fail to represent the uncertainty. Our method overcomes these problems by recursively estimating a full belief.

3.2 Overview of rCRF

In this section, we summarize our method and explain how we estimate the full belief over the activities and object affordances. Moreover, we also give an illustrative example of the rCRF with a toy scene consisting of two objects (a microwave and a bowl) and a human in Figure 3.2.

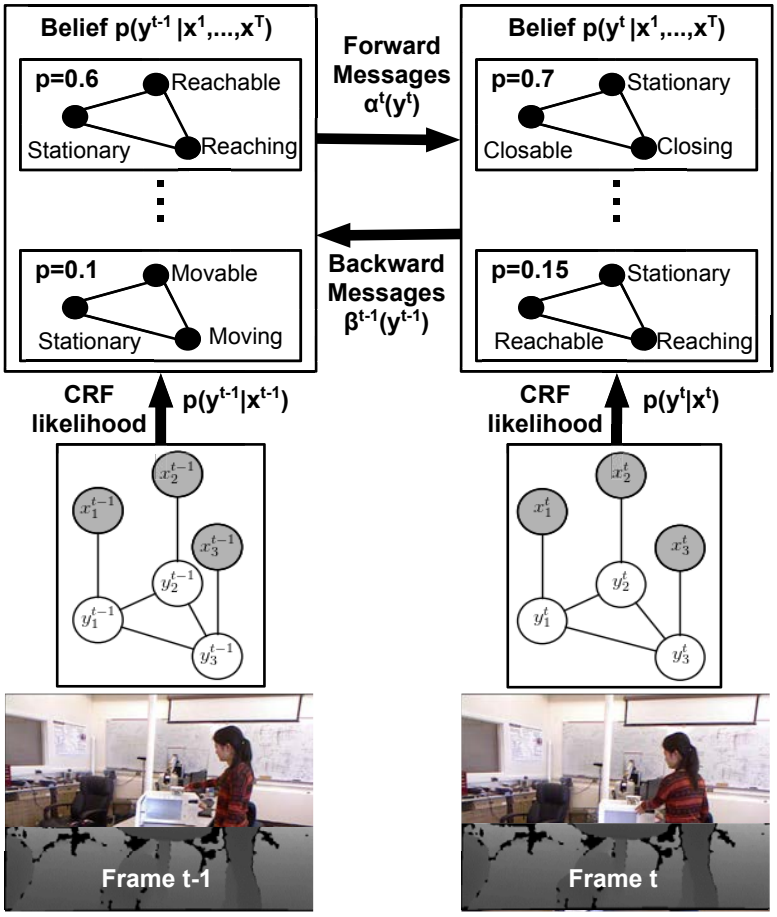


Figure 3.2: **Computing the full belief by using rCRF.** Each iteration of the recursive estimation algorithm includes computing forward and backward messages, $\alpha^t(\mathbf{y}^t)$ and $\beta^{t-1}(\mathbf{y}^{t-1})$, by using the current samples and computing the belief $p(\mathbf{y}^t|\mathbf{x}^1, \ldots, \mathbf{x}^T)$ with the computed messages. Then, we re-compute the messages and re-sample the belief until the belief converges. *Here, we only have two objects as* $\mathbf{y}^t = (\mathbf{O}_1^t, \mathbf{O}_2^t, \mathbf{A}^t)$ *and* $\mathbf{x}^t = (\mathbf{L}_1^t, \mathbf{L}_2^t, \mathbf{H}^t)$

Reasoning about activities requires not only identifying the objects but also interpreting object-object relations and human-object relations. Indeed, we capture such rich information via CRF. As shown in the Figure 3.2, each object and a human corresponds to a node in the graph on which we define the CRF. As a hidden variable, we are interested in object affordances such as *openable, graspable, movable, etc.* and the activity human is performing such as *moving, opening, grasping,*

etc.. We define the affordances as the actions that can be performed on/with the object [57]. We denote the affordance variables at time t as $\mathbf{O}_1^t, \ldots, \mathbf{O}_N^t$ for N objects and the activity variable as $\mathbf{A}^t$. Since they are not directly observed, we estimate them by using partial observations. We are using the 3D positions of the objects $\mathbf{L}_1^t, \ldots, \mathbf{L}_N^t$ and the human pose $\mathbf{H}^t$ as observations. The input video is temporally over-segmented prior to the application of the belief estimation, and the time instant t represents the t^{th} segment of the video. We explain the features and the potential functions we use while defining the CRF in Section 3.4.1.

In addition to the spatial relations between objects and humans, we are also interested in their temporal evolution. In general, the problem of estimating a belief over set of hidden variables using the entire video corresponds to a Bayesian smoothing problem. Formally, we are interested in estimating states $\mathbf{y}^t = (\mathbf{O}_1^t, \ldots, \mathbf{O}_N^t, \mathbf{A}^t)$ given set of observations $\mathbf{x}^t = (\mathbf{L}_1^t, \ldots, \mathbf{L}_N^t, \mathbf{H}^t)$. We estimate the states through successive application of the recursive Bayesian updates. In order to tractably compute the Bayesian updates, we introduce two approximations in Section 3.3. First, we compute the set of all plausible future states by using structured-diversity. Second, we use Jensen inequality in order to convert the discriminative likelihood into a generative one. After the introduction of these two machineries, we follow the recursive Bayesian estimation framework. As shown in Figure 3.2, we first compute the Bayesian updates through the forward and backward messages, $\alpha^t(\mathbf{y}^t)$ and $\beta^t(\mathbf{y}^t)$. We then compute the posterior belief $p(\mathbf{y}^t|\mathbf{x}^1, \ldots, \mathbf{x}^T)$ by using the computed messages and the CRF-likelihood $p(\mathbf{y}^t|\mathbf{x}^t)$. As a final step of the iteration, we represent the belief via diverse samples of the posterior belief. Since the belief is recursively defined, we re-compute the messages and re-sample the belief until it converges.

3.3 Belief Estimation with rCRF

In this section, we develop the Recursive Conditional Random Field (rCRF) to use CRF in a Bayesian filtering setting. rCRF jointly uses rich model of CRF and the recursive nature of the Bayesian filtering to compute an accurate belief. We first define our modeling assumptions in Section 3.3.1, and then we introduce a link between the CRF likelihood and the measurement likelihood in Section 3.3.2 in order to compute the posterior belief. In Section 3.3.2, we further show that the resulting posterior belief is equivalent to a CRF. Moreover, this equivalence enables efficient computation via the diversity based method [12] developed for CRFs.

3.3.1 Recursive Conditional Random Field

Consider a sequential estimation problem in which we are interested in variables $\mathbf{y}^t$ using observations $\mathbf{x}^t$ where t is the temporal variable. In our application, t is the temporal segment id. We note RGB-D camera reading as $\mathbf{x}^t$, and object and activity labels as $\mathbf{y}^t$. We now define the Recursive Conditional Random Field (rCRF) framework for such a problem following the assumptions of Hidden Markov Models.

Definition 1 *Let $\mathcal{G}^t = (V^t, E^t)$ be set of graphs indexed by the temporal variable t and $\mathbf{y}^t$ is indexed by the vertices of $\mathcal{G}^t$ as $\mathbf{y}^t = (y_v^t)_{v \in V^t}$. Then, ($\mathbf{x}^{1...T}$,$\mathbf{y}^{1...T}$) is a* ***Recursive Conditional Random Field*** *with dynamics $p_v(\cdot|\cdot)$ when*

1. *For each t, $(\mathbf{y}^t, \mathbf{x}^t)$ is a CRF over $\mathcal{G}^t = (V^t, E^t)$*

2. $p(\mathbf{y}^t|\mathbf{y}^1,\ldots,\mathbf{y}^{t-1}) = p(\mathbf{y}^t|\mathbf{y}^{t-1}) \quad \forall t$ *(Markov)*

3. $p(\mathbf{x}^t|\mathbf{y}^1,\ldots,\mathbf{y}^t,\mathbf{x}^1,\ldots,\mathbf{x}^{t-1}) = p(\mathbf{x}^t|\mathbf{y}^t) \quad \forall t$

4. $p(\mathbf{y}^t = \mathbf{y}|\mathbf{y}^{t-1} = \mathbf{y}') = p_v(\mathbf{y}|\mathbf{y}')$ *(stationarity)*

■

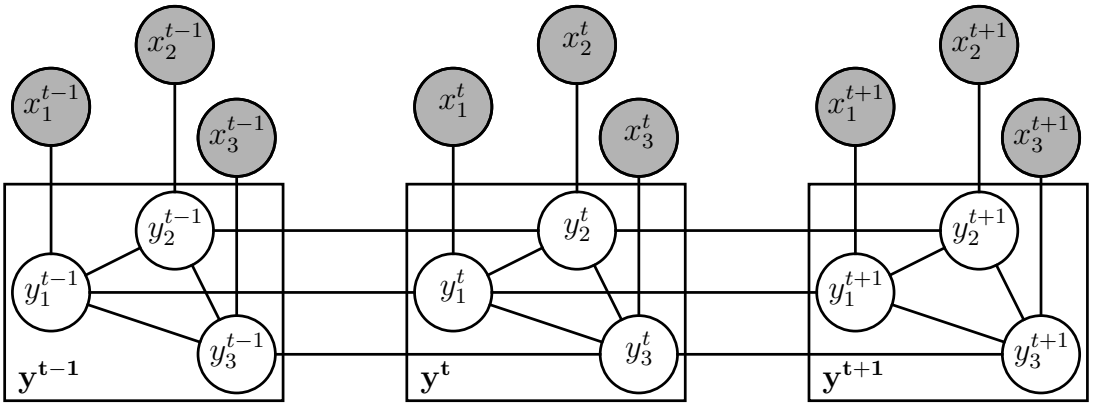


Figure 3.3: **rCRF is defined over a temporal CRF.** The graphical model, we use within the rCRF, is a temporal CRF with additional constraints. We impose a special structure through the conditions we state in the definition. For the visualization purposes, we show there nodes per segment although rCRF can handle any number of nodes.

We visualize the graphical model representation of the rCRF in Figure 3.3. In this work, we are interested in the belief over state variables at a given time instant t as:

$$bel^t(\mathbf{y}) = p(\mathbf{y}^t = \mathbf{y}|\mathbf{x}_1,\ldots,\mathbf{x}_T) \tag{3.1}$$

Here, T denotes the length of the video. Hence, in rCRF the belief of any frame is supported by the entire video. Moreover, the time instant t can be greater than the video length T as well. Hence, rCRF naturally supports anticipation setting.

We then decompose the belief by using the independence properties of the rCRF as:

$$bel^t(\mathbf{y}) \propto \underbrace{p(\mathbf{y}^t = \mathbf{y}|\mathbf{x}^1, \ldots, \mathbf{x}^t)}_{\alpha^t(\mathbf{y})} \underbrace{p(\mathbf{x}^{t+1}, \ldots, \mathbf{x}^T|\mathbf{y}^t = \mathbf{y})}_{\beta^t(\mathbf{y})} \tag{3.2}$$

Moreover, α^t and β^t can be computed recursively by using forward and backward messages. Following [153],

$$\begin{aligned} \alpha^t(\mathbf{y}^t) &= p(\mathbf{x}^t|\mathbf{y}^t) \sum_{\mathbf{y}^{t-1}} \alpha^{t-1}(\mathbf{y}^{t-1}) p(\mathbf{y}^t|\mathbf{y}^{t-1}) \\ \beta^t(\mathbf{y}^t) &= \sum_{\mathbf{y}^{t+1}} p(\mathbf{x}^{t+1}|\mathbf{y}^{t+1}) \beta^{t+1}(\mathbf{y}^{t+1}) p(\mathbf{y}^{t+1}|\mathbf{y}^t) \end{aligned} \tag{3.3}$$

with initializations $\alpha^1(\mathbf{y}^1) = p(\mathbf{x}^1|\mathbf{y}^1)$ and $\beta^T(\mathbf{y}^T) = 1$.

3.3.2 Computing the belief using an rCRF

Recursive definition in (3.3) has two significant drawbacks: firstly, CRF is modeling $p(\mathbf{y}^t|\mathbf{x}^t)$ instead of $p(\mathbf{x}^t|\mathbf{y}^t)$ and the transformation is not trivial. Secondly, computation of the messages requires a summation over the entire output space, and it has an exponential dimension. In this section, we first compute the posterior of the observation given labels $p(\mathbf{x}^t|\mathbf{y}^t)$ by using the CRF posterior likelihood $p(\mathbf{y}^t|\mathbf{x}^t)$. Then, we show that the belief function at time t, $bel^t(\mathbf{y})$, can be approximately represented as a Gibbs measure over $\mathcal{G}^t$. Then, we conclude that the belief, $bel^t(\mathbf{y})$, is a CRF over the graph $\mathcal{G}^t$ with modified energy functions.

From $p(\mathbf{y}^t|\mathbf{x}^t)$ to $p(\mathbf{x}^t|\mathbf{y}^t)$

Since $(\mathbf{x}^t, \mathbf{y}^t)$ is a CRF, the posterior of the label given the observation follows [56];

$$p(\mathbf{y}^t|\mathbf{x}^t) \propto \exp\left(\sum_{i\in V^t} \theta_{x_i^t}(y_i^t) + \sum_{i,j\in E^t} \theta_{x_i^t,x_j^t}(y_i^t, y_j^t)\right) \tag{3.4}$$

where θ is the energy function defined over the node set $v \in V^t$ as θ_v and over the edge set $(u, v) \in E^t$ as $\theta_{u,v}$.

In order to transform $p(\mathbf{y}^t|\mathbf{x}^t)$ into $p(\mathbf{x}^t|\mathbf{y}^t)$, we use Bayes rule; $p(\mathbf{x}^t|\mathbf{y}^t) \propto \frac{p(\mathbf{y}^t|\mathbf{x}^t)}{\sum_{x^t} p(\mathbf{y}^t|\mathbf{x}^t)p(\mathbf{x}^t)}$ and compute $p(\mathbf{y}^t)$ as;

$$p(\mathbf{y}^t) = \sum_{\mathbf{x}^t} \exp\left(\sum_{i\in V^t} \theta_{x_i^t}(y_i^t) + \sum_{i,j\in E^t} \theta_{x_i^t,x_j^t}(y_i^t, y_j^t)\right) p(\mathbf{x}^t) \tag{3.5}$$

For tractability, we approximate the $p(\mathbf{y}^t)$ with its lower bound after applying the Jensen inequality as;

$$p(\mathbf{y}^t) \approx \exp(\sum_{i\in V^t} \underbrace{\sum_{\mathbf{x}^t} \theta_{x_i^t}(y_i^t)p(\mathbf{x}^t)}_{\tilde{\theta}(y_i^t)} + \sum_{i,j\in E^t} \underbrace{\sum_{\mathbf{x}^t} \theta_{x_i^t,x_j^t}(y_i^t, y_j^t)p(\mathbf{x}^t)}_{\tilde{\theta}(y_i^t,y_j^t)}) \tag{3.6}$$

We then estimate the inner summations $\tilde{\theta}(\cdot)$ from the training data using Monte Carlo method as $\tilde{\theta}(\cdot) = \frac{1}{N}\sum_{i=1}^{N} \theta_{\mathbf{x}^{(i)}}(\cdot)$ where N is the number of training samples and $\mathbf{x}^{(i)}$ is the i^{th} training sample. Therefore, we can compute the observation likelihood as: $p(\mathbf{x}^t|\mathbf{y}^t) \propto$

$$\exp\left(\sum_{i\in V^t} \theta_{x_i^t}(y_i^t) - \tilde{\theta}(y_i^t) + \sum_{i,j\in E^t} \theta_{x_i^t,x_j^t}(y_i^t, y_j^t) - \tilde{\theta}(y_i^t, y_j^t)\right) \tag{3.7}$$

Belief is a CRF

Here we compute the belief (3.2) in terms of forward and backward messages and CRF likelihood. We then show that the posterior belief is a CRF. This observation

enables us to use efficient methods developed for CRFs.

In order to compute the belief (3.2), we decompose the system dynamics using the independence assumption in the graph in Fig. 3.3. This gives us $p(\mathbf{y}^t|\mathbf{y}^{t-1}) = \prod_i p(y_i^t|y_i^{t-1})$. We then compute the belief function as $bel(\mathbf{y}^t) = \alpha^t(\mathbf{y}^t)\beta^t(\mathbf{y}^t)$ by using equations (3.3) and (3.7). After algebraic manipulations, the belief function can be approximated as follows:

$$
\begin{aligned}
bel(\mathbf{y}^t) \propto \exp \Bigg[& \sum_{i,j \in E^t} \left(\theta_{x_i^t, x_j^t}(y_i^t, y_j^t) - \tilde{\theta}(y_i^t, y_j^t) \right) \\
& \sum_{i \in V^t} \Bigg(\theta_{x_i^t}(y_i^t) - \tilde{\theta}(y_i^t) + \sum_{\mathbf{y}^{t-1}} \alpha^{t-1}(\mathbf{y}^{t-1}) \log p(y_i^t|y_i^{t-1}) \\
& + \frac{1}{\gamma} \sum_{\mathbf{y}^{t+1}} \beta^{t+1}(\mathbf{y}^{t+1}) p(\mathbf{x}^{t+1}|\mathbf{y}^{t+1}) \log p(y_i^{t+1}|y_i^t) \Bigg) \Bigg]
\end{aligned}
\tag{3.8}
$$

where $\gamma = \sum_{\mathbf{y}^{t+1}} \beta^{t+1}(\mathbf{y}^{t+1}) p(\mathbf{x}^{t+1}|\mathbf{y}^{t+1})$

One property to observe is the decomposition of the belief over the graph. Resulting belief function, (3.8), is a summation over energy terms defined over nodes $i \in V^t$ and edges $i, j \in E^t$. Hence, belief $bel^t(\cdot)$ is a Gibbs measure over $\mathcal{G}^t$. By using Hammersley-Clifford theorem [69], we conclude that the posterior belief in rCRF is also a CRF. In other words, belief is a CRF defined over the same graph with a modified energy.

Belief via Diverse-Most-Likely Samples

Since we computed the belief function and showed that it is equivalent to a CRF, we now need an efficient method for computing it.

We follow the observation that CRF-likelihood over a natural scene concen-

trates on a few diverse samples [12] because each scene only has a few plausible explanations. So, we compute the belief for only those samples. In other words, let's assume the set of all plausible solutions at time t is $\mathbf{Y}^t = \mathbf{y}^{t,1}, \ldots, \mathbf{y}^{t,M}$ where $\mathbf{y}^{t,i}$ is the i^{th} sample at time t. We then redefine the belief as;

$$\text{approx\_bel}^t(\mathbf{y}) = \begin{cases} \frac{bel^t(\mathbf{y})}{\sum_{\mathbf{y}' \in \mathbf{Y}^t} bel^t(\mathbf{y}')} & \text{if } \mathbf{y} \in \mathbf{Y}^t \\ 0 & \text{o.w.} \end{cases} \tag{3.9}$$

Since there are only a few plausible explanations of a visual observation and CRF-based belief concentrates only on those samples, proper selection of the samples $\mathbf{Y}^t$ is expected to work well in practice. These samples are typically selected as the diverse-most-likely solutions of the CRF. They are most-likely samples because we are only interested in the plausible explanations. They are diverse because we are interested in the modes of the CRF other than set of samples around the MAP solution. Diversity is achieved via asserting samples to be at least δ unit apart from each other via the distance function Δ (we use hamming distance as a in our experiments). In other words, we solve the following optimization problem in order to get the samples, which represent the belief;

$$\begin{aligned} \mathbf{y}^{t,i} &= \arg\max_{\mathbf{y}} bel^t(\mathbf{y}) \\ &s.t.\ \Delta(\mathbf{y}, \mathbf{y}^{t,j}) \geq \delta \quad \forall\ j < i \end{aligned} \tag{3.10}$$

This optimization is NP-hard in general; however, since we already showed $bel^t(\mathbf{y})$ is CRF, we use the existing diverse-m-best algorithms developed for CRFs. We use the Lagrange relaxation by Batra et al. [12].

We first take the logarithm of the objective function since log is a monotonic function. We then follow the Div-M-Best procedure [12]. Div-M-Best uses Lagrange relaxation after convertingthe $\Delta(\mathbf{y}, \mathbf{y}^{\mathbf{t,j}}) \geq \delta$ constraints into dual form.

Hence, the relaxed unconstrained optimization problem is,

$$\mathbf{y}^{t,i} = \arg\max_{\mathbf{y}} \log bel^t(\mathbf{y}) + \sum_{m=0}^{i-1} \lambda_m(\Delta(\mathbf{y}, \mathbf{y}^{t,m}) - \delta) \tag{3.11}$$

Please note that we use the hamming distance for $\Delta(\cdot,\cdot)$ within all of our experiments. Hence, we substitute the Hamming distance in the optimization objective with $\Delta(\cdot,\cdot)$. We further substitute the (3.8) as,

$$\begin{aligned}
\mathbf{y}^{t,i} = \arg\max_{\mathbf{y}} & \sum_{i,j \in E^t} \left(\theta_{x_i^t, x_j^t}(y_i^t, y_j^t) - \tilde{\theta}(y_i^t, y_j^t)\right) \\
& \sum_{i \in V^t} \Bigg(\theta_{x_i^t}(y_i^t) - \tilde{\theta}(y_i^t) + \sum_{m=0}^{i-1} \lambda_m \mathbb{1}_{y_i^t \neq y_i^{t,m}} \\
& + \frac{1}{\gamma} \sum_{\mathbf{y}^{t+1}} \beta^{t+1}(\mathbf{y^{t+1}}) p(\mathbf{x}^{t+1} | \mathbf{y}^{t+1}) \log p(y_i^{t+1} | y_i^t) \\
& + \sum_{\mathbf{y}^{t-1}} \alpha^{t-1}(\mathbf{y}^{t-1}) \log p(y_i^t | y_i^{t-1}) \Bigg)
\end{aligned} \tag{3.12}$$

Where $\mathbb{1}_A$ is an indicator function, and it is 1 when A is true and 0 otherwise. Thus, the final optimization problem in (3.12) is equivalent to finding the MAP solution of a CRF with modified energy function. Moreover, we solve it by using the original inference method (Mixed Integer Programming) following [102].

In summary, we first compute the belief via (3.8) for all frames by using samples of the previous and the next frame as well as CRF likelihoods. Then, we compute the diverse samples of (3.8) by using [12]. After computing the samples, we compute the messages α^t and β^t by using the equations (3.7) and (3.3). We continue to re-sample the beliefs and re-compute the messages recursively until the convergence. Moreover, during the initialization, we only sample the observation function (3.7) since the messages are not available.

3.4 Applications

3.4.1 Human Activity Detection and Anticipation

In this section, we describe how we apply the rCRF framework to RGB-D videos for human activity detection and anticipation. We are interested in activities such as *reaching* and *moving*, and object affordances such as *reachable* and *movable* as explained in Section 3.2. We follow the approach in [102], and start with temporally segmenting the video. This step can be considered as an oversegmentation in the temporal domain. It decreases the computation complexity and enables using motion information as an observation.

We then obtain the observations $\mathbf{x}^t = (L_1^t, L_2^t, H^t)$, by detecting the objects in the first frame and then tracking them. We obtain the human pose H^t through a skeleton tracker. We consider affordances and activities as state $\mathbf{y}^t = (O_1^t, \ldots, O_N^t, A)$ where N is the number of objects. We extracted set of features from the observations following the feature functions in [102] (*e.g.* relative and absolute location of objects, human joints and their temporal displacements). After extracting the features, we define our CRF as a log-linear CRF and learn the energy function defined in (3.4) by using the Structural SVM [194] as in the case of [102]. We use the first order statistics for temporal dynamics as $p_v(y, y') = p(Y_v^t = y | Y_v^{t-1} = y') = \frac{\#(Y_v^t = y, Y_v^{t-1} = y')}{\#(Y_v^t = y')}$ where $\#(\cdot, \cdot)$ is number of the co-occurrence in training data.

After defining the observation, state and dynamics, we apply the rCRF framework. We also summarize the activity detection and anticipation application in Algorithm 1.

Algorithm 1: Compute belief the over $(O^t_{1...N}, A^t)$ for $t \in [1, T+\tau]$ in an RGB-D Video of length T

Initialization:

Compute $L^t_1, \ldots, L^t_N$, and H^t for $t \in [1, T]$ via [102].

Compute $p(L^t_{1...N}, H^t | O^t_{1...N}, A^t)$ for $t \in [1, T]$ via (3.7)

Compute the belief via (3.8) w/o messages ($\alpha = 1, \beta = 1$)

Detection:

repeat

for $t \in [1, T]$ **do**

Compute the forward/backward messages via (3.3)

Compute the belief via (3.8) re-sample via (3.10)

end for

until convergence or number of iterations limit

Anticipation:

for $t \in [T+1, T+\tau]$ **do**

Compute only the forward messages via (3.3)

Sample the belief directly from the forward messages.

end for

Moreover, since the temporal relations are modeled as causal, we do not compute the backward messages during the anticipation. In anticipation, there is also no future observation. Hence, the belief is defined solely by the forward messages. In order to compute the belief for future frames, we propagate the estimated belief. We propagate the belief to the next frame by sampling the next state of the each sample in the belief of the current frame via the temporal dynamics. Then, we choose diverse most likely samples out of the propagated samples via solving (3.10) with exhaustive search.

3.4.2 Experimental Results on Activity Detection

In order to experimentally evaluate the proposed rCRF model and the belief computation, we perform experiments on two applications. Firstly, we estimate a belief over the activity a human is performing and the affordances of the objects in the scene by using the RGB-D video. After computing the belief, we detect the most likely activity and affordance sequences and study the improvement in the detection accuracy. Secondly, we test the accuracy of the beliefs in the anticipation setting. Indeed, we show that it is possible to obtain high-quality detection and anticipation via rCRF.

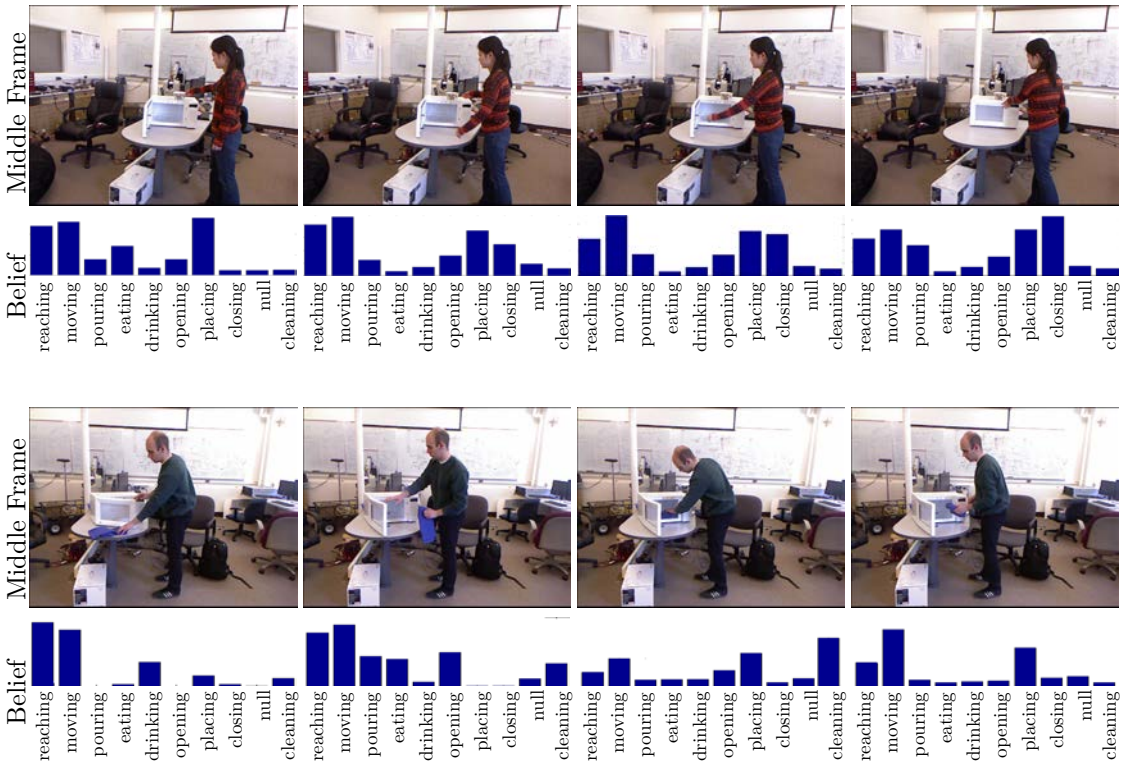


Figure 3.4: **Anticipated belief over activity.** In the first and third row, we show a middle frame of the temporal segment. In the second and fourth row, we show the anticipated belief we computed for the middle frame. Note that frames are not visible to the algorithm and only included for evaluation.

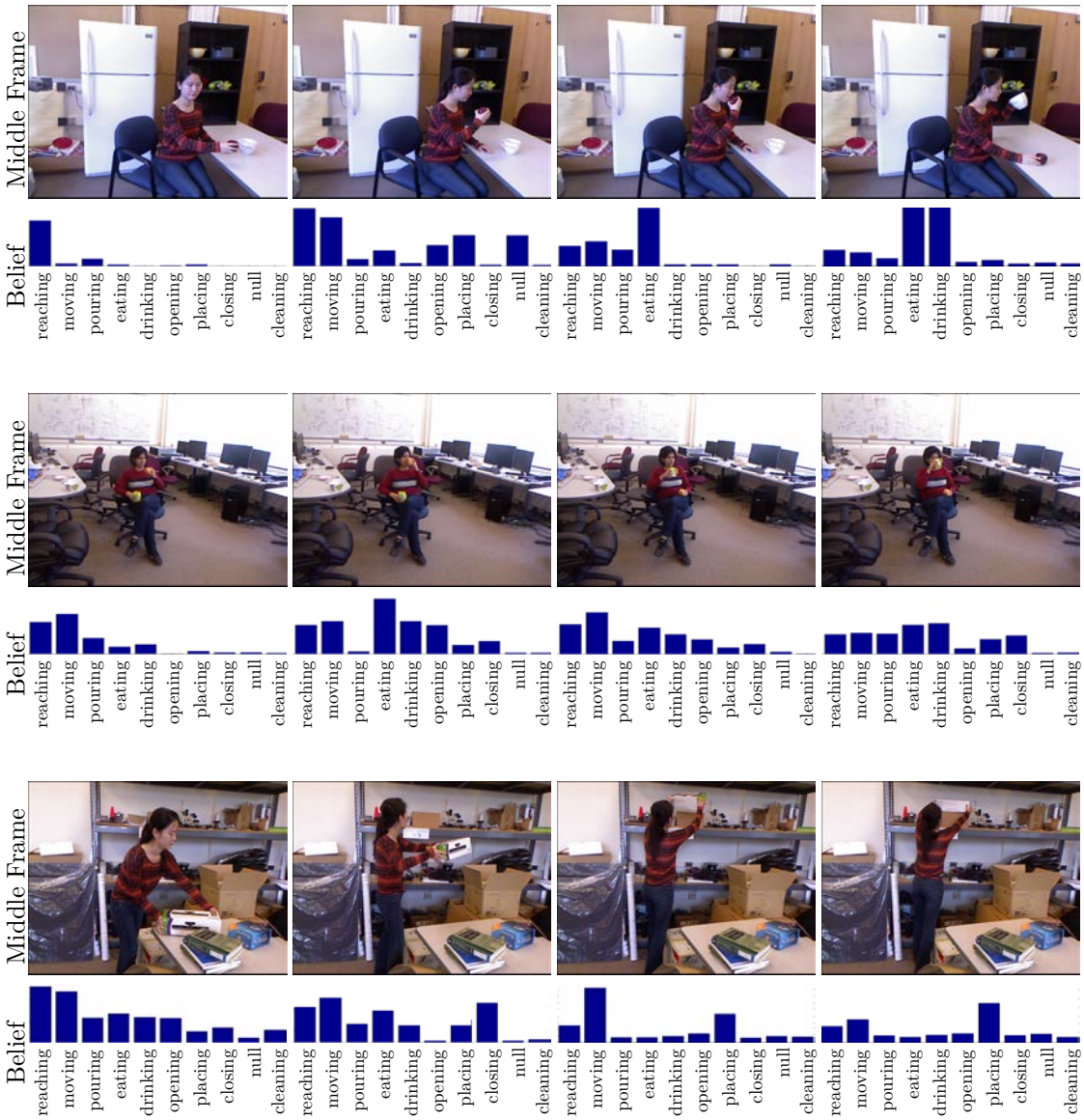


Figure 3.5: **Anticipated belief over activity.** In the odd numbered rows, we show a middle frame of the temporal segment. In the even numbered rows, we show the anticipated belief. Note that frames are not visible to the algorithm and only included for evaluation.

Data: We use CAD-120 [102] dataset in order to evaluate our method. CAD-120 dataset includes 120 RGB-D videos of four different subject performing activities *reaching, moving, pouring, etc.* while interacting with objects having affordances *reachable, movable, pourable, etc.*. There are 10 activity classes and 12 object affordance classes.

Experimental Setup: For computing the features and learning the CRF parameters, we follow the approach and the code in [102]. Following the convention in [102], we use 4-fold cross-validation by training over the data from 3 subjects and testing on the remaining subject. We then average the results over 4-folds. We implemented the rCRF as we explain in Algorithm 1 with the following parameters obtained via cross-validation; we sampled $M = 15$ diverse samples and ran the recursive message updates with the number of iterations limit as 5.

For the anticipation setting, In order to experiment the τ seconds into the future anticipation, we experiment over all feasible anticipation scenarios. In other words, we anticipated the time instant $t+\tau$ by using the segments $1 \dots t$ for all $t < T-\tau$, where T is the length of the video. Then, we averaged the score over all feasible experiments.

Baseline Algorithms: In detection setting, we compare the detection results of the rCRF to MAP solution of the spatiotemporal CRF in [102]. We also included the state-of-the art activity detection results from Hu et al. [75]. Moreover, [75] is not based on object affordances and it only outputs activity detections. For the anticipation, we compare the rCRF with the state-of-the-art anticipation methods ATCRF [100] and GP-LCRF[86]. We also include DCRF[198]. In order to evaluate the contribution of the recursive modeling and the structured diversity separately, we also compare the rCRF with a recursive approach without diversity and a diversity-based approach without recursive modeling baselines.

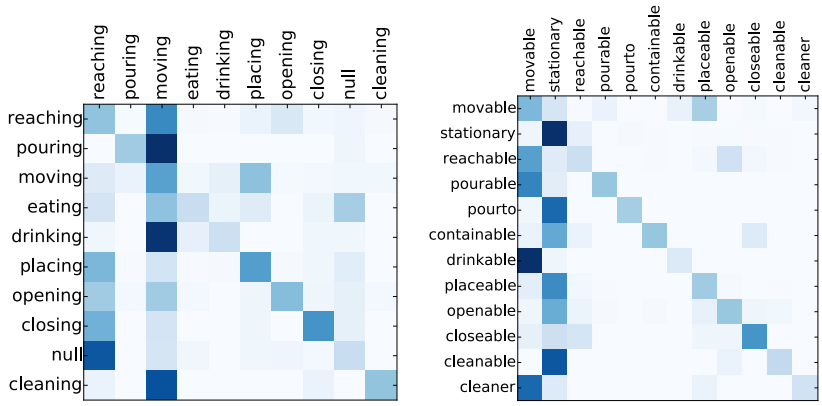
The DivMBest algorithm in [12] uses the diverse sampling method to sample CRFs defined over each frame separately. DivMBest[12] then finds the most likely sequence via Viterbi algorithm. Since it is missing the recursive modeling, it serves as *structured diversity without recursive filtering* baseline. We replace the

diversity-based sampling in our method with Gibbs sampler and consider it as *recursive filtering approach without structured diversity* baseline. For the Gibbs sampling, we sampled 50 samples per temporal segment. We denote the recursive approach with Gibbs sampling as *"rCRF w/o div"* while tabulating the results.

Evaluation Metrics: For activity detection, we compute the ratio of the correctly classified labels (*micro precision*) and the averages of the precision and recall values computed for each activity and object affordance classes (*macro precision* and *macro recall*). For anticipation, we record the ratio of the correctly classified labels *micro precision*, the average of the f-1 score that is computed for each activity and object affordance class (*macro f-1 score*), and the precision of the top 3 anticipated labels (*robot anticipation metric*). While computing the *robot anticipation metric*; if any of the top 3 anticipation is correct, it is counted as true positive.

Accuracy of the rCRF in detection setting. We evaluate the rCRF for activity detection and summarize the results in Table 3.1. Table 3.1 suggests that the rCRF outperforms the MAP solution [102] and performs similarly with the state-of-the-art solution [75]. Since rCRF and [102] are using the same spatial relations, the performance difference is due to the modeling of the temporal relations in rCRF. We use first-order statistics as temporal dynamics, and they are quite accurate as shown in the heat map in Figure 3.6. They also capture semantic information like objects become stationary after being used.

Accuracy of the rCRF in anticipation setting. We evaluate the accuracy of the belief we compute via rCRF, both quantitatively and qualitatively. For qualitative evaluation, we show the segment that we are anticipating the belief over, as well as the belief we obtain in Figure 3.5. Please note that, this visual information is not visible to the algorithm, and it is only included for the subjective



(a) Human Activity (b) Object Affordance

Figure 3.6: Heat map of the first-order statistics of activity and object affordance classes. They are used as temporal dynamics by rCRF.

evaluation.

As shown in the figure, anticipated belief is capturing the scene accurately. Belief is accurate even for the case of concurrent activities. For example, in the second column of the second row in Figure 3.5, subject is reaching the microwave and moving the cleaner. Our method assigns similar likelihood values to both reaching and moving.

We also perform quantitative analysis over anticipation accuracy. We anticipate 3 seconds into the future and summarize the results in Table 3.2. As shown in the Table 3.2, rCRF outperforms the state-of-the-art heuristic method [100] and the GP-LCRF method [86] significantly as well as all other baselines. We believe this result is due to the accurate joint-modeling of the temporal relations and the CRF model. We further analyzed this behavior in the subsequent sections.

How important is the recursive modeling? DivMBest[12] is the application of the structured diversity without recursive modeling of the Bayesian filtering. In all experiments (Table 3.1 and 3.2), rCRF outperforms the DivMBest [12]. We believe this is because rCRF samples $p(y^t|x^1,\ldots,x^T)$ instead of $p(y^t|x^t)$ as in the

Table 3.1: **Detection Performance over CAD-120.** We compare rCRF with MAP solution and baselines for detections accuracy.

	Sub-activity			Object Affordance		
	micro	macro		micro	macro	
	prec(%)	prec(%)	rec(%)	prec(%)	prec(%)	rec(%)
Chance	10.0±0.1	10.0±0.1	10.0±0.1	8.3±0.1	8.3±0.1	8.3±0.1
Hu et al.[75]	67.8±1.4	65.5±3.5	**63.5±6.6**	N/A	N/A	N/A
MAP Sol[102]	63.4±1.6	65.3±2.3	54.0±4.6	79.4±0.8	62.5±5.4	50.2±4.9
DivMBest[12]	64.0±1.3	61.7±2.1	56.4±2.7	80.1±1.0	76.2±2.5	53.2±3.2
DCRF[198]	61.2±2.1	62.8±2.8	54.3±1.5	71.9±2.9	80.6±2.4	62.5±3.6
rCRF w/o div	61.2±1.8	64.0±1.8	52.7±3.8	75.2±2.4	79.3±3.1	63.7±2.9
rCRF	**68.1±1.3**	**66.1±2.7**	57.2±3.9	**81.5±1.1**	**85.2±2.4**	**71.6±3.9**

Table 3.2: **Anticipation performance for the anticipating 3 seconds in the future.** We compare rCRF with state-of-the-art anticipation algorithm and baselines for anticipation accuracy.

	Sub-activity			Object Affordance		
	micro	macro	robot ant.	micro	macro	robot ant.
Method	prec(%)	f1-scr(%)	metric(%)	prec(%)	f1-scr(%)	metric(%)
Chance	10.0±0.1	10.0±0.1	30.0±0.1	8.3±0.1	8.3±0.1	24.9±0.1
GP-LCRF [86]	52.1±1.2	43.2±1.5	76.1±1.5	68.1±1.0	44.2±1.2	74.9±1.1
ATCRF [100]	47.7±1.6	37.9±2.6	69.2±2.1	66.1±1.9	36.7±2.3	71.3±1.7
DivMBest[12]	47.9±1.4	43.2±3.6	71.5±2.7	61.3±1.4	56.3±2.1	73.3±0.5
DCRF[198]	48.3±2.6	35.4±1.8	66.6±1.1	55.2±3.1	48.5±3.1	71.24±2.2
rCRF w/o div	49.6±2.1	39.7±2.6	65.1±1.1	56.2±1.9	47.4±3.1	70.8±2.5
rCRF	**54.3±3.9**	**45.8±2.7**	**76.5±2.6**	**78.7±3.4**	**74.9±3.8**	**82.1±2.9**

case of [12]. In other words, DivMBest [12] samples without considering temporal relations; on the contrary, we sample the full belief directly.

Moreover, the improvement over the DCRF model shows the important of accurate recursive modeling. DCRF uses the recursive modeling without the proposed conversion of the discriminative likelihood into generative one and it performs

poorly. Hence, the proposed conversion is a necessary step.

We also studied the effect of anticipation horizon. We computed precision of all methods for horizons between 1 and 10 seconds and plotted in Figure 3.8 and 3.9. We see significant improvements over longer anticipation time horizons.

In Figure 3.8 and 3.9, accuracy of all algorithms decreases with the increasing horizon. One interesting observation is decrease rate of DivMBest is steeper than others. Since DivMBest misses the recursive nature of the problem, accuracy of the belief it computes is limited; hence, the resulting belief does not stay informative with increasing horizon.

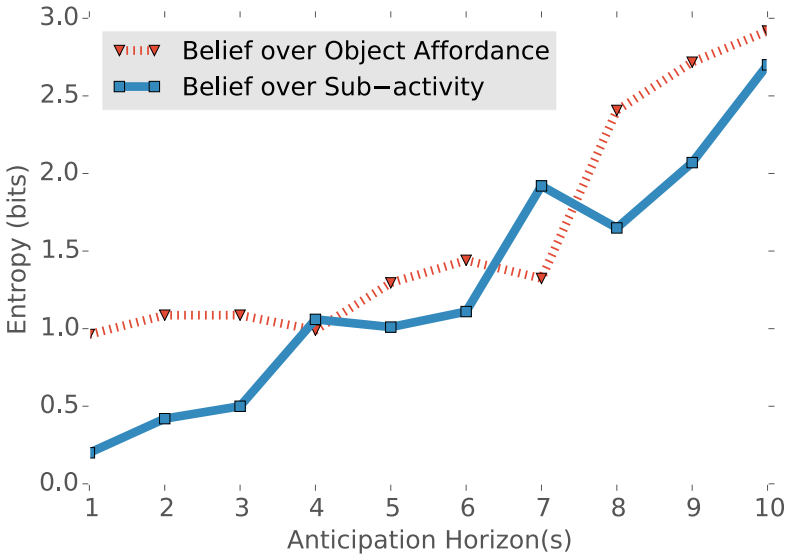


Figure 3.7: Entropy of the belief vs. time *(uniform dist. has* $\approx$ 3.32 *bit entropy)*

We further computed the entropy of the belief rCRF computes and plotted its average in Figure 3.7. The decrease rate of the accuracy is much smaller than the increase rate of the entropy. In summary, recursive modeling is necessary for an accurate belief estimation and rCRF computes flatter yet still informative beliefs

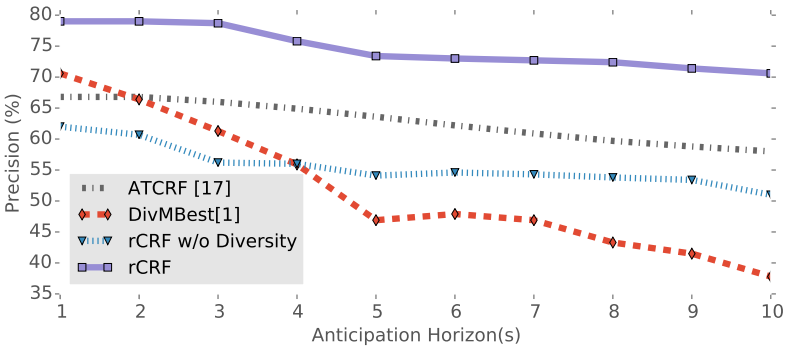


Figure 3.8: Precision vs. anticipation horizon for object affordance.

with increasing horizon.

How to efficiently cover the output space? In order to see the effect of structural diversity on covering the output space, we compare the rCRF with a version of it in which we replace diverse sampling with the Gibbs sampler. As expected, Gibbs sampler only sampled the small region around the posterior and failed to cover the output space. Within all experiments, rCRF outperforms Gibbs sampler baseline. Another interesting observation is, as shown in Figure 3.8&3.9, although Gibbs sampler based method performed slightly better than other baselines for short horizon activity anticipation, it performed much worse for object affordance. We believe this is because of the dimensionality. Activity space has dimension 10^T whereas the object affordance space has dimension $12^{T \cdot M}$ where T is the length of the video and M is the number of objects. Hence, diversity plays bigger role with increasing dimension. Moreover, [100] uses the domain knowledge by selectively sampling points around the hand, etc. and it performs better than both baselines with increasing horizon. We believe this result is due to the efficient coverage of the output space with heuristics.

Computationally-efficient inference: We evaluated the computational efficiency by computing the average computation time for anticipating 3 second in the future via rCRF and the fastest available anticipation algorithm (the

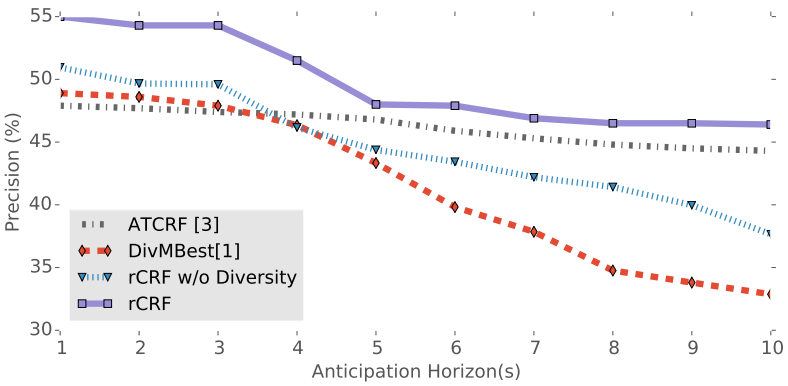


Figure 3.9: Precision vs. anticipation horizon for subject activities.

ATCRF[100]). Within our experiments, we did not include any pre-processing or feature extraction computation (they are same for all algorithms). Our experiments suggest that the rCRF is faster than [100] as shown in Table 3.3. Hence, rCRF model outperforms the state-of-the-art anticipation algorithm in terms of speed in addition to the accuracy.

Table 3.3: Computation time for anticipating 3 seconds in the future excluding pre-processing (*see supplementary material for details*).

ATCRF [100]	34.1s	rCRF	1.41s

Can rCRF generalize to RGB data?: Since there is no RGB activity dataset with object labels, it is hard to compare our algorithm in the RGB activity analysis setting. Removing the concept of the objet form the graph, makes it a chain-CRF and the inference and learning becomes straightforward. However, we still implement our rCRF over a linear-chain CRF for RGB activity analysis. We based our implementation on MPII cooking activity dataset [159] and use the publicly distributed features from the authors webpage. The shared features are HOG, HOF, dense trajetory features and MBH [30].

As shown in the Table 3.4, our method outperforms all baselines and competing

Table 3.4: Anticipation performance for the anticipating 3 seconds in the future in MPII Cooking Dataset[159].

Method	micro prec(%)	macro prec(%)	macro recall(%)
Chance	1.5±0.6	1.5±0.6	1.5±0.6
ATCRF [100]	33.4±3.3	52.1±4.6	12.1±1.4
DivMBest[12]	34.4±2.8	55.3±5.0	14.3±1.2
rCRF	**37.4±2.9**	**63.2±5.5**	**26.1±2.6**

algorithms. We did not include Gibbs sampling here since the dimension of the activity space is rather low and the experiment over diversity is not informative. We believe this result is due to the accurate handling of temporal information in rCRF and it shows that it generalizes to other modalities.

3.5 Conclusions

In this work, we consider the problem of using rich CRF-based scene models in Bayesian filtering setting. We presented the rCRF model, which uses rich modeling power of CRFs in recursive Bayesian filtering. We further developed a computationally-tractable method based on Jensen inequality and structured diversity. We performed extensive experiments that show rCRF accurately anticipates the future beliefs over CRFs. We also experimentally demonstrated that the recursive framework significantly improves the accuracy of anticipation. Our rCRF not only resulted in more accurate anticipation but also improved the computation time.

CHAPTER 4

LEARNING AT SCALE - LEARNING ACTIONABLE REPRESENTATIONS FROM VIDEOS WITH NO SUPERVISION

"And what is the use of a book," thought Alice, "without pictures or conversation?"

Lewis Carroll

Alice's Adventures in Wonderland

In the last decade, we have seen a dramatic democratization in the way we access and generate information. One of the major shifts was moving from an expert curated information sources into crowd-generated large scale knowledge bases like Wikipedia. For example, the way we generate and access *cooking recipes* has been transformed significantly. Google Trends[62] indicates that in the year of 2005 number of Google searches for *cookbooks* were 1.56 times larger than the number of searches for *cooking videos*. In the year of 2016, the number of searches for *cooking videos* is 8.6 times larger than that of *cookbooks*. This behavior is mostly due to the large volume of cooking videos available on the Internet. In an era an average user getting 2 million videos for the query *How to make a pancake?*, we need computer vision algorithms which can understand such information and represent it to users in a compact form. Such an algorithm is not only useful for humans to digest millions of videos but also useful for robots to learn concepts from online video collections by themselves.

The feasibility of an algorithm, which can understand a large-scale video collection is mostly motivated by the structure of the videos we generate. Human

communication takes many forms, including language and videos. For instance, explaining "how-to" perform a certain task can be communicated via language (*e.g.* , Do-It-Yourself books) as well as visual (e.g., instructional YouTube videos) information. Regardless of the form, such human-generated communication is generally structured and has a clear beginning, end, and a set of steps in between. Finding this hidden and objective steps of human communication is a critical step to understand large video collections.

Language and vision provide different, but correlating and complementary information. Challenge lies in that both video frames and language (from subtitles generated via Automatic Speech Recognition) are only a noisy, partial observation of the actions being performed. However, the complementary nature of language and vision gives the opportunity to understand the activities only from these partial observations. In this chapter, we present a unified model, considering both of the modalities, in order to parse human activities into activity steps with no form of supervision other than requiring videos to be the same category (*e.g.* , all cooking eggs, changing tires, etc.).

The key idea in our approach is the observation that the large collection of videos, pertaining to the same activity class, typically include only a few objective activity steps, and the variability is the result of exponentially many ways of generating videos from activity steps through subset selection and time ordering. We study this construction based on the large-scale information available in YouTube in the form of instructional videos (*e.g.* , "Making pancake", "How to tie a bow tie"). Instructional videos have many desirable properties like the volume of the information and a well defined notion of activity step. However, the proposed parsing method is applicable to any type of videos as long as they are composed

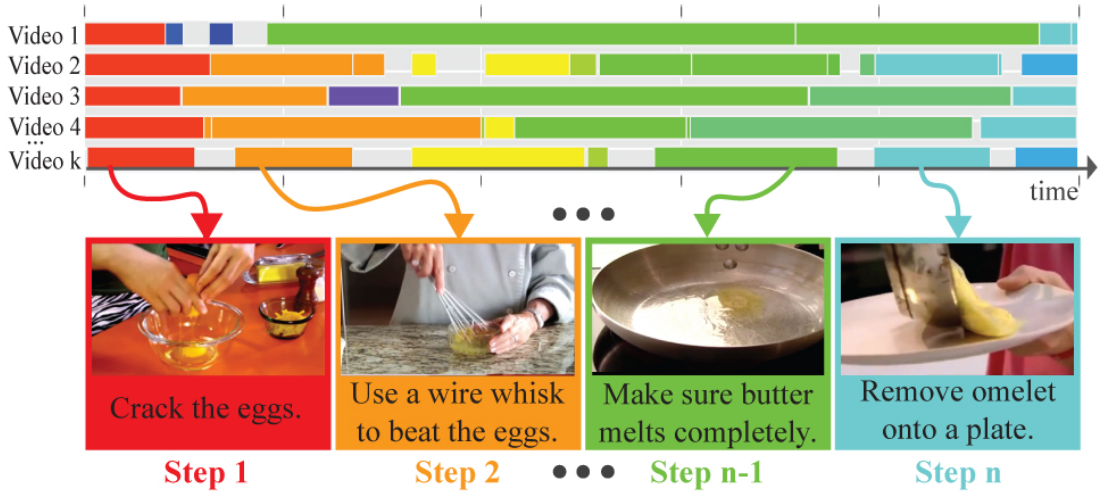


Figure 4.1: Given a large video collection (frames and subtitles) of an instructional category (*e.g.* , How to cook an ommelette?), we discover activity steps (*e.g.* , crack the eggs). We also parse the videos based on the discovered steps.

of a set of steps.

The output of our method can be seen as the semantic "storyline" of a rather long and complex video collection (see Fig. 4.1). This storyline provides what particular steps are taking place in the video collection, when they are occurring, and what their meaning is (*what-when-how*). This method also puts videos performing the same overall task in common ground and capture their high-level relations.

In the proposed approach, given a collection of videos, we first generate a set of language and visual atoms. These atoms are the result of relating object proposals from each frame as well as detecting the frequent words from subtitles. We then employ a generative *beta process mixture model*, which identifies the activity steps shared among the videos of the same category based on a representation using learned atoms. Although we do not explicitly enforce these steps to be semantically meaningful, our results highly correlate with the semantic steps. In our method, we do neither use any spatial or temporal label on actions/steps nor any labels on

object categories. We later learn a Markov language model to provide a textual description of the activity steps based on the language atoms it frequently uses.

We evaluate our approach on various settings. First of all, we collected a large-scale dataset of instructional videos from YouTube following the most frequently performed *how to* queries. Then, we evaluate temporal parsing quality per video and also a semantic clustering per category (how-to query). Second of all, we extensively analyze the contribution of each modality as well as the robustness against the language noise. Robustness against the language noise is a critical one since ASR always expected to have some errors. Moreover, results suggest that both language and vision is critical for semantic parsing. Finally, we discuss and present a novel robotics application. We start with a single query and generate a detailed physical plan to perform the task. We present compelling simulation results suggesting that our algorithm has a great potential for robotics applications.

4.1 Related Work on Parsing Video Collections

Designing an artificial intelligence agent, which can understand human generated videos have been topic of computer vision and robotics researchers for decades. Motivated by the application of surveillance, video summarization was one of the earliest methods that are related to our problem. The surveillance applications further motivated the activity and event recognition methods. With the help of the availability of larger datasets, researchers managed to train machine learning models, which can detect certain events. Recently, the datasets have gotten larger and cross-modal enabling algorithms, which can link vision with language. In the mean time, the focus of robotics community was on parsing recipes directly for

manipulation. We list and discuss related works from each field in the following sections.

4.1.1 Video Summarization:

Summarizing an input video as a sequence of key frames (static) or video clips (dynamic) is useful for both multimedia search interfaces and retrieval purposes. Early works in the area are summarized in [193] and mostly focus on *choosing keyframes*.

Summarizing videos is particularly important for some specific domains like ego-centric videos and news reports as they are generally long in duration. There are many successful works [119, 126, 160]; however, they mostly rely on characteristics specific to the domain.

Summarization is also applied to the large image collections by recovering the temporal ordering and visual similarity of images [93], and by Gupta et al. [67] to videos in a supervised framework using action annotations. These collections are also used for key-frame selection [91] and further extended to video clip selection [92, 149]. Unlike all of these methods, which focus on forming a set of key frames/clips for a compact summary (which is not necessarily semantically meaningful), we provide a fresh approach to video summarization by performing it through semantic parsing on vision and language. However, regardless of this dissimilarity, we experimentally compare our method against them.

4.1.2 Modeling Visual and Language Information:

Learning the relationship between the visual and language data is a crucial problem due to its immense applications. Early methods [11] in this area focus on learning a common multi-modal space in order to jointly represent language and vision. They are further extended to learning higher level relations between object segments and words [178]. Similarly, Zitnick et al.[214, 213] used abstracted clip-arts to understand spatial relations of objects and their language correspondences. Kong et al. [98] and Fidler et al. [51] both accomplished the task of learning spatial reasoning by only using the image captions. Relations extracted from image-caption pairs, are further used to help semantic parsing [206] and activity recognition [138]. Recent works also focus on automatic generation of image captions with underlying ideas ranging from finding similar images and transferring their captions [146] to learning language models conditioned on the image features [94, 179, 46]; their employed approach to learning language models is typically either based on graphical models [46] or neural networks [179, 94, 87].

All aforementioned methods are using supervised labels either as strong image-word pairs or weak image-caption pairs, while our method is fully unsupervised.

4.1.3 Activity/Event Recognition:

The literature of activity recognition is broad. The closest techniques to ours are either supervised or focus on detecting a particular (and often short) action in a weakly/unsupervised manner. Also, a large body of action recognition methods are intended for trimmed videos clips or remain limited to detecting very short actions [108, 180, 141, 113, 44, 162]. Even though some recent works attempted

action recognition in untrimmed videos [82, 145, 80], they are mostly fully supervised.

Additionally, several methods for localizing instances of actions in rather longer video sequences have been developed [41, 73, 114, 17, 148]. Our work is different from those in terms of being multimodal, unsupervised, applicable to a video collection, and not limited to identifying predefined actions or the ones with short temporal spans. Also, the previous works on finding action primitives such as [141, 205, 79, 112, 110] are primarily limited to discovering atomic sub-actions, and therefore, fail to identify complex and high-level parts of a long video.

Recently, event recounting has attracted much interest and intends to identify the evidential segments for which a video belongs to a certain class [184, 31, 10]. Event recounting is a relatively new topic and the existing methods mostly employ a supervised approach. Also, their end goal is to identify what parts of a video are highly related to an event, and not parsing the video into semantic steps.

4.1.4 Recipe Understanding:

Following the interest in community generated recipes in the web, there have been many attempts to automatically process recipes. Recent methods on natural language processing [130, 189] focus on semantic parsing of language recipes in order to extract actions and the objects in the form of predicates. Tenorth et al.[189] further process the predicates in order to form a complete logic plan. The aforementioned approaches focus only on the language modality and they are not applicable to the videos. The recent advances [15, 20] in robotics use the parsed recipe in order to perform cooking tasks. They use supervised object detectors and report a

successful autonomous experiment. In addition to the language based approaches, Malmaud et al.[129] consider both language and vision modalities and propose a method to align an input video to a recipe. However, it can not extract the steps automatically and requires a ground truth recipe to align. On the contrary, our method uses both visual and language modalities and extracts the actions while autonomously discovering the steps. There is also an approach, which generates multi-modal recipes from expert demonstrations [63]. However, it is developed only for the domain of "teaching user interfaces" and are not applicable to videos.

In summary, three aspects differentiate this work from the majority of existing techniques: 1) discovering semantic steps from a video category, 2) being unsupervised, 3) adopting a multi-modal joint vision-language model for video parsing.

4.2 Overview of Our Unsupervised Approach to Learning Actionable Representations

Our algorithm takes a *how-to* sentence as an input query which we further use to download a large-collection of videos. We then learn a multi-modal dictionary using a novel hierarchical clustering approach. We finally use the learned dictionary in order to discover and localize activity steps. We visualize this process in Figure 4.2 with a toy example. The output of our algorithm is temporal parsing of each video as well as an id for each semantic activity step. In other words, we not only temporally segment each video, we also relate the occurrence of same activity over multiple videos with each other. We further visualize the output in Figure 4.1.

Atoms: Given a large video-collection composed of visual information as well as

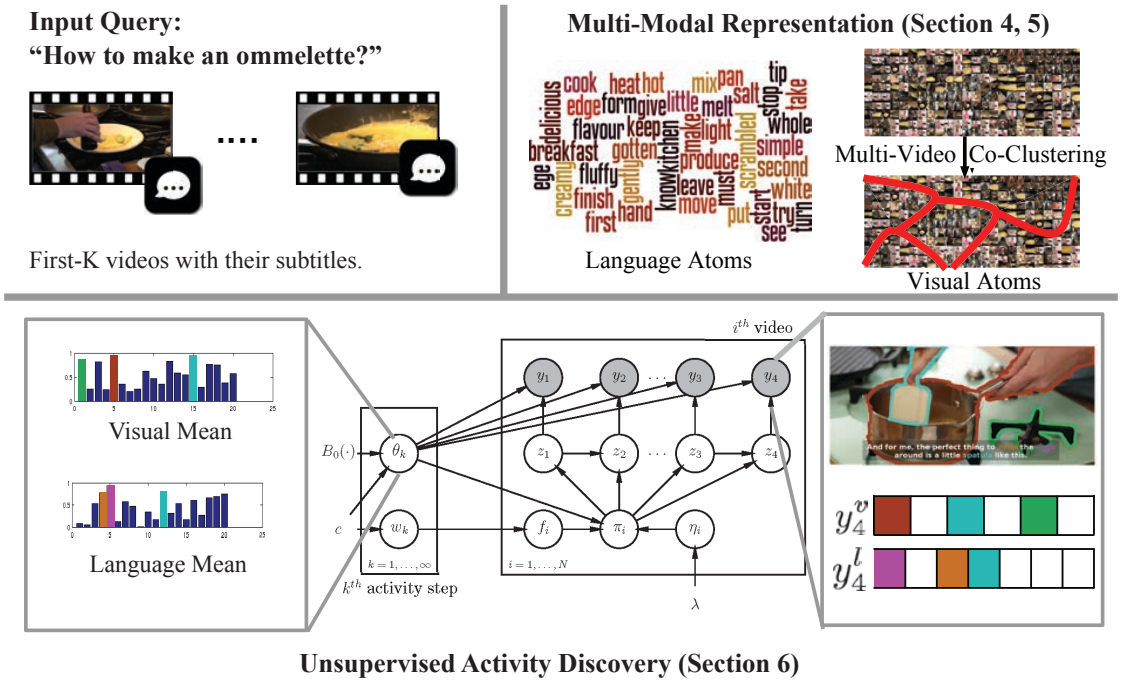


Figure 4.2: **Summary of our method.** We start with a single natural language query like *how to make an omelette* and then we crawl the top K videos returned by this query from YouTube. We learn a multi-modal dictionary composed of salient words and object proposals. Rest of the algorithm represents frames and activities in terms of the learned dictionary. For example, in the bottom figure, colors represent such atoms and both activity descriptions Θ and frame representations y_t are defined in terms of these atoms. (see Fig 4.1 for output)

subtitles, our algorithm starts with learning a set of visual and language *atoms* which are further used for representing multimodal information (Section 4.3). These atoms are designed to be likely to correspond to the mid-level semantic concepts like actions and objects. In order to learn language *atoms*, we find frequently occurring salient words among the subtitles using tf-idf like approach. Learning visual atoms is slightly trickier due to the intra-cluster variability of visual concepts. We generate object proposals and jointly-cluster them into mid-level atoms to obtain visual atoms. We develop a hierarchical clustering algorithm for this purpose (Section 4.4).

Discovering Activities: After learning the atoms, we represent the multi-modal information in each frame based on the occurrence statistics of the atoms. Given the multi-modal representation of each frame, we discover set of temporal clusters occurring over multiple videos using a non-parametric Bayesian method (Section 4.5). We expect these clusters to correspond to the activity steps, which construct the high level activities. Our empirical results confirm this as the resulting clusters significantly correlates with the semantic activity steps.

4.3 Multi-Modal Representation with Atoms

Finding the set of activity steps over large collection of videos having large visual varieties requires us to represent the semantic information in addition to the low-level visual cues. Hence, we find our language and visual atoms by using mid-level cues like object proposals and frequent words.

Learning Visual Atoms: In order to learn visual atoms, we create a large collection of object proposals by independently generating object proposals from each frame of each video. These proposals are generated using the Constrained Parametric Min-Cut (CPMC) [26] algorithm based on both appearance and motion cues. We note the k^{th} proposal of t^{th} frame of i^{th} video as $r_t^{(i),k}$. Moreover, we drop the video index (i) if it is clearly implied in the context.

In order to group these object proposals into mid-level visual atoms, we follow a clustering approach. Although any graph clustering approach (eg. Keysegments [120]) can be applied for this, the joint processing of a large video collection requires handling large visual variability among multiple videos. We propose a new method to jointly cluster object proposals over multiple videos in Section 4.4. Each cluster

of object proposals corresponds to a visual atom.

Learning Language Atoms: We define the language atoms as the salient words which occur more often than their ordinary rates based on the *tf-idf* measure. The *document* is defined as the concatenation of all subtitles of all frames of all videos in the collection as $D = \bigcup_{i \in N_C} \bigcup_{t \in T^{(i)}} L_t^i$. Then, we follow the classical tf-idf measure and use it as $tfidf(w, D) = f_{w,D} \times \log\left(1 + \frac{N}{n_w}\right)$ where w is the word we are computing the tf-idf score for, $f_{w,D}$ is the frequency of the word in the *document* D, N is the total number of video collections we are processing, and n_w is the number of video collections whose subtitle include the word w.

We sort words with their "tf-idf" values and choose the top K words as language atoms ($K = 100$ *in our experiments*). As an example, we show the language atoms learned for the category *making scrambled egg* in Figure 4.2

Representing Frames with Atoms: After learning the visual and language atoms, we represent each frame via the occurrence of atoms (binary histogram). Formally, the representation of the t^{th} frame of the i^{th} video is denoted as $\mathbf{y_t^{(i)}}$ and computed as $\mathbf{y_t^{(i)}} = [\mathbf{y_t^{(i),l}}, \mathbf{y_t^{(i),v}}]$ such that k^{th} entry of the $\mathbf{y_t^{(i),l}}$ is 1 if the subtitle of the frame has the k^{th} language atom and 0 otherwise. $\mathbf{y_t^{(i),v}}$ is also a binary vector similarly defined over visual atoms. We visualize the representation of a sample frame in the Figure 4.3.

4.4 Joint Clustering over Video Collection

Given a set of object proposals generated from "multiple videos", simply combining them into a single collection and clustering them into atoms is not desirable for

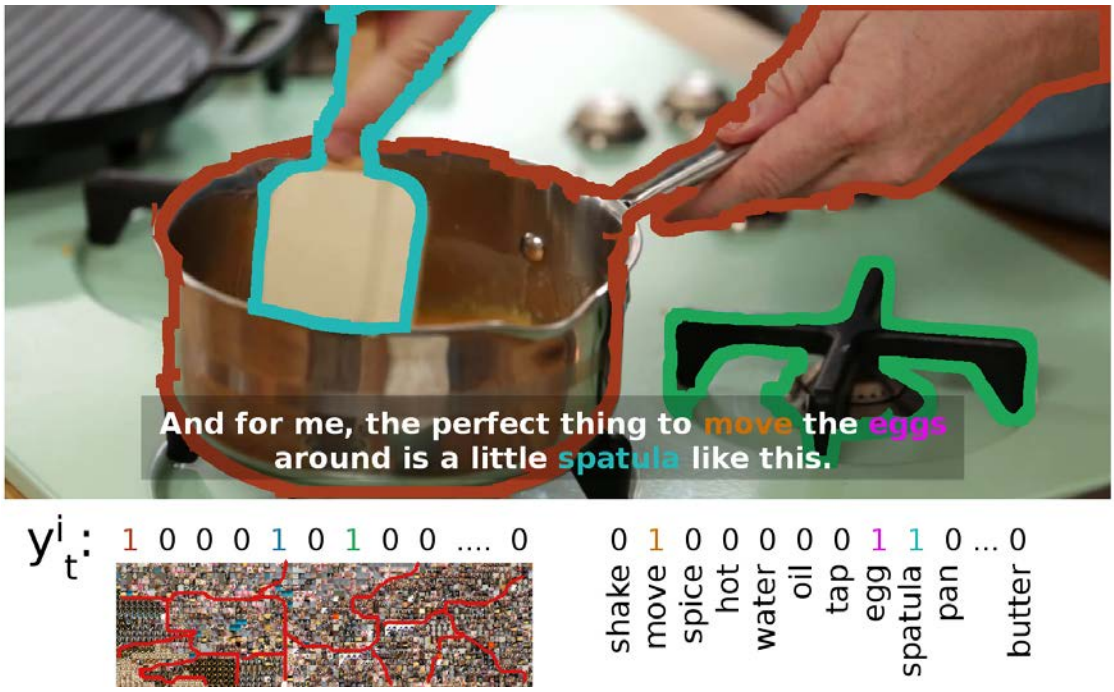


Figure 4.3: **Representation for a sample frame.** Three of the object proposals of sample frame are in the visual atoms and three of the words are in the language atoms.

two reasons: (1) semantic concepts have large visual differences among different videos and accurately clustering them into a single atom is hard, (2) atoms should contain object proposals from multiple videos in order to semantically relate the videos. In order to satisfy these requirements, we propose a joint extension to spectral clustering. Note that the purpose of this clustering is generating atoms where each clusters represents an atom.

Basic Graph Clustering: Consider the set of object proposals extracted from a single video $\{r_t^k\}$, and a pairwise similarity metric $d(\cdot,\cdot)$ for them. We follow the single cluster graph partitioning (SCGP)[144] approach to find the dominant cluster, which maximizes the intra-cluster similarity:

$$\arg\max_{x_t^k} \frac{\sum_{(k_1,t_1),(k_2,t_2)\in K\times T} x_{t_1}^{k_1} x_{t_2}^{k_2} d(r_{t_1}^{k_1}, r_{t_2}^{k_2})}{\sum_{(k,t)\in K\times T} x_t^k} \tag{4.1}$$

where x_t^k is a binary variable which is 1 if r_t^k is included in the cluster, T is

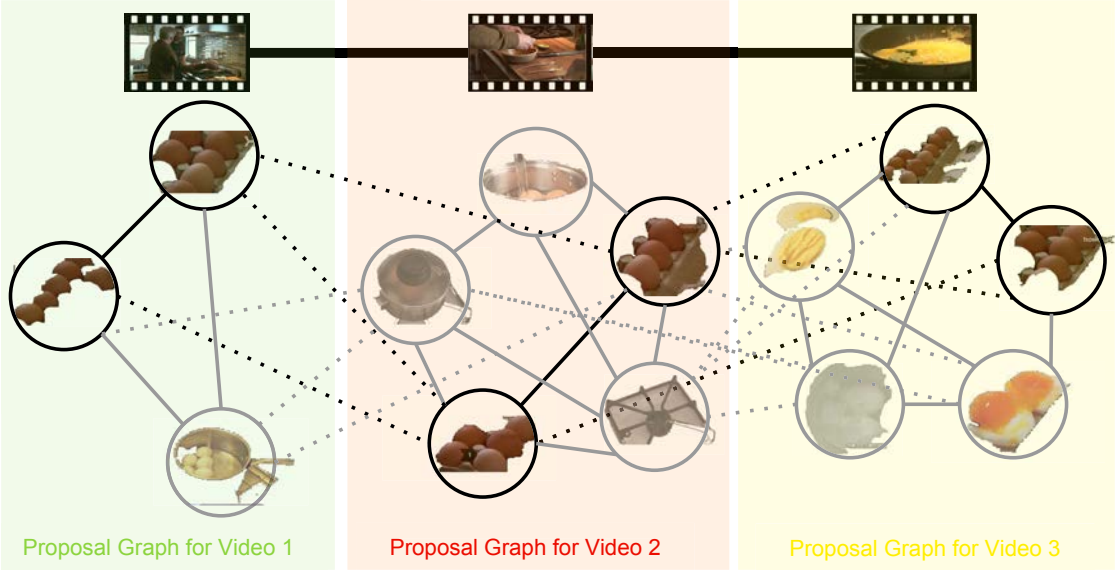


Figure 4.4: **Joint proposal clustering.** Here, we show the $1^{st}NN$ video graph and $2^{nd}NN$ region graph. Each object proposal is linked to its two NNs from the video it belongs and two NNs from the videos it is neighbour of. Black nodes are the proposals selected as part of the cluster and the gray ones are not selected. Moreover, dashed lines are intra-video edges and solid ones are inter-video edges.

the number of frames and K is the number of clusters per frame. Adopting the vector form of the indicator variables as $\mathbf{x_{tK+k}} = x_t^k$ and the pairwise distance matrix as $\mathbf{A}_{t_1K+k_1,t_2K+k_2} = d(r_{t_1}^{k_1}, r_{t_2}^{k_2})$, equation (4.1) can be compactly written as $\arg\max_{\mathbf{x}} \frac{\mathbf{x^T}A\mathbf{x}}{\mathbf{x^T}\mathbf{x}}$ This can be solved by finding the dominant eigenvector of $\mathbf{x}$ after relaxing x_t^k to $[0, 1]$ [144, 147]. Upon finding the cluster, the members of the selected cluster are removed from the collection and the same algorithm is applied to find remaining clusters.

Joint Clustering: Our extension of the SCGP into multiple videos is based on the assumption that the key objects occur in most of the videos. Hence, we reformulate the problem by enforcing the homogeneity of the cluster over all videos.

We first create a kNN graph of the videos based on the distance between their

textual descriptions. We use the χ^2 distance of the bag-of-words computed from the video description. We also create the kNN graph of object proposals in each video based on the pre-trained "fc7" features of AlexNet[105]. This hierarchical graph structure is visualized in Figure 4.4 for 3 videos sample. After creating this graph, we impose both "inter-video" and "intra-video" similarity among the object proposals of each cluster. Main rationale behind this construction is having a separate notion of distance for inter-video and intra-video relations since the visual similarity decreases drastically for inter-video ones.

Given the intra-video distance matrices $\mathbf{A^{(i)}}$, the binary indicator vectors $\mathbf{x^{(i)}}$, and the inter-video distance matrices as $\mathbf{A^{(i,j)}}$, we define our optimization problem as;

$$\arg\max \sum_{i \in N} \frac{\mathbf{x^{(i)^T} A^{(i)} x^{(i)}}}{\mathbf{x^{(i)^T} x^{(i)}}} + \sum_{i \in N} \sum_{j \in \mathcal{N}(i)} \frac{\mathbf{x^{(i)^T} A^{(i,j)} x^{(j)}}}{\mathbf{x^{(i)^T}} \mathbb{1}\mathbb{1}^T \mathbf{x^{(j)}}}, \tag{4.2}$$

where $\mathcal{N}(i)$ is the neighbours of the video i in the kNN graph, $\mathbb{1}$ is vector of ones and N is the number of videos.

Although we can not use the efficient eigen-decomposition approach from [144, 147] as a result of the modification, we can use Stochastic Gradient Descent as the cost function is quasi-convex when relaxed. We use the SGD with the following analytic gradient function:

$$\nabla_{\mathbf{x^{(i)}}} = \frac{2\mathbf{A^{(i)} x^{(i)}} - 2\mathbf{x^{(i)}} r^{(i)}}{\mathbf{x^{(i)^T} x^{(i)}}} + \sum_{i \in N} \frac{\mathbf{A^{i,j} x^j} - \mathbf{x^{(j)^T}} \mathbb{1} r^{(i,j)}}{\mathbf{x^{(i)^T}} \mathbb{1}\mathbb{1}^T \mathbf{x^{(j)}}}, \tag{4.3}$$

where $r^{(i)} = \frac{\mathbf{x^{(i)^T} A^{(i)} x^{(i)}}}{\mathbf{x^{(i)^T} x^{(i)}}}$ and $r^{(i,j)} = \frac{\mathbf{x^{(i)^T} A^{(i,j)} x^{(j)}}}{\mathbf{x^{(i)^T}} \mathbb{1}\mathbb{1}^T \mathbf{x^{(j)}}}$

We iteratively use the method to find clusters, and stop after the $K = 20$ clusters are found as the remaining object proposals were deemed not relevant to the activity. Each cluster corresponds to a visual atom for our application.

In Figure 4.5, we visualize some of the atoms (*i.e.*clusters) we learned for the

query *How to Hard Boil an Egg?*. As apparent in the figure, the resulting atoms are highly correlated and correspond to semantic objects and concepts regardless of their significant intra-class variability.

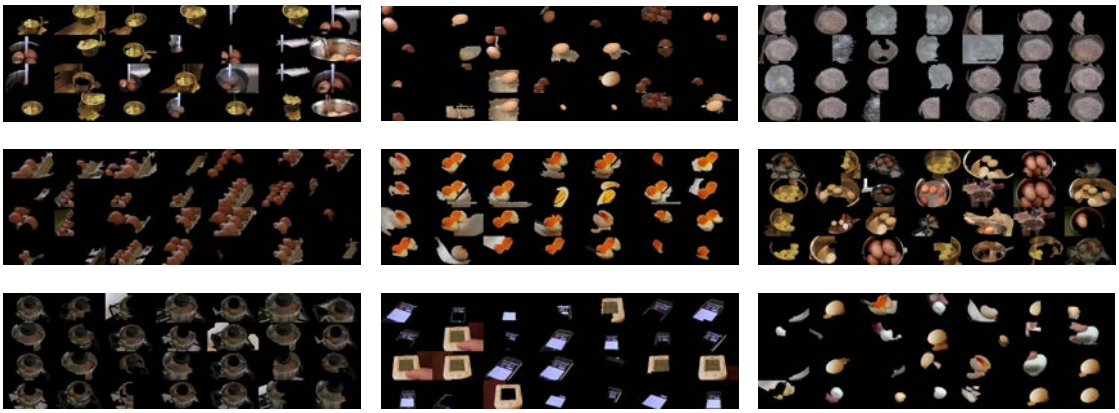


Figure 4.5: **Randomly selected images of four randomly selected clusters learned for *How to hard boil an egg?*** Please note that the objects in the same cluster is not only coming from a single video but discovered over multiple videos. Hence, this stage helps in linking videos with each other. Resulting clusters are semantically accurate since they typically belong to a single semantic concept like water filling the pot.

4.5 Unsupervised Activity Representation

In this section, we explain our model for discovering the activity steps from a video collection given the language and visual atoms. The main idea behind this step is utilizing the repetitive nature of steps. In other words, although there are large number of videos in any chosen category, the underlying set of steps are very few. Hence, we tried to find smallest set of activities, which can generate all the videos we crawl.

We note the extracted representation of the frame t of video i as $\mathbf{y_t^{(i)}}$. We

model our algorithm based on activity steps and note the activity label of the t^{th} frame of the i^{th} video as $z_t^{(i)}$. We do not fix the number of activities and use a non-parametric approach.

In our model, each activity step is represented over the atoms as the likelihood of including them. In other words, each activity step is a Bernoulli distribution over the visual and language atoms as $\theta_k = [\theta_k^l, \theta_k^v]$ such that m^{th} entry of the θ_k^l is the likelihood of observing m^{th} language atom in the frame of an activity k. Similarly, m^{th} entry of the θ_k^v represents the likelihood of seeing m^{th} visual atom. In other words, each frame's representation $\mathbf{y_t^{(i)}}$ is sampled from the distribution corresponding to its activity as $\mathbf{y_t^{(i)}}|z_t^{(i)} = k \sim Ber(\theta_k)$. As a prior over θ, we use its conjugate distribution – *Beta distribution* –.

Given the model above, we explain the generative model which links activity steps and frames in Section 4.5.1.

4.5.1 Beta Process Hidden Markov Model

For the understanding of the time-series information, Fox et al.[53] proposed the Beta Process Hidden Markov Models (BP-HMM). In BP-HMM setting, each time-series exhibits a subset of available features. Similarly, in our setup each video exhibits a subset of activity steps.

Our model follows the construction of Fox et al.[53] and differs in the choice of probability distributions since [53] considers Gaussian observations whereas we adopt binary observations of atoms. In our model, each video i chooses a set of activity steps through an activity step vector $\mathbf{f^{(i)}}$ such that $f_k^{(i)}$ is 1 if i^th video has the activity step k, and 0 otherwise. When the activity step vectors of all videos

are concatenated, it becomes an activity step matrix $\mathbf{F}$ such that i^th row of the $\mathbf{F}$ is the activity step vector $\mathbf{f^{(i)}}$. Moreover, each activity step k also has a prior probability b_k and a distribution parameter θ_k, which is the Bernoulli distribution as we explained in the Section 4.5.

In this setting, the activity step parameters θ_k and b_k follow the *beta process* as;

$$B|B_0, \gamma, \beta \sim \mathrm{BP}(\beta, \gamma B_o), B = \sum_{k=1}^{\infty} b_k \delta_{\theta_k} \tag{4.4}$$

where B_0 and the b_k are determined by the underlying Poisson process [64] and the feature vector is determined as independent Bernoulli draws as $f_k^{(i)} \sim Ber(b_k)$. After marginalizing over the b_k and θ_k, this distribution is shown to be equivalent to Indian Buffet Process (IBP)[64]. In the IBP analogy, each video is a customer and each activity step is a dish in the buffet. The first customer (video) chooses a Poisson(γ) unique dishes (activity steps). The following customer (video) i chooses previously sampled dish (activity step) k with probability $\frac{m_k}{i}$, proportional to the number of customers (m_k) chosen the dish k, and it also chooses Poisson($\frac{\gamma}{i}$) new dishes(activity steps). Here, γ controls the number of selected activities in each video and β promotes the activities getting shared by videos.

The above IBP construction represents the activity step discovery part of our method. In addition, we also need to model the video parsing over discovered steps. Moreover, we need to model these two steps jointly. We model the each video as an Hidden Markov Model (HMM) over the selected activity steps. Each frame has the hidden state –activity step– ($z_t^{(i)}$) and we observe the multi-modal frame representation $\mathbf{y_t^{(i)}}$. Since we model each activity step as a Bernoulli distribution, the emission probabilities follow the Bernoulli distribution as $p(\mathbf{y_t^{(i)}}|z_t^{(i)}) = Ber(\theta_{z_t^{(i)}})$.

For the transition probabilities of the HMM, we do not put any constraint and

simply model it as any point from a probability simplex which can be sampled by drawing a set of Gamma random variables and normalizing them [53]. For each video i, a Gamma random variable is sampled for the transition between activity step j and activity step k if both of the activity steps are included by the video (*i.e.*if f_k^i and f_j^i are both 1). After sampling these random variables, we normalize them to make transition probabilities to sum up 1. This procedure can be represented formally as

$$\eta_{j,k}^{(i)} \sim Gam(\alpha + \kappa\delta_{j,k}, 1), \quad \pi_{\mathbf{j}}^{(\mathbf{i})} = \frac{\eta_{\mathbf{j}}^{(\mathbf{i})} \circ \mathbf{f}^{(\mathbf{i})}}{\sum_k \eta_{j,k}^{(i)} f_k^{(i)}} \tag{4.5}$$

Where κ is the persistence parameter promoting the self state transitions a.k.a. more coherent temporal boundaries, $\circ$ is the element-wise product and π_j^i is the transition probabilities in video i from activity step j to other steps. This model is also presented as a graphical model in Figure 4.6

4.5.2 Gibbs sampling for BP-HMM

We employ Markov Chain Monte Carlo (MCMC) method for learning and inference of the BP-HMM. We follow the exact sampler proposed by Fox et al.[53]. It marginalize over activity likelihoods w and activity assignments $\mathbf{z}$ and samples the rest. MCMC procedure iteratively samples the conditional likelihood of activity matrix $\mathbf{F}$, activity parameters θ and transition weights η. We divide the explanation of this samplers into two sections, sampling the activities through activity matrix ($\mathbf{F}$) and activity parameters (θ), and sampling the HMM parameters η. Marginalization over activity assignments follows the efficient dynamic programming approach.

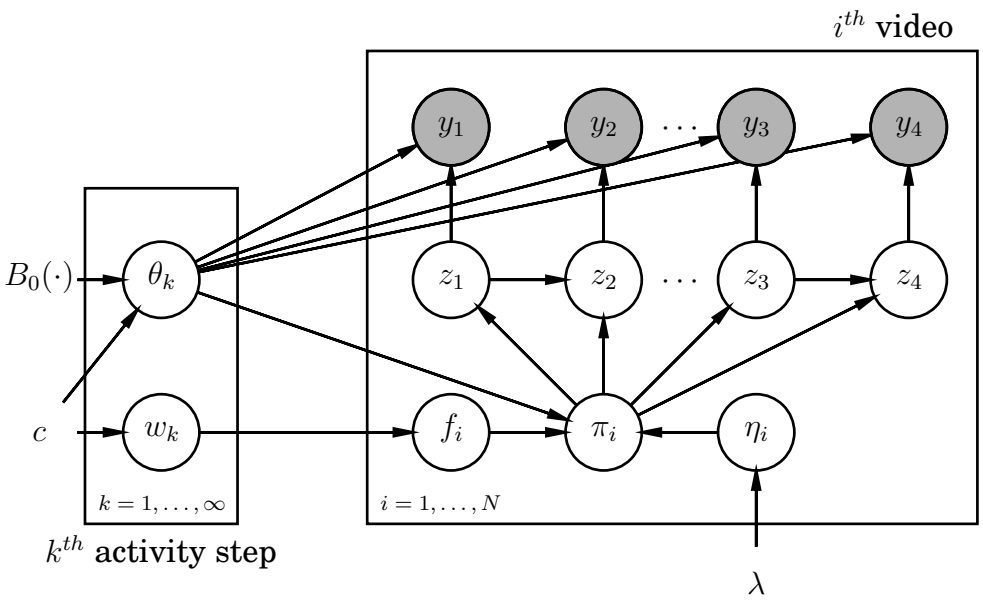


Figure 4.6: **Graphical model for BP-HMM:** The left plate represent the activity steps and the right plate represent the videos. *i.e.*the left plate is for the activity step discovery and right plate is for parsing. *See Section 4.5.1 for details.*

Sampling the Activities: Consider the binary activity inclusion matrix **F** such that $F_{i,j}$ is 1 if the i^{th} video has the j^{th} activity. Following the sampler of Fox et al.[53], we divide the sampling **F** into two parts, namely, sampling the shared activities and sampling the novel activities. Sampling shared activities correspond the re-sampling of existing entries of **F**. We simply iterate over each entry and propose a flip (*i.e.*if the i^{th} video has the j^{th} activity, we propose to flip it and not to include j^{th} activity in the i^{th} video). We accept or reject this proposals following the Metropolis-Hasting rule.

In order to sample the novel activities, we follow the data-driven sampler [77]. Consider the case in which we want to propose a novel activity by setting the $F_{i,j+1}$ to 0. In other words, we introduce a new activity ($j + 1^{th}$ activity) such that i^{th} video includes it. In order to sample the parameters θ_{j+1} of it, we first sample a

temporal window W over the i^{th} video. This window is sampled by sampling the starting frame and the length of the window from a uniform distribution. Then, we sample the novel activity from *Beta distribution* as;

$$\theta_{k,n}|W \sim \text{Beta}(\alpha_n, \beta_n) \tag{4.6}$$

where $\theta_{k,n}$ is the n^{th} entry of θ_k, α_n is the number of frames in the window W which have the atom n, and β_n is the number of frames which do not have the atom n. We use *Beta distribution* because it is the conjugate prior of the *Bernoulli distribution* that we use to model activities.

Sampling the HMM Parameters: When the activities are defined via Θ and each video selects a subset of them via ($\mathbf{F}$), we can compute the likelihood of each state assignment by using the dynamic programming given the transition probabilities η. By using the likelihoods, we sample the state assignments $\mathbf{z}$.

When the states are sampled, we can use the closed-form sampler derived in [53]. Fox et al.[53] shows that the transition probabilities can be sampled through a *Dirichlet* random variable and scaling it with a *Gamma* random variable as;

$$\pi^{(\mathbf{i})} \sim Dir(\ldots, N_{j,k}^{(i)} + \alpha + \delta_{j,k}\kappa, \ldots) \tag{4.7}$$

followed by $\eta^{(\mathbf{i})} = \pi^{(\mathbf{i})} \times C^{(i)}$ such that

$C^{(i)} \sim Gamma(K_+^{(i)}\lambda + \kappa, 1)$. Here, $N_{j,k}^{(i)}$ represents the number of transitions between state j and state k in the video i, α, λ and κ are hyperparameters which we learn with cross-validation, $\delta_{j,k}$ is 1 if $j = k$ and 0 o.w., and $K_+^{(i)}$ is the number of activities the i^{th} video has chosen.

At the end of the Gibbs sampling, our algorithm ends with a set of activities each represented with respect to the discovered atoms *i.e.*$\Theta_1 \ldots \Theta_k$ and label of

each frame among the discovered activities $[1, \ldots, k]$. Θ_i can be considered as a generative distribution of each discovered activity. In other words, if we want to sample a frame from activity i, we simply sample set of language and visual atoms from Θ_i. We perform this sampling in order to generate a language caption for each discovered activity. We also consistently visualize the results of discovery using story lines as shown in Figure 4.1. We assign a color code to each discovered activity and sample keyframes from 4 four different clips of same activity. We further generate a natural language description as well as display the temporal segmentation of each video as a colored timeline.

4.6 Experiments on Large-Scale Video Parsing

In order to experiment the proposed method, we first collected a dataset (details in Section 4.6.1). We labeled small part of the dataset with frame-wise activity step labels and used it as an evaluation corpus. Neither the set of labels, nor the temporal boundaries are exposed to our algorithm since the set-up is completely unsupervised. We evaluate our algorithm against the several unsupervised clustering baselines and state-of-the-art algorithms from video summarization literature, which are applicable.

4.6.1 Dataset

We use WikiHow[2] in order to obtain the top100 queries the Internet users are interested in and choose the ones, which are directly related to the physical world. Resulting queries are;

How to*Bake Boneless Skinless Chicken, Make Jello Shots, Cook Steak, Bake Chicken Breast, Hard Boil an Egg, Make Yogurt, Make a Milkshake, Make Beef Jerky, Tie a Tie, Clean a Coffee Maker, Make Scrambled Eggs, Broil Steak, Cook an Omelet, Make Ice Cream, Make Pancakes, Remove Gum from Clothes, Unclog a Bathtub Drain*

For each of the queries, we crawled YouTube and got the top 100 videos. We also downloaded the English subtitles if they exist. We further randomly choose 5 videos out of 100 per query. Although the choice was random, we discarded outlier videos at this stage and re-sampled without replacement to have 5 inlier evaluation video per query. Other than outlier removal, no human super vision is used to choose evaluation videos. Hence, we have total of 125 evaluation videos and 2375 unlabeled videos.

For each evaluation video, we asked an independent labeler to label them. The dataset is labeled by 5 independent labelers each annotating 5 categories. We asked labelers to label start and end frame of each activity step as well as the name of the step. We simply asked them the question *What are the activity steps and where does each of starts and end?*. All labelers are shown 5 wikiHow[2] video recipes with detailed steps before starting the annotation process as a baseline.

Outlier Video Removal

The video collection we obtain without any expert intervention might have outliers; since, our queries are typical daily activities and there are many cartoons, funny videos, and music videos about them. Hence, we have an automatic filtering stage. The key-idea behind the filtering algorithm is the fact that instructional videos have a distinguishable text description when compared with outliers. To exploit this, we use a clustering algorithm to find the large cluster of instructional videos

with no outlier. Given a large video collection, we use the graph we explain in Section 4.4 and compute the dominant video cluster by using the Single Cluster Graph Partitioning [144] and discard the remaining videos as outlier. We represent each video as a bag-of-words of their textual description. In Figure 4.7, we visualize some of the discarded videos. Although our algorithm have false positives while detecting outliers, we always have enough number of videos (minimum 50) after the outlier detection thanks to the large-scale dataset.

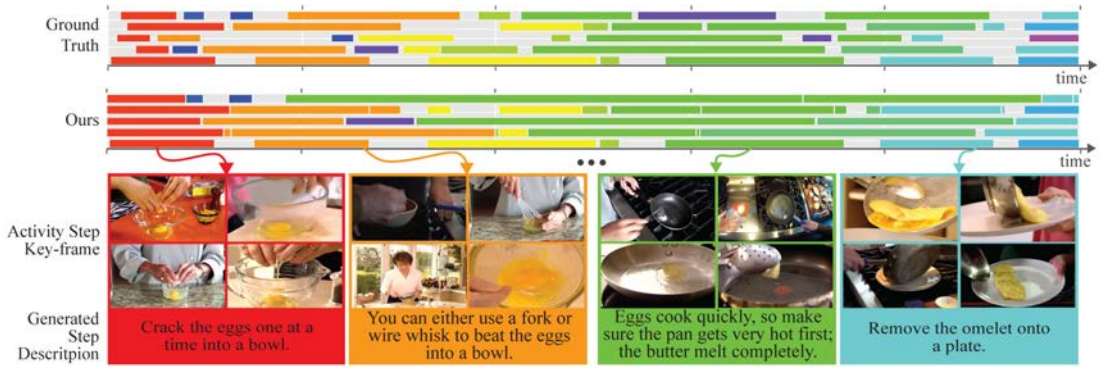
Figure 4.7: **Sample videos which our algorithm discards as an outlier for various queries.** A toy milkshake, a milkshake charm, a funny video about How to NOT make smoothie, a video about the danger of a fire, a cartoon video, a neck-tie video erroneously labeled as bow-tie, a song, and a lamb cooking mislabeled as chicken.

4.6.2 Qualitative Results

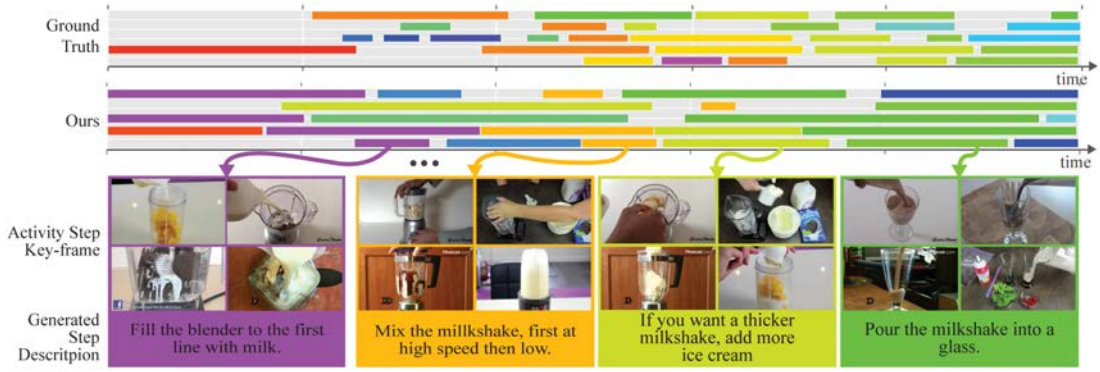
After independently running our algorithm on all categories, we discover activity steps and parse the videos according to discovered steps. We visualize some of these categories qualitatively in Figure 4.8 with the temporal parsing of evaluation videos as well as the ground truth parsing.

To visualize the content of each activity step, we display key-frames from dif-

ferent videos. We also train a 3rd order Markov language model[172] by using the subtitles. Moreover, we generate a caption for each activity step by sampling this model conditioned on the θ_k^l. We explain the details of this process in the appendix.



(a) How to make an omelet?



(b) How to make a milkshake?

Figure 4.8: **Video storylines for queries *How to make an omelet?* and *How to make a milkshake?*** Temporal segmentation of the videos and ground truth segmentation. We also color code the activity steps we discovered and visualize their key-frames and the automatically generated captions. *Best viewed in color.*

As shown in the Figures 4.8(a)&4.8(b), resulting steps are semantically meaningful. Moreover, the language captions are also quite informative hence we can conclude that there is enough language context within the subtitles in order to detect activities. On the other hand, some of the activity steps always occur together

and our algorithm merges them into a single step while promoting sparsity.

4.6.3 Quantitative Results

We compare our algorithm with the following baselines.

Low-level features (LLF):

In order to experiment the effect of learned atoms, we compare with low-level features. As features, we use the state-of-the-art Fisher vector representation of HOG, HOF and MBH features [82].

Single modality:

To experiment the effect of multi-modal approach, we compare with single modality approach by only using the atoms of a single modality.

Hidden Markov Model (HMM):

To experiment the effect of joint generative model, we compare our algorithm with an HMM. We use the Baum-Welch [154] with cross-validation.

Kernel Temporal Segmentation[149]:

Kernel Temporal Segmentation (KTS) proposed by Potapov et al.[149] can detect the temporal boundaries of the events/activities in the video from a time series data

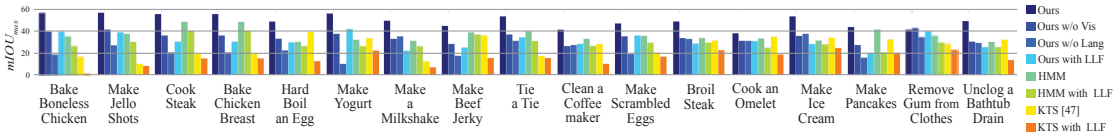


Figure 4.9: IOU_{max} **values for all categories, for all competing algorithms.** Results suggest that our algorithm is outperforming all other baselines. It also suggests that the visual information is contributing more than language for temporal intersection over union. This is rather expected since people tend to talk about things they did and will do; hence, language is expected to have low localization accuracy.

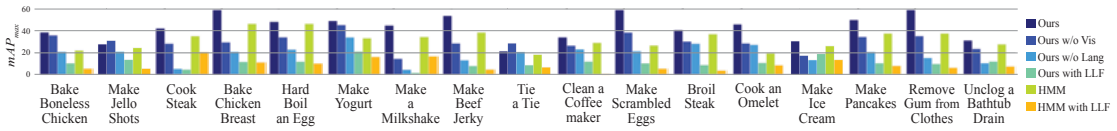


Figure 4.10: AP_{max} **values for all categories, for all competing algorithms.** Results suggest that our algorithm is outperforming all other baselines in most of the cases. The failure cases included recipes like *How to tie a tie?* which is rather expected since a video about tying a tie only includes a tie in the scene which is not informative enough to distinguish steps. The results also suggest that language contributes more than visual information for average precision, which is also rather expected since same step has very high visual variance and generally referred by using same or similar words.

without any supervision. It enforces a local similarity of each resultant segment.

Given parsing results and the ground truth, we evaluate both the quality of temporal segmentation and the activity step discovery. We base our evaluation on two widely used metrics; intersection over union (IOU) and mean average precision(mAP). **IOU** measures the quality of temporal segmentation and it is defined as; $\frac{1}{N}\sum_{i=1}^{N}\frac{\tau_i^\star \cap \tau_i'}{\tau_i^\star \cup \tau_i'}$ where N is the number of segments, $\tau_i^\star$ is ground truth segment and τ_i' is the detected segment. **mAP** is defined per activity step and can be computed based on a precision-recall curve [82]. In order to adopt these metrics

Figure 4.11: **Qualitative results for parsing 'Travel San Francisco' category.** The results suggest that our algorithm can generalize categories beyond instructional videos. For example, travel videos can also be parsed using our method.

into unsupervised setting, we use cluster similarity measure(csm)[123] that enables us to use any metric in unsupervised setting. It chooses a matching of ground truth labels with predicted labels by searching over all matching and choosing the ones giving highest score. We use mAP_{csm} and IOU_{csm} as evaluation metrics.

Accuracy of the temporal parsing. We compute, and plot in Figure4.9, the IOU_{cms} values for all competing algorithms and all categories. We also average over the categories and summarize the results in the Table 4.1. As the Figure 4.9 and Table 4.1 suggests, proposed method consistently outperforms the competing algorithms and its variations. One interesting observation is the importance of both modalities as a result of dramatic difference between the accuracy of our method and its single modal versions.

Moreover, the difference between our method and HMM is also significant. We believe this is due to the ill-posed definition of activities in HMM since the granularity of the activity steps is subjective. On the other hand, our method starts with the well-defined definition of finding set of steps, which generate the entire collection. Hence, our algorithm does not suffer from granularity problem.

Table 4.1: **Average of IOU_{cms} and mAP_{cms} over recipes.** The results suggest that our algorithm outperforms all baselines. The results also suggest that both of the modalities as well as semantic representations of visual information are all required for successful parsing of video collections.

	KTS [149] w/ LLF	KTS[149] w/ Sem	HMM w/ LLF	HMM w/Sem	Ours w/ LLF	Ours w/o Vis	Ours w/o Lang	Our full
IOU_{cms}	16.80	28.01	30.84	37.69	33.16	36.50	29.91	52.36
mAP_{cms}	n/a	n/a	9.35	32.30	11.33	30.50	19.50	44.09

Coherency and accuracy of activity step discovery. Although IOU_{cms} successfully measures the accuracy of the temporal segmentation, it can not measure the quality of discovered activities. In other words, we also need to evaluate the consistency of the activity steps detected over multiple videos. For this, we use unsupervised version of mean average precision mAP_{cms}. We plot the mAP_{cms} values per category in Figure 4.10 and their average over categories in Table 4.1. As the Figure 4.10 and the Table 4.1 suggests, our proposed method outperforms all competing algorithms. One interesting observation is the significant difference between semantic and low-level features. Hence, the mid-level features are key for linking multiple videos.

Semantics of activity steps. In order to evaluate the role of semantics, we performed a subjective analysis. We concatenated the activity step labels in the ground-truth into a label collection. Then, we ask non-expert users to choose a label for each discovered activity for each algorithm. In other words, we replaced the maximization step with subjective labels. We designed our experiments in a way that each clip received annotations from 5 different users. We randomized the ordering of videos and algorithms during the subjective evaluation. Using the labels provided by subjects, we compute the mean average precision (mAP_{sem}).

Table 4.2: **Semantic mean-average-precision mAP_{sem}.** The results suggest that our algorithm outperforms all baselines. The results also suggest that both of the modalities are required for accurate parsing for video collections.

	HMM w/ LLF	HMM w/Sem	Ours w/ LLF	Ours w/o Vis	Ours w/o Lang	Our full
mAP_{sem}	6.44	24.83	7.28	28.93	14.83	39.01

Both mAP_{cms} and mAP_{sem} metrics suggest that our method consistently outperforms the competing ones. There is only one recipe in which our method is outperformed by our based line of no visual information. This is mostly because of the specific nature of the recipe *How to tie a tie?*. In such videos the notion of object is not useful since all videos use a single object -tie-.

The importance of each modality. As shown in Figure 4.9 and 4.10, performance significantly drops when any of the modalities is ignored consistently in all categories. Hence, the joint usage is necessary. One interesting observation is the fact that using only language information performed slightly better than using only visual information. We believe this is due to the less intra-class variance in the language modality (*i.e.*people use same words for same activities). However, it lacks many details(less complete) and more noisy than visual information. Hence these results validate the complementary nature of language and vision.

Generalization to generic structured videos. We experiment the applicability of our method beyond How-To videos by evaluating it on non-How-To categories. In Figure 4.11, we visualize the results for the videos retrieved using the query "Travel San Francisco". The resulting clusters follow semantically meaningful activities and landmarks and show the applicability of our method beyond

How-To queries. It is interesting to note that Chinatown and Clement St ended up in the same cluster. Considering the fact that Clement St is known for its Chinese food, this suggests that the discovered clusters are semantically meaningful.

Noise in the subtitles. We experiment and analyze the robustness to noise in subtitles. Handling noisy subtitles is an important requirement since the scale of large-video collections makes it intractable to transcribe all instructional videos. One study suggests that it would take 374k human-year efforts to transcribe all Youtube videos. Hence, we expect to have combination of automatic speech recognition(ASR) generated subtitles with user uploaded ones as an input to any unsupervised parsing algorithm.

We study the effect of noise, introduced by ASR, by evaluating our algorithm on three different video corpuses. First, we only use the videos with user uploaded subtitles. Second, we only use the videos with ASR generated subtitles. Third, we use the entire dataset as union of first two. The results are summarized in Table 4.3. Results indicate that noise-free subtitle improves the accuracy as expected. Moreover, the difference between the results obtained with full corpus and user uploaded subtitles corpus is very small when compared with ASR only corpus. Hence, our algorithm can fuse information from noisy and noise-free examples in order to compensate for errors in the ASR.

4.7 Grounding into Robotic Instructions

In this section, we demonstrate how we can apply our algorithm to the task of grounding recipe steps into robotic actions.

Table 4.3: **Average of IOU_{cms} and mAP_{cms} over recipes with and without user uploaded subtitles.** The results show that noise in the subtitles has an effect in the parsing accuracy. The results also indicate that our algorithm shows robustness to the noise since our accuracy results are comparable to the version using only user uploaded subtitles.

	IOU_{cms}	mAP_{cms}	mAP_{sem}
ASR only	47.61	39.13	33.27
User uploaded only	54.63	46.21	42.32
Combination	52.36	44.09	39.01

One of the most important applications of our algorithms is in robotics. In future, robots will need to perform many tasks upon user's requests. We envision that the robots can use our video parser to first download a large video collection for a task and then parse it. For example, if a user asks to the robot *Please make a ramen.*, the robot can download all videos returned from the query *How to make a ramen.*. Robot can further parse the scene using any of the available RGB-D/RGB/Point Cloud segmentation algorithms [5, 157, 8]. Robot can use the resulting segmentation in order to find the most similar recipe simply using the object categories, which we output.

In order to demonstrate this application, we use a state of the art language grounding algorithm [136, 135] that can convert the generated descriptions into robot actions based on the environment. Tell Me Dave algorithm of Misra et al[136, 135] uses a semantic simulator, which encodes the common sense knowledge about the physical world. It takes the tuple of language, instructions and the environment as an input and outputs a series of robot actions to perform the task. In order to learn the transformation, it uses a large-scale game log of language instruction, environment and robot action tuples, and models them in terms of

a graphical model. The environment is defined in terms of objects and their 3D positions, language is series of free-form English sentences describing each step and actions are low-level robot commands.

In our experimental setup, we choose two basic household actions, which Tell Me Dave can simulate; namely, *How to make a ramen?* and *How to make an affogato?*. We also chose a random environment for each query from Tell Me Dave environment dataset. We directly feed aforementioned how-to queries into YouTube and parse the resulting video collections. The resulting storylines are visualized in Figure 4.12.

To complete the loop until low-level robot commands, we than manually labeled the object categories our algorithm discovered. For example, if the category we discovered is mostly images of eggs, we labeled this category as *egg*. This step can easily be automated using any object recognition algorithm [35].

By using these labels, our algorithm chose the video, which is closest to the environment query. We simply use the video having maximum number of objects from the environment. In other words, our algorithm finds the recipe video, which is superset to the environment our robot lives in. Finally, we feed the environment and generated captions into the Tell Me Dave algorithm to obtain the physical plan robots need to execute to perform each of the activity. We visualize each plan and the simulation in the Figure 4.12.

Our results are shown in the Figure 4.12, demonstrating that our approach can be used for robotics applications with limited supervision. There were some errors in translation of video storyline steps to actual grounded steps. Example errors include turning on both of the knobs of the stove other than the single

one. However, the resulting plans were still feasible in a way they can accomplish the required high-level task. Therefore, we believe our proposed method is an important direction in personal robotics.

4.8 Conclusions

In this chapter, we captured the underlying structure of human communication by jointly considering visual and language cues. We experimentally validate that given a large-video collection having subtitles, it is possible to discover activities without any supervision over activities or objects. We also demonstrated that the resulting discovered recipes are useful in robotics scenarios.

So; "is it possible to understand large-video collections without any supervision?". Given video and speech information, the storylines we generate successfully summarize the large video collection. This compact representation of a hundreds of videos is expected to be useful in designing user interfaces, which users interact with instructional videos. We believe this is an important step in the direction of future video webpages. Yet another very important question is "can machines understand large-video collections?". Clearly, we needed a small amount of manual information and even used a method, which is trained with human supervision in our robotic demonstration. Hence, it is too early to claim a success for machines watching large-collection of videos. On the other hand, the results are very promising and we believe algorithms, which can convert a free-form input query into robot trajectories, are possible in near future. We also believe our algorithm is an important step in this ambitious target.

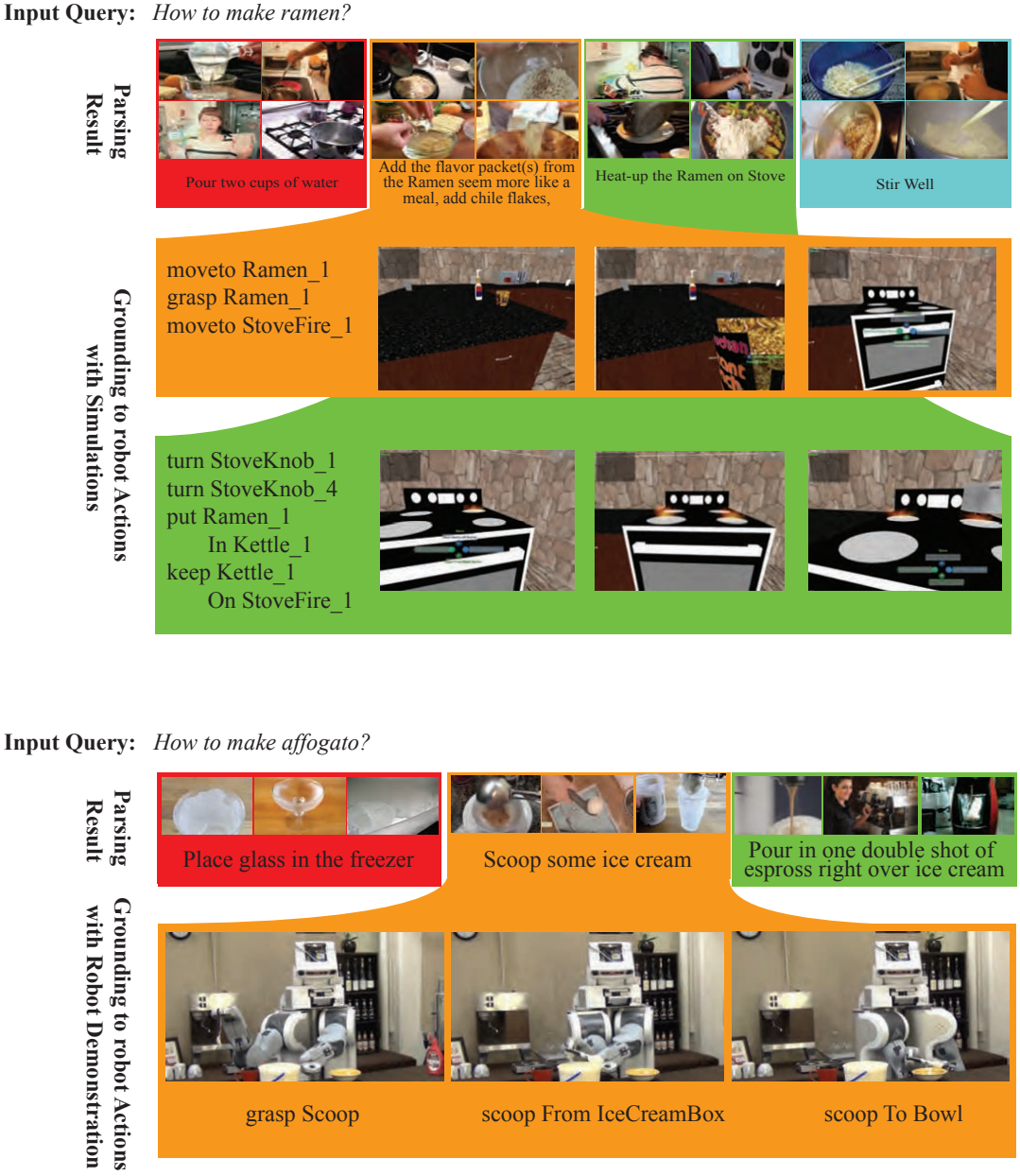


Figure 4.12: **Demonstration on robotic grounding.** We considered two queries by the user: *How to make a ramen?* and *How to make an affogato?*. Given the result by our video parsing system, we find the grounded instruction for each recipe step. Top row shows the results as a storyline from our video parser, and the bottom row shows the robotic simulator and an actual robotic demonstration respectively. During this demonstration, we manually label each object category and fully automate rest of the task. In order to simulate/and implement the resulting steps on robots, we simply used the publicly available simulator/source code distributed by Tell Me Dave [136, 135]

CHAPTER 5

SHARING AT SCALE - SHARING KNOWLEDGE BETWEEN DOMAINS

There are things known and there are things unknown, and in between are the doors of perception.

Aldous Huxley

Recently, deep convolutional neural networks [105, 175, 186] have propelled unprecedented advances in artificial intelligence including object recognition, speech recognition, and image captioning. One of the major drawbacks of the method is that the network requires a lot of labelled training data to fit millions of parameters in the complex network model. However, creating such datasets with complete annotations is not only tedious and error prone, but also extremely costly. More importantly in robotics systems, supervision is typically available for only subset of the domains. For example, it might be the case that we have supervision for objects but not for humans. In this regard, the research community has proposed different mechanisms such as semi-supervised learning [176, 209, 207], transfer learning [155, 191], weakly labelled learning, and domain adaptation. Among these approaches, domain adaptation is one of the most appealing techniques when a fully annotated dataset (e.g. ImageNet [34], Sports1M [88]) is available as a reference.

Formally, the goal of unsupervised domain adaptation is: given a fully labeled source dataset and an unlabeled target dataset, to learn a model which can generalize to the target domain while taking the domain shift across the datasets

into account. The majority of the literature [61, 183, 48, 182, 192] in unsupervised domain adaptation formulates a learning problem where the task is to find a transformation matrix to align the labelled source data distribution to the unlabelled target data distribution. Although these approaches show promising results, they do not take the actual target inference procedure into the learning algorithm. We solve this problem by incorporating the unknown target labels into the training procedure.

Concretely, we formulate a unified framework where the domain transformation parameter and the target labels are jointly optimized in two alternating stages. In the transduction stage, given a fixed domain transform parameter, we jointly infer all target labels by solving a discrete multi-label energy minimization problem. In the adaptation stage, given a fixed target label assignment, we seek to find the optimal asymmetric metric between the source and the target data. The advantage of our method is that we can learn a domain transformation parameter, which is aware of the subsequent transductive inference procedure.

5.1 Related Work

This chapter is closely related to two active research areas: (1) Unsupervised domain adaptation, and (2) Transductive learning.

Unsupervised domain adaptation: [61, 183, 48, 182] proposed subspace alignment based approaches to unsupervised domain adaptation where the task is to learn a joint transformation and projection where the difference between the source and the target covariance is minimized. However, these methods learn the transform matrices on the whole source and target dataset without utilizing the

source labels.

[192] utilizes local max margin metric learning objective [200] to first assign the target labels with the nearest neighbor scheme and then learn a distance metric to enforce that the negative pairwise distances are larger than the positive pairwise distances. However, this method learns a symmetric distance matrix shared by both the source and the target domains so the method is susceptible to the discrepancies between the source and the target distributions. Recently, [55, 195] proposed a deep learning based method to learn domain invariant features by providing the reversed gradient signal from the binary domain classifiers. Although this method performs better than aforementioned approaches, their accuracy is limited since domain invariance does not necessarily imply discriminative features in the target domain.

Transductive learning: In the transductive learning [54], the model has access to unlabelled test samples during training. Recently, [90] tackled a classification problem where predictions are made jointly across all test examples in a transductive [54] setting. The method essentially enforces the notion that the true labels vary smoothly with respect to the input data. We extend this notion to infer the labels of unsupervised target data in a k-NN graph.

To summarize, our main contribution is to formulate a joint optimization framework where we alternate between inferring target labels via discrete energy minimization (*transduction*) and learning an asymmetric transformation (*adaptation*) between source and target examples. Our experiments on digit classification using MNIST [118] and SVHN[140] as well as the object recognition experiments on Office [164] datasets show state of the art results outperforming all existing methods by a substantial margin.

5.2 Method

We address the problem of unsupervised transductive domain adaptation by jointly solving for the label assignment of unsupervised target domain as well as the shift between the domains. We first define our model in Section 5.2.1 and explain the two sub-problems of transduction and adaptation. We further explain the details of transduction in Section 5.2.2 and the details of adaptation in Section 5.2.3.

5.2.1 Problem Definition

In the unsupervised domain adaptation problem, one of the domains (source) is fully supervised $\{\hat{\mathbf{x}}_i, \hat{y}_i\}_{i\in[N^s]}$ with N^s data points $\hat{\mathbf{x}}_i$ and corresponding labels $\hat{y}_i$ from a discrete set $\hat{y}_i \in \{1, \ldots, k\}$. The other domain (target), on the other hand is unsupervised and has N^u data points $\{\mathbf{x}_i\}_{i\in[N^u]}$.

We further assume that both domains have different distributions $\hat{\mathbf{x}}_i \sim p_s$ and $\mathbf{x}_i \sim p_t$ defined on the same space as $\hat{\mathbf{x}}_i, \mathbf{x}_i \in \mathcal{X}$ and there exists a feature function $\Phi : \mathcal{X} \to \mathcal{R}^d$ that is applicable to both. We further study the case where the feature function is parametric with a parameter θ defined as $\Phi_\theta : \mathcal{X} \to \mathcal{R}^d$, and we develop a method to learn the parameters.

Our model has two main components, transduction and adaptation. The transduction is the sub-problem of labelling unsupervised data points and the adaptation is solving for the domain shift.

For adaptation, we explicitly model the domain shift in the form of an asymmetric similarity as

$$s_W(\hat{\mathbf{x}}_i, \mathbf{x}_j) = \Phi(\hat{\mathbf{x}}_i)^\intercal \mathbf{W} \Phi(\mathbf{x}_j) \tag{5.1}$$

such that it is high if two data points, $\hat{\mathbf{x}}_i$ from the source and $\mathbf{x}_j$ from the target, are from the same class.

We further model our transduction in the form of a nearest neighbor and we follow the triplet loss defined in [200] in order to solve adaptation by taking the nearest neighbor inference into learning. While the original triplet loss [200] enforces a margin α between the similarity of any point to its nearest neighbor from the same class and the nearest neighbor from other classes, we extend this construction to the unsupervised domain adaptation by enforcing a similar margin. For each source point, we enforce a margin between its similarity with the nearest neighbor from the target having the same label and having a different label as; $s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^+}) > s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^-}) + \alpha$ where $\mathbf{x}_{i^+}$ is the nearest target having the same class as $\hat{\mathbf{x}}_i$ and $\mathbf{x}_{i^-}$ is the nearest target having a different class label.

Since we model our problem as transduction, we include target labels as part of the joint learning and introduce a target label consistency term as well. We enforce that similar unsupervised data points should have the same label after the transduction by penalizing label disagreements between similar images.

Our model leads to the following optimization problem, over the target labels y_i and the similarity metric $\mathbf{W}$, jointly solving transduction and adaptation.

$$\begin{aligned} \min_{\mathbf{W}, y_1, \ldots y_{N^u}} & \sum_{i \in [N^s]} [s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^-}) - s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^+}) + \alpha]_+ \\ & + \lambda \sum_{i \in N^u} \sum_{j \in \mathcal{N}(\mathbf{x}_i)} sim(\mathbf{x}_i, \mathbf{x}_j) \mathbb{1}(y_i \neq y_j) \\ s.t. \quad & i^+ = \arg\max_{j | y_j = \hat{y}_i} s_W(\hat{\mathbf{x}}_i, \mathbf{x}_j) \\ & i^- = \arg\max_{j | y_j \neq \hat{y}_i} s_W(\hat{\mathbf{x}}_i, \mathbf{x}_j) \end{aligned} \tag{5.2}$$

where $\mathbb{1}(a)$ is an indicator function which is 1 if a is true and 0 otherwise. $[a]_+$ is a rectifier function which is equal to $\max(0, a)$, and sim is any similarity function.

We use cosine similarity as $sim(\mathbf{x}_i, \mathbf{x}_j) = \frac{\Phi(\mathbf{x}_i)^\intercal \Phi(\mathbf{x}_j)}{|\Phi(\mathbf{x}_i)||\Phi(\mathbf{x}_j)|}$.

We solve this optimization problem via alternating minimization through iterating over solving for unsupervised labels y_i(transduction) and learning the similarity metric $\mathbf{W}$ (adaptation). We explain these two steps in detail in the following sections.

5.2.2 Transduction: Labeling Target Domain

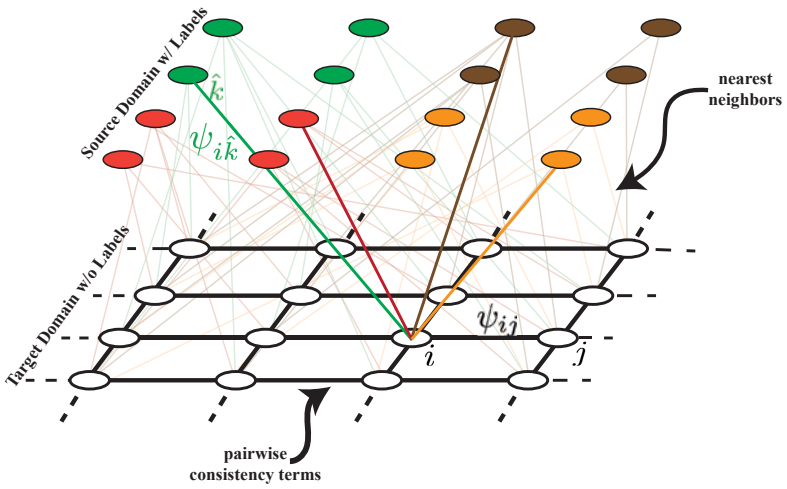


Figure 5.1: **Visualization of the Label Propogation.** We create a k-NN graph of unsupervised target points to enforce consistency with pairwise terms $\psi_{ij} = sim(\mathbf{x}_i, \mathbf{x}_j)\mathbb{1}(y_i \neq y_j)$ and use closest supervised source points to each class as $\psi_{i\hat{k}} s_W(\hat{\mathbf{x}}_k, \mathbf{x}_i)$.

In order to label the unsupervised points, we use the nearest-neighbor rule. We simply compute the NN supervised data point for each unsupervised data point using the learned metric $s_W(\cdot, \cdot)$ and transfer the corresponding label. In the initial stages of our optimization, our tranduction needs to be accurate even with a sub-optimal similarity metric due to the iterative fashion of our algorithm. Hence, we enforce a consistent labeling via label propagation. We first formally define the

NN-rule and then introduce the label propagation.

Given a similarity metric $s_W(\cdot,\cdot)$, the NN rule is:

$$(y_i)^{pred} = \hat{y}_{\arg\max_j s_W(\mathbf{x_i},\hat{\mathbf{x}}_\mathbf{j})} \tag{5.3}$$

We use label propagation to enforce consistency of the predicted labels of unsupervised data points. Our label propagation is similar to existing graph transduction algorithms [16, 208]. In order to enforce this consistency, we create a k-nearest neighbor (k-NN) graph over the unsupervised data points such that neighbors $\mathcal{N}(\mathbf{x}_i)$ for $\mathbf{x}_i$ is the k-unsupervised data point having highest similarity to $\mathbf{x}_i$ using the cosine similarity in the feature space. After the k-NN graph is created, we solve the following optimization problem for labeling unsupervised data points with label propagation:

$$\begin{aligned}\min_{y_1,\dots y_{N^u}} \sum_{i\in N^u} & -\max_{\hat{y}_j=y_i} s_W(\hat{\mathbf{x}}_j,\mathbf{x}_i) \\ & + \lambda \sum_{i\in N^u}\sum_{j\in\mathcal{N}(\mathbf{x}_i)} sim(\mathbf{x}_i,\mathbf{x}_j)\mathbb{1}(y_i\neq y_j)\end{aligned} \tag{5.4}$$

This problem can approximately be solved using many existing methods such as α-β swapping, quadratic pseudo-boolean optimization (QPBO) and linear programming through roof-duality. We use the α-β swapping algorithm from [22] since it is experimentally shown to be efficient and accurate. In order to further explain the label propagation, we visualize an example with $k = 4$ and 4-class classification problem in Figure 5.1.

It is also critical that this formulation requires solving high number of nearest neighbors, which is computationally challenging. However, our choice of optimization method makes this computation tractable. We use stochastic gradient descent in our adaptation stage with a carefully chosen batch size, which requires us to only solve the transduction over a batch.

5.2.3 Adaptation: Learning the Metric

Given the predicted labels y_i for unsupervised data points $\mathbf{x}_i$, we need to learn an asymmetric metric in order to minimize the loss function defined in (5.2).

The main intuition behind our formulation is to seek a metric which can label the supervised points correctly using the unsupervised points and their predicted labels. In other words, we reverse the labeling direction. At this stage we have already predicted a label for each unsupervised point; hence, we can estimate a label for each supervised point using the predicted labels. We also have ground truth labels for the supervised points and we combine them to find an asymmetric metric. In other words, the goal of the adaptation stage is:

- predicting $\hat{y}_j^{pred}$ using $\mathbf{x}_i, \hat{\mathbf{x}}_j, y_i$
- learning $s_W(\cdot,\cdot)$ by penalizing $(\hat{y}_j)^{pred} \neq \hat{y}_j$

Fortunately, this can be jointly solved by minimizing the triplet loss defined with supervised data points and their closest same class and different class neighbors among the unsupervised points. Formally, we find the closest same class and different class points as;

$$\begin{aligned} i^+ &= \arg\max_{j|y_j=\hat{y}_i} s_W(\hat{\mathbf{x}}_i, \mathbf{x}_i) \\ i^- &= \arg\max_{j|y_j\neq\hat{y}_i} s_W(\hat{\mathbf{x}}_i, \mathbf{x}_j) \end{aligned} \tag{5.5}$$

We further define the loss function with a regularizer using the nearest neighbors as: $loss(\mathbf{W}) =$

$$\sum_{i\in[N^s]} [s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^-}) - s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^+}) + \alpha]_+ + r(\mathbf{W}) \tag{5.6}$$

which is convex in terms of the $\mathbf{W}$ if the regularizer is convex; and we optimize it via stochastic gradient descent using the subgradient $\frac{\partial loss(y_i,\mathbf{W})}{\partial \mathbf{W}} = \frac{\partial r(\mathbf{W})}{\partial \mathbf{W}} +$

$$\sum_{i\in[N^s]} \mathbb{1}(s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^-}) - s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^+}) > \alpha) \times (\Phi(\hat{\mathbf{x}}_i)\Phi(\mathbf{x}_{i^-})^\intercal - \Phi(\hat{\mathbf{x}}_i)\Phi(\mathbf{x}_{i^+})^\intercal) \tag{5.7}$$

As a regularizer use the Frobenius norm of the similarity matrix as $r(\mathbf{W}) = \frac{1}{2}\|\mathbf{W}\|_F^2$. We explain the details of this optimization routine in the Section 5.3.3.

5.2.4 Learning Features

In Section 5.2.2 and 5.2.3, we explained our transduction with label propagation as well as the adaptation algorithm using a pre-defined feature function Φ. However, the current trends in machine learning suggest that learning this feature function Φ from the data using deep neural networks is a promising direction especially for visual domains. Hence, we consider the case where Φ_θ is a parameterized feature function with parameter set, θ. A typical example is CNNs (Convolutional Neural Networks) with θ as concatenation of weights and biases in the layers of CNN. We learn the feature function parameters as part of the adaptation stage with an update; $\frac{\partial loss(y_i,\mathbf{W})}{\partial \theta} =$

$$\sum_{i\in[N^s]} \mathbb{1}(s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^-}) - s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^+}) > \alpha) \times \left(\frac{\partial s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^-})}{\partial \theta} - \frac{\partial s_W(\hat{\mathbf{x}}_i, \mathbf{x}_{i^+})}{\partial \theta}\right) \tag{5.8}$$

where $\frac{\partial s_W(\hat{\mathbf{x}}_i,\mathbf{x}_j)}{\partial \theta} = \Phi(\mathbf{x}_j)^\intercal \mathbf{W}^\intercal \frac{\partial \phi_\theta(\hat{\mathbf{x}}_i)}{\partial \theta} + \Phi(\hat{\mathbf{x}}_i)^\intercal \mathbf{W} \frac{\partial \phi_\theta(\mathbf{x}_j)}{\partial \theta}$

Algorithm 2: Transduction with Domain Shift

Input: source $\mathbf{x}_i$, target $\hat{\mathbf{x}}_i$, y_i, batch size B

repeat

Sample $\{\mathbf{x^b}_{1\cdots B}\}$, $\{\hat{\mathbf{x}}^b_{1\cdots B}, \hat{y}^b_{1\cdots B}\}$

Solve (5.4) for $\{y_{1\cdots B}\}$

for $i = 1$ **to** B **do**

if $\hat{y}_i$ **in** $y_{1\cdots y_B}$ **then**

Compute (i^+, i^-) using $\{y_{1\cdots B}\}$ in (5.5)

Update $\frac{\partial loss}{\partial \theta}$ and $\frac{\partial loss}{\partial \mathbf{W}}$ using (5.7,5.8)

end if

end for

$\mathbf{W} \leftarrow \mathbf{W} + \alpha \frac{\partial loss(y_i, \mathbf{W})}{\partial \mathbf{W}}$

$\theta \leftarrow \theta + \alpha \frac{\partial loss(y_i, \mathbf{W})}{\partial \theta}$

until CONVERGENCE or MAX_ITER

5.3 Experimental Results

We evaluate our algorithm on various unsupervised domain adaptation tasks while focusing on two different problems, hand-written digit classification and object recognition. For each experiment, we use three domains and evaluate all adaptation scenarios.

5.3.1 Dataset

We use MNIST[118], Street View House Number[140] and the artificially generated version of MNIST -MNIST-M- [55] to experiment our algorithm on the digit

Table 5.1: Accuracy of our method and the state-of-the-art algorithms on Office dataset and various adaptation settings

Source	Amazon	D-SLR	Webcam	Webcam	Amazon	D-SLR
Target	Webcam	Webcam	D-SLR	Amazon	D-SLR	Amazon
GFK [60]	.398	.791	.746	.371	.379	.379
SA* [48]	.450	.648	.699	.393	.388	.420
DLID [29]	.519	.782	.899	-	-	-
DDC [195]	.618	.950	.985	.522	.644	.521
DAN [125]	.685	.960	.990	.531	.670	.540
Backprop [55]	.730	**.964**	**.992**	.536	.728	.544
Source Only	.642	.961	.978	.452	.668	.476
Our Method	**.804**	.962	.989	**.625**	**.839**	**.567**

Table 5.2: Accuracy on the digit classification task.

Source	M-M	MNIST	SVHN	MNIST
Target	MNIST	M-M	MNIST	SVHN
SA* [48]	.523	.569	.593	.211
BP [55]	.732	.766	.738	.289
Source Only	.483	.522	.549	.162
Our Method	**.835**	**.855**	**.774**	**.323**

classification task. MNIST-M is a simply a blend of the digit images of the original MNIST dataset and the color images of BSDS500[7] following the method explained in [55]. Since the dataset is not distributed directly by the authors, we generated the dataset using the same procedure and further confirmed that the performance is the same as the one reported in [55]. Street View House Numbers

dataset is a collection of house numbers collected directly from Google street view images. Each of these three domains are quite different from each other and among many important differences, the most significant ones are MNIST being gray scale and the others being colored, and SVHN images having extra confusing digits around the centered digit of interest. Moreover, all three domains are large-scale having at least 60k examples over 10 classes.

In addition, we use the Office[164] dataset to evaluate our algorithm on the object recognition task. Office dataset includes images of the objects taken from Amazon, captured with a webcam and captured with a D-SLR. Differences between domains include the white background of Amazon images vs realistic backgrounds of webcam images, and the resolution differences. The Office dataset has fewer images, with a maximum of 2478 per domain with 31 classes.

5.3.2 Baselines

We compare our method with a variety of methods with and without feature learning. Considering the two different lines of work, **SA***[48] is the dominant state-of-the-art approach not employing any feature learning, and **Backprop(BP)**[55] is the dominant state-of-the-art employing feature learning. We use the available source code of [55] and [48] and following the evaluation procedure in [55], we choose the hyper-parameter of [48] as the highest performing one among various alternatives. We also compare our method with the **source only** baseline which is a convolutional neural network trained only using the source data. This classifier is clearly different from our nearest neighbor classifier; however, we experimentally validated that CNN always outperformed the nearest neighbor based classifier. Hence, we report the highest performing source only method.

5.3.3 Implementation Details

Although our algorithm has very few hyper-parameters and we choose most of them either using cross-validation or exhaustive grid search, our algorithm uses an existing differentiable feature function. Following the unparalleled success of convolutional neural networks (CNNs), we use CNNs as our feature functions. In order to have a fair comparison with existing algorithms, we follow the same architecture used by [55] by only changing the final feature dimensionality (embedding size). We use the following architectures for domains:

MNIST and **SVHN:** LeNet[117] as

Office: AlexNet[105] as

where **C** is convolution, **P** is max-pooling, **R** is ReLU and **F** is fully connected layer.

Since the office dataset is quite small, we do not learn the full network for office experiments and instead we only optimize for fully connected layers initializing with the weights pre-trained on ImageNet. In all of our experiments, we set the feature dimension as 128. We use stochastic gradient descent to learn the feature function as well as the similarity metric with AdaGrad[42]. We initialize variables with truncated normals having unit variance and use the learning rate 2.5E$-$4 and the batch size 256.

5.3.4 Evaluation Procedure

We evaluate all algorithms in fully transductive setup following the standard evaluation setup of [164]. We feed training images and labels of the first domain as the source and training images of the second domain as the target. We further evaluate the accuracy on the target domain labels as the ratio of correctly labeled images to all target images.

5.3.5 Results

Following the fully transductive evaluation, we summarize the results in Table 5.1 and Table 5.2. Table 5.1 summarizes the results on the object recognition task using office dataset whereas Table 5.2 summarizes the digit classification task on MNIST and SVHN.

Table 5.1&5.2 shows results on object recognition and digit classification tasks exhaustively covering all adaptation scenarios. Our algorithm shows state-of-the-art performance. Moreover, our algorithm significantly outperforms all state-of-the-art methods when there is a large domain difference like MNIST↔MNIST-M, MNIST↔SVHN, Amazon↔Webcam and Amazon↔D-SLR. We hypothesize this performance is due to the transductive modeling. State-of-the-art algorithms like [55] are seeking for set of features invariant to the domains whereas we seek for an explicit similarity metric explaining both differences and similarities of domains. In other words, instead of seeking for an invariance, we seek for an equivariance.

Table 5.1 suggests that accuracy of our algorithm is limited for D-SLR↔Webcam experiments. This is rather expected since the domain difference

is very minor between D-SLR and webcam images and the minor domain difference results in saturation of accuracies for all algorithms. Moreover, since we use nearest neighbor classifier, our algorithm needs a large-dataset to be successful. Both webcam and D-SLR datasets are rather small (300 to 700 examples) which limits the accuracy of nearest neighbor algorithm as well.

Table 5.2 further suggests our algorithm is the only one, which can generalize that well from MNIST to SVHN dataset. We believe this is thanks to the feature learning in the transductive setup. Clearly the features, which are learned from MNIST cannot generalize to SVHN since the SVHN has concepts like color and occlusion, which are not available in MNIST. Hence, our algorithm learns SVHN specific features by enforcing accurate transduction in the adaptation stage.

Another interesting conclusion is the asymmetric nature of the results. For example, the accuracy of adapting webcam to amazon and adapting Amazon to webcam is significantly different in Table 5.1. The similar behavior exists in MNIST and SVHN domains as well in Table 5.1. This observation validates the importance of an asymmetric modeling.

Qualitative Analysis

To further study the learned representations as well as the similarity metric, we perform a series of qualitative analysis in the form of nearest neighbor analyses and tSNE[196] plots.

In Figure 5.3, we visualize example target images from MNIST and their corresponding source images. First of all, both our experimental procedure and qualitative analysis suggest that MNIST and SVHN are the two domains with the

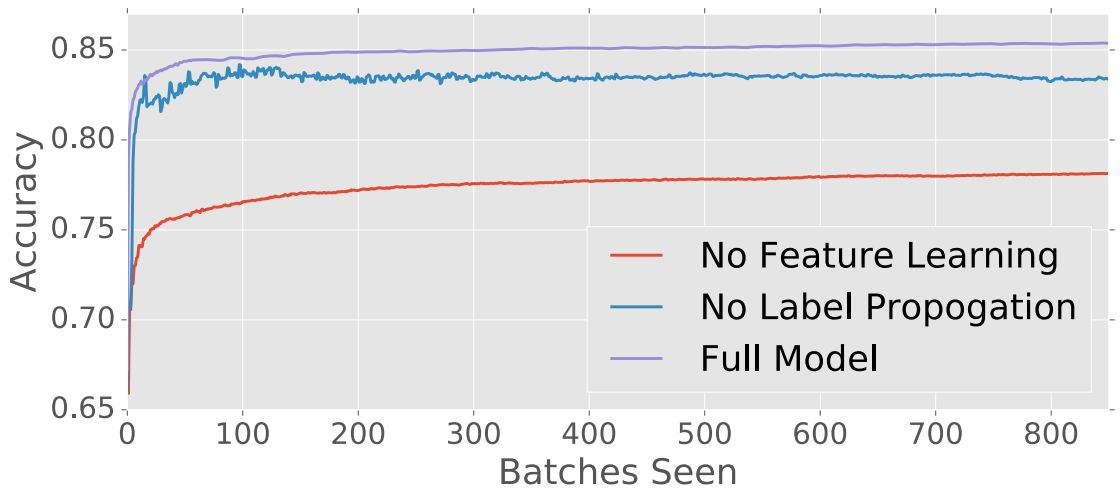


Figure 5.2: Accuracy vs number of iterations for our method and its variant without label propagation as well as the variant without feature learning. As the figure suggests the label propagation increases both the stability of the gradients as well as the final accuracy. Moreover, the feature learning also has a significant effect on the accuracy.

largest difference. Hence, we believe MNIST↔SVHN is very challenging set-up and despite the huge visual differences, our algorithm results in accurate nearest neighbors.

In Figure 5.4, we visualize example target images from webcam and their corresponding nearest source images from Amazon. The accuracy of the domain adaptation is also visible in this task.

The difference between invariance and equivariance is clearer in the tSNE plots of the Office dataset in Figure 5.5 as well as digit classification task Figure 5.6. In Figure 5.5, we plot the distribution of features before and after adaptation for source and target while color coding class labels. As Figure 5.5 suggests, the source domain is well clustered according to the object classes with and without adaptation. Moreover, this is expected since the features are specifically fine-tuned to the source domain before the adaptation starts. However, target domain features

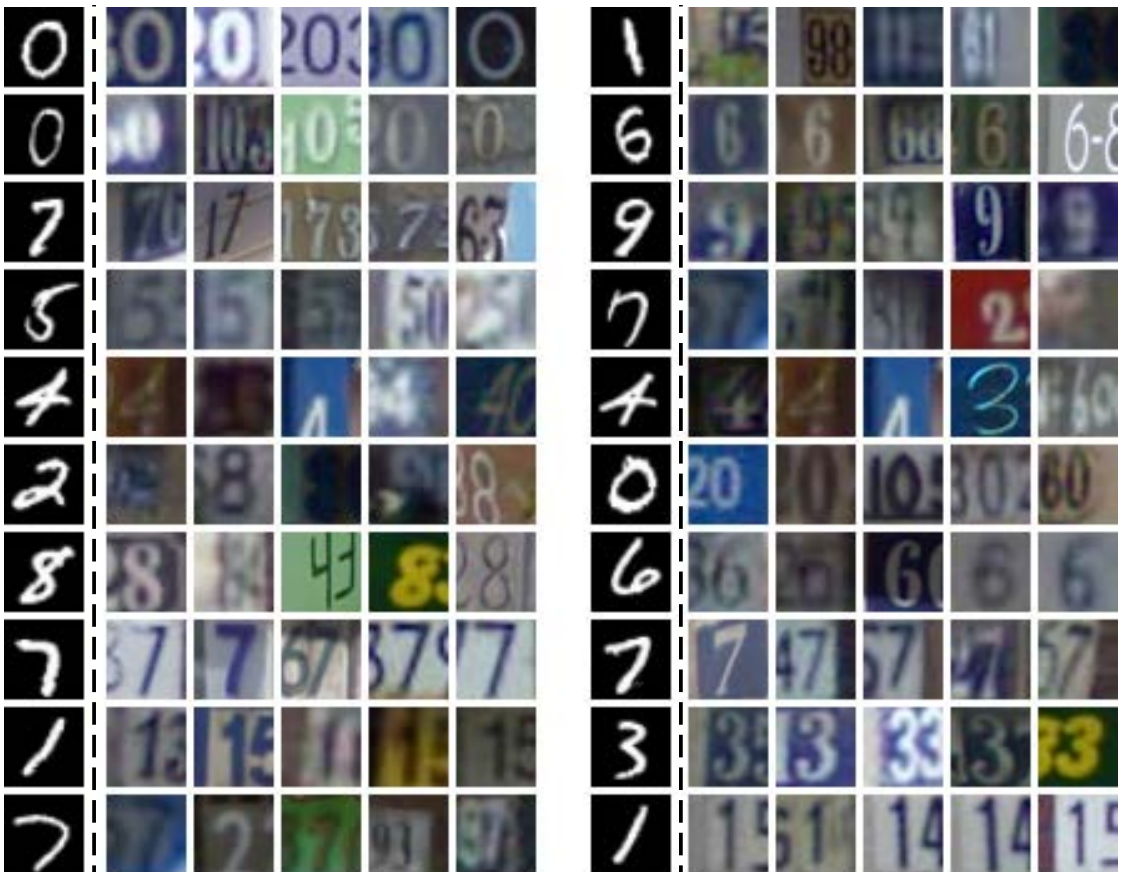


Figure 5.3: Example nearest neighbors for SVHN→MNIST experiment. We show an example MNIST image and 5-NN SVHN images. Please note the large domain difference.

have no structure before adaptation. This is also expected since the algorithm did not see any image from the target domain. After the adaptation, target images also get clustered according to the object classes.

In Figure 5.6, we show the digit images of source and target after the adaptation. Clearly, the target is well clustered according to the classes and source is not very well clustered although it has some structure. Since we learn the entire network for digit classification, our networks learn discriminative features in target domain as our loss depends directly on classification scores in target domain. Moreover, discriminative features in target arise because of the transductive mod-

eling. In comparison, state of the art domain invariance based algorithms only try to be invariant to the domains without explicit modeling of discriminativeness on the target domain. Hence, our similarity metric explicitly models the relationship between the domains and results in an equivariant model while enforcing discriminative behavior in the target. We also draw lines between the nearest neighbor images of source and target images to show the accuracy of the metric function. Moreover, the nearest neighbors are quite accurate confirming the quantitative results.

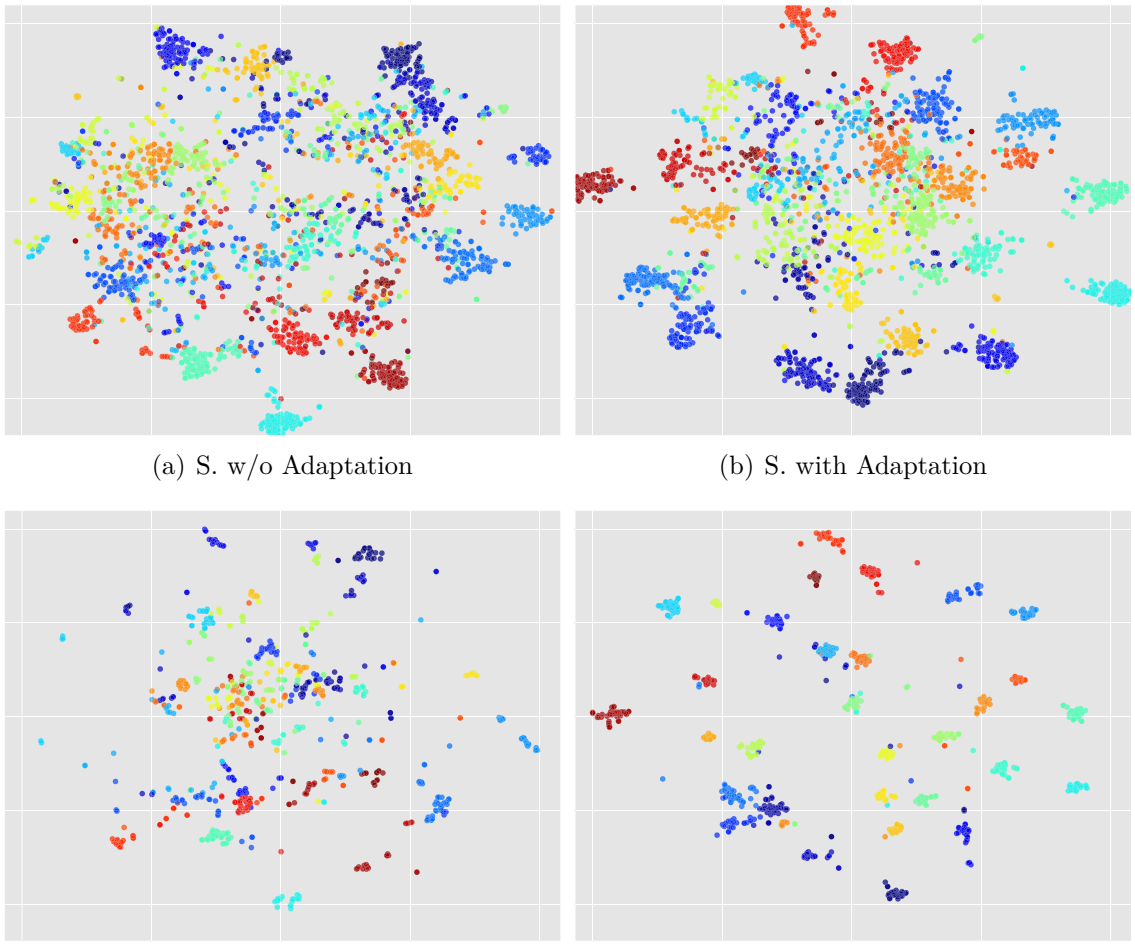
Label propagation & feature learning

In order to evaluate the effect of having robust label propagation and feature learning, we compare our method without the label propagation (noted as *No Label Propogation*) and without feature learning (noted as *No Feature Learning*). We plot the accuracy vs number of iterations in order to evaluate both the effect on learning rate as well as the accuracy. Although we plot the results only for MNIST→MNIST-M, the other experiments have similar results and not displayed for the sake of clarity. Results are shown in the Figure 5.2, and it suggests that both feature learning and label propagation is crucial for successful transduction. Another interesting observation is the unstable behavior when we disable label propagation. This is also expected since without label propagation, the labeling stage will have more mis-classifications and they will decrease the accuracy of the metric.

5.4 Conclusion

We described a transductive approach to the unsupervised domain adaptation problem by defining a joint learning problem on the transductive target label assignment and an asymmetric similarity metric across the domains. We further described a method to learn deep features, which are discriminative in the target domain. Experimental results on digit classification using MNIST[118] and SVHN[140] as well as on object recognition using Office[164] dataset show state of the art performance with a significant margin. We will make our learned models as well as the source code available immediately upon acceptance.

Figure 5.4: Example nearest neighbors for Amazon↔Webcam experiment. We show an example source image and 3-NN target images. The drop in the accuracy after the nearest neighbors is expected since our loss function only models the nearest one.



(a) S. w/o Adaptation

(b) S. with Adaptation

(c) T w/o Adaptation

(d) T with Adaptation

Figure 5.5: tSNE plots for office dataset Webcam(S)→Amazon(T). Source features were discriminative and stayed discriminative as expected. On the other hand, target features became quite discriminative after the adaptation.

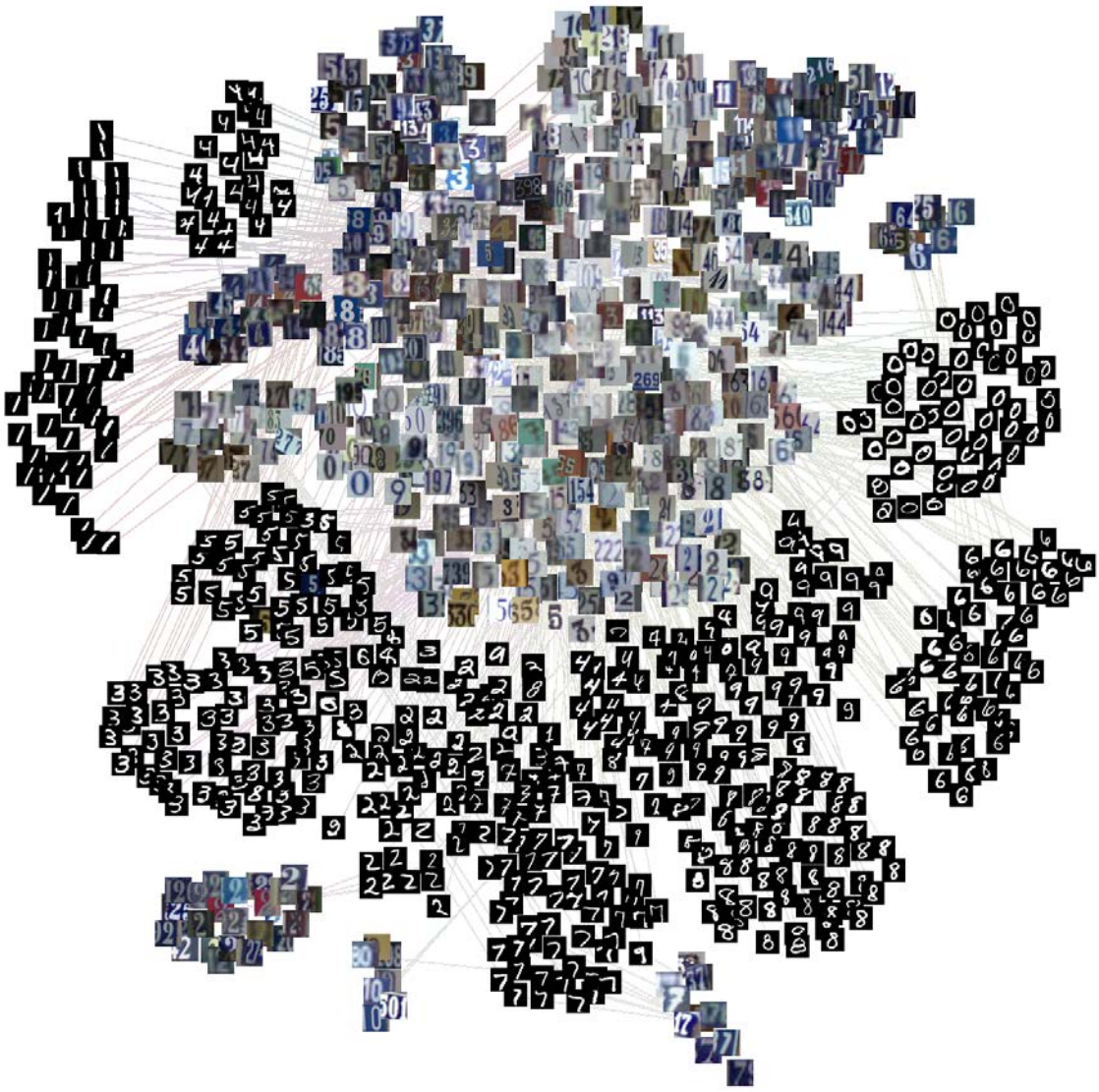


Figure 5.6: tSNE plot for SVHN→MNIST experiment. Please note that the discriminative behavior only emerges in the unsupervised target instead of the source domain. This explains the motivation behind modeling the problem as transduction. In other words, our algorithm is designed to be accurate and discriminative in the target domain which is the domain we are interested in.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

> "Down, down, down. Would the fall never come to an end! I wonder how many miles I've"
>
> *Lewis Carroll*
>
> *Alice's Adventures in Wonderland*

We started this dissertation with the question of "Can we extract structured and physically grounded knowledge from the publicly available information on the web". Although our answer to this question is not a definite "yes" yet, we showed a very promising direction following a study of knowledge-bases, representation learning and uncertainty modeling.

Our first major step was identifying key challenges preventing us from using existing large-scale knowledge bases. After identifying these challenges, we designed and developed a new knowledge based carefully tailored for robots *www.robobrain.me*.

Obtaining a large-scaled knowledge, our next step was developing large-scale machine learning algorithms, which can learn meaningful representations from the available information. Considering the cost of labelling such a large knowledge base, we designed unsupervised video parsing algorithms. Our proposed algorithm discovered emerging patterns from the videos. It learned objects and their dynamics as well as the activities humans perform.

Learning semantic representations of objects and activities from large-scale collection of videos, resulted in an ambitious question. "Can we go from instructional

videos to physical robot plans?". Main rationale behind asking this question was the availability of very large collections of instructional videos in YouTube. For example, there are 200 thousand videos only to describe *how to make a poached egg?*. Such an end-to-end system requires solving the domain shift between planning knowledge and perception knowledge. Moreover, we proposed a domain adaptation algorithm for this purpose and showed a successful robot plans generated completely from large-scale data with no human supervision.

Our final step was handling the common challenge of all data-driven systems, unknowns. Since none of the knowledge-bases are complete, all data-driven systems will eventually face an environment they never saw. We believe the key to handle this is explicitly modeling uncertainty. We proposed an estimation algorithm to explicitly model uncertainty.

Using all these methods, we showed a successful demonstration of robots directly going from unsupervised large-scale data to physically plausible plans in collaboration with humans. Our demonstrations showed a promising direction but we believe there is still a long way to go. In the rest of this section, we discuss the next steps in this ambitious aim.

One big challenge is natural language processing. Within the scope of this thesis, we used very primitive algorithms to parse and understand natural language. We even used a limited human supervision in some of the applications. We believe it is possible to develop very expressive natural language representations for robotics. The key difference to existing natural language processing methods is the representations need to be multi-domain. For example, the representation of "Cup" not only includes its language properties but also physical and visual properties. In other words, we need joint representations of the words to be used

in robotics.

Yet another challenge is the necessity to model common-sense. We all know that butter is most likely to be found in the fridge and screwdriver is probably in the garage. However, we never explicitly mention this kind of knowledge neither in our written text nor in our instructional videos. Hence, we need algorithms, which can learn these common-sense concepts. We hypothesize this is possible by personal robots constantly gathering information from our houses and sharing with them. Sharing is a key here since it is not tractable for a single robot to learn the entire human common-sense in a tractable amount of time.

In summary, we studied the problem of robot perception in this dissertation with a focus on large-scale learning approaches. We showed that large-scale data is extremely useful for robots as long as it satisfy a few properties like being multi-modal, multi-domains and physically grounded. We further showed that visual data have strong structural priors like structural diversity, consistency and hierarchy. We developed unsupervised and multi-domain machine learning algorithms using this priors resulting in state-of-the-art performance in many robot perception tasks. After all, we believe we will be able to answer the aforementioned question with a definite "yes" as long as we continue to exploit the structure and share the representations.

BIBLIOGRAPHY

[1] Rabbit's ph.d. thesis and lion's watch repair business. `https://www.cs.hmc.edu/~fleck/parable.html`. Accessed: 2016-04-20.

[2] Wikihow-how to do anything. `http://www.wikihow.com`, 2015.

[3] P. Abbeel, A. Coates, and A. Y. Ng. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 29(13), 2010.

[4] R. Alami, A. Clodic, V. Montreuil, E. A. Sisbot, and R. Chatila. Toward human-aware robot task planning. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, 2006.

[5] A. Anand, H. S. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for 3d point clouds. *IJRR*, 2012.

[6] R. D. Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Tran. on Automation Science and Engineering (T-ASE)*, 9(4), 2012.

[7] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *T-PAMI*, 33(5):898–916, 2011.

[8] Iro Armeni, Ozan Sener, Amir Zamir, Helen Jiang, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, 2016.

[9] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. *Dbpedia: A nucleus for a web of open data*. Springer, 2007.

[10] Andrei Barbu, Alexander Bridge, Zachary Burchill, Dan Coroian, Sven Dickinson, Sanja Fidler, Aaron Michaux, Sam Mussman, Siddharth

Narayanaswamy, Dhaval Salvi, et al. Video in sentences out. *arXiv preprint arXiv:1204.2742*, 2012.

[11] Kobus Barnard, Pinar Duygulu, David Forsyth, Nando De Freitas, David M Blei, and Michael I Jordan. Matching words and pictures. *JMLR*, 3:1107–1135, 2003.

[12] Dhruv Batra, Payman Yadollahpour, Abner Guzman-Rivera, and Gregory Shakhnarovich. Diverse m-best solutions in markov random fields. In *Proc. ECCV*, pages 1–16. Springer, 2012.

[13] M. Beetz, U. Klank, I. Kresse, A. Maldonado, L. Mosenlechner, D. Pangercic, T. Ruhr, and M. Tenorth. Robotic roommates making pancakes. In *Humanoids*, pages 529–536. IEEE, 2011.

[14] M. Beetz, M. Tenorth, and J. Winkler. âĂIJopen-ease âĂŤ a knowledge processing service for robots and robotics/ai researchers. *TZI Technical Report*, 74, 2014.

[15] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, L Mosenlechner, Dejan Pangercic, T Ruhr, and Moritz Tenorth. Robotic roommates making pancakes. In *Humanoids*, 2011.

[16] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. In *ICML*, 2001.

[17] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic. Weakly supervised action labeling in videos under ordering constraints. In *ECCV*, 2014.

[18] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a

collaboratively created graph database for structuring human knowledge. In *Proc. ACM SIGMOD*, pages 1247–1250, 2008.

[19] M. Bollini, S. Tellex, T. Thompson, M. Roy, and D. Rus. Interpreting and executing recipes with a cooking robot. In *ISER*, 2012.

[20] Mario Bollini, Jennifer Barry, and Daniela Rus. Bakebot: Baking cookies with the pr2. In *The PR2 Workshop, IROS*, 2011.

[21] Konstantinos Bousmalis, Stefanos Zafeiriou, L Morency, and Maja Pantic. Infinite hidden conditional random fields for human behavior analysis. *Neural Networks and Learning Systems, IEEE Trans*, 24(1):170–177, 2013.

[22] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *T-PAMI*, 26(9):1124–1137, 2004.

[23] Rodney A Brooks. Symbolic reasoning among 3-d models and 2-d images. *Artificial intelligence*, 17(1):285–348, 1981.

[24] M. Cakmak, S. S. Srinivasa, M. K. Lee, J. Forlizzi, and S.B. Kiesler. Human preferences for robot-human hand-over configurations. In *IROS*, 2011.

[25] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3, 2010.

[26] Joao Carreira and Cristian Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, 2010.

[27] Chao Chen, Vladimir Kolmogorov, Yan Zhu, Dimitris Metaxas, and

Christoph Lampert. Computing the m most probable modes of a graphical model. In *AISTATS*, pages 161–169, 2013.

[28] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *ICCV*, 2013.

[29] Sumit Chopra, Suhrid Balakrishnan, and Raghuraman Gopalan. Dlid: Deep learning for domain adaptation by interpolating between domains. In *ICML W*, volume 2, page 5, 2013.

[30] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[31] Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *CVPR*, 2013.

[32] Pierre Del Moral. *Mean field simulation for Monte Carlo integration*. CRC Press, 2013.

[33] V. Delaitre, D. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. Efros. Scene semantics from long-term observation of people. In *Proc. ECCV*, 2012.

[34] J. Deng, W. Dong, R. Socher, L-J Li, K. Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[35] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[36] Lewis Carroll (Charles L. Dodgson). *Alice's Adventures in Wonderland*. George MacDonald, 1865.

[37] X.L. Dong, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[38] Arnaud Doucet, Nando De Freitas, Kevin Murphy, and Stuart Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *UAI*, 2000.

[39] A. Dragan and S. Srinivasa. Generating legible motion. In *RSS*, June 2013.

[40] Anca D Dragan and Siddhartha S Srinivasa. Formalizing assistive teleoperation. *RSS*, 2008.

[41] Olivier Duchenne, Ivan Laptev, Josef Sivic, Francis Bash, and Jean Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.

[42] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011.

[43] S. Edelkamp and P. Kissmann. Optimal symbolic planning with action costs and preferences. In *IJCAI*, 2009.

[44] Alexei A. Efros, Alexander C. Berg, Greg Mori, and Jitendra Malik. Recognizing action at a distance. In *ICCV*, 2003.

[45] O. ?ener, K. Ugur, and A. A. Alatan. Efficient mrf energy propagation for video segmentation via bilateral filters. *IEEE Transactions on Multimedia*, 16(5):1292–1302, Aug 2014.

[46] Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. Every picture tells a story: Generating sentences from images. 2010.

[47] C. Fellbaum. *WordNet*. Wiley Online Library, 1998.

[48] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, pages 2960–2967. IEEE, 2013.

[49] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

[50] D. A. Ferrucci. Introduction to this is watson. *IBM J. of RnD*, 56(3.4):1–1, 2012.

[51] Sanja Fidler, Abhishek Sharma, and Raquel Urtasun. A sentence is worth a thousand pixels. In *CVPR*. IEEE, 2013.

[52] E.B. Fox. *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*. Ph.D. thesis, MIT, Cambridge, MA, 2009.

[53] E.B. Fox, M.C. Hughes, E.B. Sudderth, and M.I. Jordan. Joint modeling of multiple related time series via the beta process with application to motion capture segmentation. *Annals of Applied Statistics*, 8(3):1281–1313, 2014.

[54] Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. In *UAI*, pages 148–155. Morgan Kaufmann Publishers Inc., 1998.

[55] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In David Blei and Francis Bach, editors, *ICML*, pages 1180–1189. JMLR Workshop and Conference Proceedings, 2015.

[56] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE PAMI*, (6):721–741, 1984.

[57] James Jerome Gibson. *The ecological approach to visual perception*. Psychology Press, 1986.

[58] M. J. Gielniak, C. Karen Liu, and A. L. Thomaz. Generating human-like motion for robots. *IJRR*, 32(11), 2013.

[59] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[60] Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML*, pages 222–230, 2013.

[61] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, 2012.

[62] Google. Google trends. `https://www.google.com/trends/`, 2016. Accessed: 2016-04-23.

[63] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. Generating photo manipulation tutorials by demonstration. *TOG*, 28(3):66, 2009.

[64] Thomas Griffiths and Zoubin Ghahramani. Infinite latent feature models and the indian buffet process. *Gatsby Unit*, 2005.

[65] S. Guadarrama, L. Riano, D. Golland, D. Gouhring, Y. Jia, D. Klein, P. Abbeel, and T. Darrell. Grounding spatial relations for human-robot interaction. In *IROS*, pages 1640–1647, 2013.

[66] Abhinav Gupta, Aniruddha Kembhavi, and Larry S Davis. Observing human-object interactions: Using spatial and functional compatibility for recognition. *IEEE PAMI*, 31(10):1775–1789, 2009.

[67] Abhinav Gupta, Praveen Srinivasan, Jianbo Shi, and Larry S Davis. Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos. In *CVPR*, 2009.

[68] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *Proc. ECCV*, 2014.

[69] John M Hammersley and Peter Clifford. Markov fields on finite graphs and lattices. 1971.

[70] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[71] Minh Hoai and Fernando De la Torre. Max-margin early event detectors. In *CVPR*, pages 2863–2870. IEEE, 2012.

[72] Minh Hoai, Zhen-Zhong Lan, and Fernando De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.

[73] Minh Hoai, Zhen-Zhong Lan, and Fernando De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.

[74] J. Hoffart, F. M Suchanek, K. Berberich, and G. Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[75] Ninghang Hu, Gwenn Englebienne, Zhongyu Lou, and Ben Kröse. Learning latent structure for activity recognition. In *ICRA*, 2014.

[76] Ninghang Hu, Zhongyu Lou, Gwenn Englebienne, and Ben Kröse. Learning to recognize human activities from soft labeled data. *RSS*, 2014.

[77] Michael C Hughes and Erik B Sudderth. Nonparametric discovery of activity patterns from video collections. In *CVPR Workshop*, pages 25–32, 2012.

[78] A. Jain, B. Wojcik, T. Joachims, and A. Saxena. Learning trajectory preferences for manipulators via iterative improvement. In *NIPS*, 2013.

[79] M. Jain, H. Jegou, and P. Bouthemy. Better exploiting motion for better action recognition. In *CVPR*, 2013.

[80] Mihir Jain, Jan van Gemert, and Cees GM Snoek. University of amsterdam at thumos challenge 2014. *ECCV International Workshop and Competition on Action Recognition with a Large Number of Classes*, 2014.

[81] Y. Jiang, H. Koppula, and A. Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *CVPR*, 2013.

[82] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. `http://crcv.ucf.edu/THUMOS14/`, 2014.

[83] Yun Jiang, Hema Koppula, and Ashutosh Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *CVPR*, 2013.

[84] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. In *ICML*, 2012.

[85] Yun Jiang, Marcus Lim, and Ashutosh Saxena. Learning object arrangements in 3d scenes using human context. In *ICML*, 2012.

[86] Yun Jiang and Ashutosh Saxena. Modeling high-dimensional humans for activity anticipation using gaussian process latent crfs. In *RSS*, 2014.

[87] A. Karpathy and L. Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions. *ArXiv e-prints*, 2014.

[88] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[89] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. *Autonomous Robots*, 37(4), 2014.

[90] Sameh Khamis and Christoph Lampert. Coconut: Co-classification with output space regularization. In *BMVC*, 2014.

[91] Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. Large-scale video summarization using web-image priors. In *CVPR*, 2013.

[92] Gunhee Kim, Leonid Sigal, and Eric P Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *CVPR*, 2014.

[93] Gunhee Kim and Eric P Xing. Reconstructing storyline graphs for image recommendation from web community photos. In *CVPR*, 2014.

[94] Ryan Kiros, Ruslan Salakhutdinov, and Rich Zemel. Multimodal neural language models. In *ICML*, 2014.

[95] K. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *Proc. ECCV*, 2012.

[96] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *Proc. ECCV*, pages 201–214. Springer, 2012.

[97] R. A. Knepper, T. Layton, J. Romanishin, and D. Rus. Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In *ICRA*, 2013.

[98] Chen Kong, Dahua Lin, Mohit Bansal, Raquel Urtasun, and Sanja Fidler. What are you talking about? text-to-image coreference. In *CVPR*, 2014.

[99] Hema Koppula, Ashesh Jain, and Ashutosh Saxena. Anticipatory planning for humanrobot teams. ISER, 2014.

[100] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *RSS*, 2013.

[101] Hema S Koppula and Ashutosh Saxena. Physically grounded spatio-temporal object affordances. In *Computer Vision–ECCV 2014*, pages 831–847. Springer International Publishing, 2014.

[102] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 32(8):951–970, 2013.

[103] H.S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.

[104] H.S. Koppula and A. Saxena. Physically grounded spatio-temporal object affordances. In *Proc. ECCV*, 2013.

[105] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[106] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *RSS*, 2012.

[107] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *RSS*, 2012.

[108] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, 2011.

[109] K. Lai, L. Bo, X. Ren, and D. Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *ICRA*, 2011.

[110] Tian Lan, Lei Chen, Zhiwei Deng, Guang-Tong Zhou, and Greg Mori. Learning action primitives for multi-level video event understanding. In *Workshop on Visual Surveillance and Re-Identification*, 2014.

[111] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *Proc. ECCV*, pages 689–704. Springer, 2014.

[112] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014.

[113] Ivan Laptev, Marcin Marszalek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[114] Ivan Laptev and Patrick Pérez. Retrieving actions in movies. In *ICCV*, 2007.

[115] P. A. Lasota, G. F. Rossano, and J. A. Shah. Toward safe close-proximity human-robot interaction with standard industrial robots. In *CASE*, 2014.

[116] Eugene L Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science*, 18(7):401–405, 1972.

[117] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[118] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[119] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, 2012.

[120] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-segments for video object segmentation. In *ICCV*, 2011.

[121] D. B Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.

[122] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. In *RSS*, 2013.

[123] T Warren Liao. Clustering of time series dataâĂŤa survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[124] Benson Limketkai, Dieter Fox, and Lin Liao. Crf-filters: Discriminative particle filters for sequential state estimation. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 3142–3147. IEEE, 2007.

[125] Mingsheng Long and Jianmin Wang. Learning transferable features with deep adaptation networks. *arXiv preprint arXiv:1502.02791*, 2015.

[126] Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. In *CVPR*, 2013.

[127] J. Mainprice and D. Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *IROS*, 2013.

[128] Jim Mainprice and Dmitry Berenson. Human-robot collaborative manipulation planning using early prediction of human motion. In *Intelligent Robots and Systems (IROS)*, pages 299–306. IEEE, 2013.

[129] J. Malmaud, J. Huang, V. Rathod, N. Johnston, A. Rabinovich, and K. Murphy. What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision. *ArXiv e-prints*, 2015.

[130] Jon Malmaud, Earl J Wagner, Nancy Chang, and Kevin Murphy. Cooking with semantics. *ACL*, 2014.

[131] Pyry Matikainen, Rahul Sukthankar, and Martial Hebert. Model recommendation for action recognition. In *CVPR*, 2012.

[132] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In *Proc. of the 13th International Symposium on Experimental Robotics (ISER)*, June 2012.

[133] C. McManus, B. Upcroft, and P. Newman. Scene signatures: Localised and point-less features for localisation. In *RSS*, 2014.

[134] Franziska Meier, Amir Globerson, and Fei Sha. The more the merrier: Pa-

rameter learning for graphical models with multiple maps. In *ICML Workshop on Interactions between Inference and Learning*, 2013.

[135] Dipendra K Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. Environment-driven lexicon induction for high-level instructions. *ACL*, 2015.

[136] D.K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *RSS*, 2014.

[137] A. R. Mohamed, T. N. Sainath, G. Dahl, B. Ramabhadran, G. E. Hinton, and M. A. Picheny. Deep belief networks using discriminative features for phone recognition. In *(ICASSP)*, pages 5060–5063, 2011.

[138] Tanvi S Motwani and Raymond J Mooney. Improving video activity recognition using object recognition and text mining. In *ECAI*, 2012.

[139] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Robust visual robot localization across seasons using network flows. In *AAAI*, 2014.

[140] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS W*, volume 2011, page 5. Granada, Spain, 2011.

[141] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.

[142] S. Nikolaidis, P. Lasota, G. Rossano, C. Martinez, T. Fuhlbrigge, and J. Shah. Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action. In *Intl. Sym. on Robotics*, 2013.

[143] S. Nikolaidis, P. A. Lasota, G. F. Rossano, C. Martinez, T. A. Fuhlbrigge, and J. A. Shah. Human-robot collaboration in manufacturing: Quantitative evaluation of predictable, convergent joint action. In *ISR*, 2013.

[144] Edwin Olson, Matthew Walter, Seth J Teller, and John J Leonard. Single-cluster spectral graph partitioning for robotics applications. In *RSS*, 2005.

[145] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. The lear submission at thumos 2014. *ECCV International Workshop and Competition on Action Recognition with a Large Number of Classes*, 2014.

[146] Vicente Ordonez, Girish Kulkarni, and Tamara L Berg. Im2text: Describing images using 1 million captioned photographs. In *NIPS*, 2011.

[147] Pietro Perona and William Freeman. A factorization approach to grouping. In *ECCV*. 1998.

[148] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *CVPR*, 2014.

[149] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *ECCV*. 2014.

[150] Vittal Premachandran, Daniel Tarlow, and Dhruv Batra. Empirical minimum bayes risk prediction: How to extract an extra few% performance from vision models with just three more parameters. *CVPR*, 2014.

[151] Ariadna Quattoni, Sybor Wang, Louis-Phillipe Morency, Michael Collins, Trevor Darrell, and Mit Csail. Hidden-state conditional random fields. *IEEE PAMI*, 29(10):1848–1852, 2007.

[152] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[153] Lawrence Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[154] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286, 1989.

[155] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM, 2007.

[156] N. Ratliff, J. A. Bagnell, and M. Zinkevich. Maximum margin planning. In *ICML*, 2006.

[157] Xiaofeng Ren, Liefeng Bo, and Dieter Fox. Rgb-(d) scene labeling: Features and algorithms. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2759–2766. IEEE, 2012.

[158] J. Rintanen. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193:45–86, 2012.

[159] Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. A database for fine grained activity detection of cooking activities. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1194–1201. IEEE, 2012.

[160] Yong Rui, Anoop Gupta, and Alex Acero. Automatically extracting highlights for tv baseball programs. In *ACM MM*, 2000.

[161] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3), 2008.

[162] M. Ryoo and J. Aggarwal. Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. In *ICCV*, 2009.

[163] MS Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, pages 1036–1043. IEEE, 2011.

[164] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *ECCV*, pages 213–226. Springer, 2010.

[165] Simo Särkkä. *Bayesian filtering and smoothing*, volume 3. Cambridge University Press, 2013.

[166] Ashutosh Saxena, Ashesh Jain, Ozan Sener, Aditya Jami, Dipendra K Misra, and Hema S Koppula. Robo brain: Large-scale knowledge engine for robots. *International Symposium on Robotics Research (ISRR)*, 2015.

[167] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *RSS*, 2013.

[168] O. Sener, H. O. Song, A. Saxena, and S. Savarese. Unsupervised Transductive Domain Adaptation. *ArXiv e-prints*, 2016.

[169] O. Sener, A. R. Zamir, Wu. C., S. Savarese, and A. Saxena. Unsupervised Semantic Action Discovery from Video Collections . *ArXiv e-prints*, 2016.

[170] Ozan Sener and Ashutosh Saxena. rcrf: Recursive belief estimation over crfs in rgb-d activity videos. In *Robotics Science and Systems (RSS)*, 2015.

[171] Ozan Sener, Amir Zamir, Silvio Savarese, and Ashutosh Saxena. Unsupervised semantic parsing of video collections. In *International Conference on Computer Vision (ICCV)*, 2015.

[172] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001.

[173] J.M. Shepard, M.C. Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *ICRA*, 2010.

[174] Qinfeng Shi, Li Cheng, Li Wang, and Alex Smola. Human action segmentation and recognition using discriminative semi-markov models. *IJCV*, 93(1):22–32, 2011.

[175] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[176] Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. Beyond the point cloud: from transductive to semi-supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 824–831. ACM, 2005.

[177] Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Conditional models for contextual human motion recognition. *CVIU*, 104(2):210–220, 2006.

[178] Richard Socher and Li Fei-Fei. Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora. In *CVPR*, pages 966–973, 2010.

[179] Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. Grounded compositional semantics for finding and describing images with sentences. *TACL*, 2:207–218, 2014.

[180] K. Soomro, A. Roshan Zamir, and M. Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. In *CRCV-TR-12-01*, 2012.

[181] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, pages 697–706, 2007.

[182] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *AAAI*, 2016.

[183] Baochen Sun and Kate Saenko. Subspace distributed alignment for unsupervised domain adaptation. In *BMVC*, 2015.

[184] Chen Sun and Ram Nevatia. Discover: Discovering important segments for classification of video events and recounting. In *CVPR*, 2014.

[185] Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *JMLR*, 8:693–723, 2007.

[186] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[187] S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. Asking for help using inverse semantics. In *RSS*, 2014.

[188] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *AAAI*, 2011.

[189] Moritz Tenorth, Daniel Nyga, and Michael Beetz. Understanding and executing instructions for everyday manipulation tasks from the world wide web. In *ICRA*, 2010.

[190] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[191] Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

[192] Tatiana Tommasi and Barbara Caputo. Frustratingly easy NBNN domain adaptation. In *ICCV*, 2013.

[193] Ba Tu Truong and Svetha Venkatesh. Video abstraction: A systematic review and classification. *ACM TOMM*, 3(1):3, 2007.

[194] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, page 104. ACM, 2004.

[195] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[196] Laurens van der maaten. Accelerating t-sne using tree-based algorithms. In *JMLR*, 2014.

[197] M. Waibel, M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Häussermann, J. Montiel, A. Perzylo, B. Schießle, A. Zweigle, and R. van de Molengraft. Roboearth: A world wide web for robots. *IEEE Robotics & Automation Magazine*, 2011.

[198] Yang Wang and Qiang Ji. A dynamic conditional random field model for object segmentation in image sequences. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 264–270. IEEE, 2005.

[199] Zhikun Wang, Katharina Mülling, Marc Peter Deisenroth, Heni Ben Amor, David Vogt, Bernhard Schölkopf, and Jan Peters. Probabilistic movement modeling for intention inference in human–robot interaction. *The International Journal of Robotics Research*, 32(7):841–858, 2013.

[200] Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. In *NIPS*, 2006.

[201] C. Wu, I. Lenz, and A. Saxena. Hierarchical semantic labeling for task-relevant rgb-d perception. In *RSS*, 2014.

[202] C. Wu, J. Zhang, O. Sener, B. Selman, S. Savarese, and A. Saxena. Watch-n-Patch: Unsupervised Learning of Actions and Relations. *ArXiv e-prints*, 2016.

[203] Payman Yadollahpour, Dhruv Batra, and Gregory Shakhnarovich. Discriminative re-ranking of diverse segmentations. *CVPR*, 2013.

[204] Bangpeng Yao and Li Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, pages 17–24. IEEE, 2010.

[205] Bangpeng Yao and Li Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[206] Haonan Yu and Jeffrey Mark Siskind. Grounded language learning from video described with sentences. In *ACL*, 2013.

[207] Xiaojin Zhu. Semi-supervised learning literature survey. 2005.

[208] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer, 2002.

[209] Xiaojin Zhu, Zoubin Ghahramani, John Lafferty, et al. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919, 2003.

[210] Y. Zhu, A. Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *Proc. ECCV*, 2014.

[211] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, 2008.

[212] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, James A Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *IROS*, pages 3931–3936. IEEE, 2009.

[213] C Lawrence Zitnick and Devi Parikh. Bringing semantics into focus using visual abstraction. In *CVPR*, 2013.

[214] C Lawrence Zitnick, Devi Parikh, and Lucy Vanderwende. Learning the visual interpretation of sentences. In *CVPR*, 2013.

[215] M. Zucker, N. D. Ratliff, A. D. Dragan, M. Pivtoraiko, M. K., C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. CHOMP: covariant hamiltonian optimization for motion planning. *IJRR*, 32(9-10), 2013.