ABSTRACT

Title of dissertation: MANIPULATION ACTION UNDERSTANDING FOR OBSERVATION AND EXECUTION

Yezhou Yang, Doctor of Philosophy, 2015

Dissertation directed by: Professor Yiannis Aloimonos
Department of Computer Science

Modern intelligent agents will need to learn the actions that humans perform. They will need to recognize these actions when they see them and they will need to perform these actions themselves. We want to propose a cognitive system that interprets human manipulation actions from perceptual information (image and depth data) and consists of perceptual modules and reasoning modules that are in interaction with each other. The contributions of this work are given along two core problems at the heart of action understanding: a.) the grounding of relevant information about actions in perception (the perception - action integration problem), and b.) the organization of perceptual and high-level symbolic information for interpreting the actions (the sequencing problem). At the high level, actions are represented with the Manipulation Action Context-free Grammar (MACFG) , a syntactic grammar and associated parsing algorithms, which organizes actions as a sequence of sub-events. Each sub-event is described by the hand (as well as grasp type), movements (actions) and the objects and tools involved, and the relevant information about these quantities is obtained from biological-inspired perception modules.

These modules track the hands and objects and recognize the hand grasp, actions, segmentation, and action consequences. Furthermore, a probabilistic semantic parsing framework based on CCG (Combinatory Categorial Grammar) theory is adopted to model the semantic meaning of human manipulation actions.

Additionally, the lesson from the findings on mirror neurons is that the two processes of interpreting visually observed action and generating actions, should share the same underlying cognitive process. Recent studies have shown that grammatical structures underlie the representation of manipulation actions, which are used both to understand and to execute these actions. Analogically, understanding manipulation actions is like understanding language, while executing them is like generating language. Experiments on two tasks, 1) a robot observing people performing manipulation actions, and 2) a robot then executing manipulation actions accordingly, are presented to validate the formalism. The technical parts of this thesis are devoted to the experimental setting of task (1), while the task (2) is given as a live demonstration.

MANIPULATION ACTION UNDERSTANDING FOR OBSERVATION AND EXECUTION

by

Yezhou Yang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2015

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Dr. Cornelia Fermüller, Co-Advisor
Professor John Baras
Professor Hal Daumé III
Professor Don Perlis

© Copyright by
Yezhou Yang
2015

Preface

John McCarthy, who started the term Artificial Intelligence back in 1955 [1], defines it as the "science and engineering of making intelligent machines", in which an intelligent agent is a system that perceives its environment and takes actions, such as manipulating objects, that maximize its chances of success. After 60 years of advancement, we are now in the year of 2015, and the AI research is becoming highly technical and specialized. Deeply divided subfields emerge that it seems almost impossible to find connections from each other. The study of AI loses its unity and many in the field get hold of one or some aspects of the original pursuit to enjoy for themselves. The case is like the senses of hearing, sight, smell, and taste, which have specific functions, but cannot be interchanged. The school of computational perception, including computer vision and speech recognition, focuses on the aspect "...perceives its environment...". The school of symbolic AI research focuses on symbolic reasoning to "maximize its chances of success", the school of statistical learning or machine learning, focuses on "maximize its chances of success" by methods based on probability and mathematical optimization. The school of robotics, on the other hand, focuses on creating or building "an intelligent agent", either physical or virtual, that "takes actions, such as manipulating objects". The school of cognitive systems, including some parts of humanoid and human-machine interaction research, takes a view beyond not only "making intelligent machines", but also studying human beings through the methodology of reverse engineering. Even within a specific school of method, such as deep learning, researchers are usually divided into subgroups. Some care more about the empirical results on

specific applications, or the performance without, while some care more about the method's innate relation to the biological and physical systems, or the principle within. Such severe division and the long tradition of conducting vertical research make it extremely difficult to have a whole picture of AI and pursuit the ultimate goal: principle within and performance without.

I started my journey in the land of AI by creating robots playing soccer (RoboCup), a specialized system aiming at a specific application when I was an undergraduate student. Later I was fortunate enough to start my research in Computer Vision and Computational Linguistics at University of Maryland Computer Vision Lab by conducting research on the topic of combining vision and language. This initiates my horizontal viewpoint of AI. With the involvement of the European Union cognitive system project, some collaborations with symbolic AI researchers and the deployment of humanoid robots (Baxter robot) in the lab, I set my mind to conduct a horizontal PhD thesis instead of a usual vertical one. At the beginning it seemed overwhelmingly difficult, after I set the focus on human manipulation actions, it started to become feasible. In this thesis, we present researches that are conducted in the fields of Computer Vision, Computational Linguistics, Robotics and even Common-sense Reasoning, that all surround the central theme: from understanding to executing manipulation actions for intelligent agents.

I want to say that the horizontal study presented in this thesis is by no means close to the ultimate unity of AI. It might advance itself through a continuing practice, and this is, hopefully, the future work of my research career.

Dedication

To my beloved wife and parents.

Acknowledgments

I would like to thank: Dr. Yiannis Aloimonos, for his continuous guidance and support; Dr. Cornelia Fermüller, for calibrating my study with extreme patience; Dr. John Baras, Dr. Hal Daumé III, Dr. Don Perlis and Dr. Chitta Baral, for their time and expertise to improve my work; Dr. Yi Li, Dr. Ching Teo, Dr. Xiaodong Yu and Dr. Douglas Summers-Stay, for numerous discussions and debates; The robotic visual learner team: Konstantinos Zampogiannis, Yi Zhang, Yuchen Zhou and Michael Stevens, for not abandoning me because of my moodiness; My fellow peers: Aleksandrs Ecins, Austin Myers, Anupam Guha, Ren Mao, Fang Wang, Somak Aditya and others, for the joy of working together;

Telluride workshop folks, Dr. Francisco Barranco, Dr. Michael Pfeiffer, Dr. Ryad Benosman, Dr. Andreas Andreou, Dr. Tobi Delbruck and many others, for rocking me out of the comfort zone in my subfield; Poeticon++ project collaborators, Dr. Katerina Pastra, Dr. Giulio Sandini, Dr. Luciano Fadiga and Dr. Vadim Tikhanoff and many others, for passing on their passion towards cognitive robots; Qualcomm Innovation Fellowship mentors: Dr. Ashwin Sampath and Snehesh Shrestha, for giving me a chance to glance the commercial high-tech world; SPAR Workshop team, Dr. Eren Aksoy, Dr. Neil Dantam, Dr. Karinne Ramirez Amaro and Dr. Tamim Asfour, for sharing insight and inspiration;

My uncle, Dr. Jun Ye, for influencing me from almost every aspect of life; And ultimately my lifelong project partner and wife, Dr. Yang Wen, for building resilience together with me under stress and sharing happiness with me afterwards.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Problem Statement and Motivation

Intelligent agent, robots, and cognitive systems interacting with humans need to be able to interpret human actions. Here we are concerned with manipulation actions, that are actions performed by agents (humans or robots) on objects, which result in some physical change of the objects. The sensory-motor bridge connecting the two tasks is essential, and a great amount of attention in AI, Robotics as well as Neurophysiology has been devoted to understanding it. Experiments conducted on primates have discovered that certain neurons, the so-called mirror neurons, fire during both observation and execution of identical manipulation tasks ([9,10]). This suggests that the same process is involved in both the observation and execution of actions. From a functionalist point of view, such a process should be able to first build up a semantic structure from observations, reasoning over action goals, and then the decomposition of same structure should occur when the intelligent agent executes tasks. Thus in this thesis, we study the three aspects of a such cognitive system, namely 1) Perception of various aspects of manipulation actions, such as action consequences, grasp types etc; 2) Reasoning within a grammatical framework to model manipulation actions in both syntactical and semantic way; 3) Execution

of manipulation tasks on a humanoid platform (Baxter research robot). We mainly focus on the first two aspects, and provide some seminal study and results for the third aspect.

1.2 Related Work

The topic of analyzing human manipulation actions for robotic applications has been studied from multiple perspectives in the last decade. Due to its innately interdisciplinary nature, related study span over the fields of Computer Vision, Robotics, Computational Linguistics and Cognitive Systems. At visual signal processing level, [11–13] proposed to use statistical methods to model the relationship between different entities involved in manipulation actions for better visual grounding. [14, 15] proposed a semantic event chain (SEC) as middle level representation to model and learn the semantic segment-wise relationship transition from spatial-temporal video segmentation. In the field of Robotics, [16] studied the transferring of manipulation skills to robot through a semantic representation obtained from observing human activities. [17] first discussed a Chomskyan grammar for understanding complex actions as a theoretical concept, [18] provided an implementation of such a grammar using as perceptual input only objects. [19] also modeled the robot imitation learning using probabilistic activity grammars. In the field of cognitive systems, [20] proposed an integration of information from natural-language statements and the simultaneous resolution of both visual and linguistic ambiguity for robot manipulation.

1.3 Contributions of the thesis

1.3.1 The perception aspect

The input to the systems for interpreting manipulation actions is perceptual data, specifically sequences of images and depth maps. Therefore, a crucial part of our system are the vision processes, which obtain atomic symbols from perceptual data.

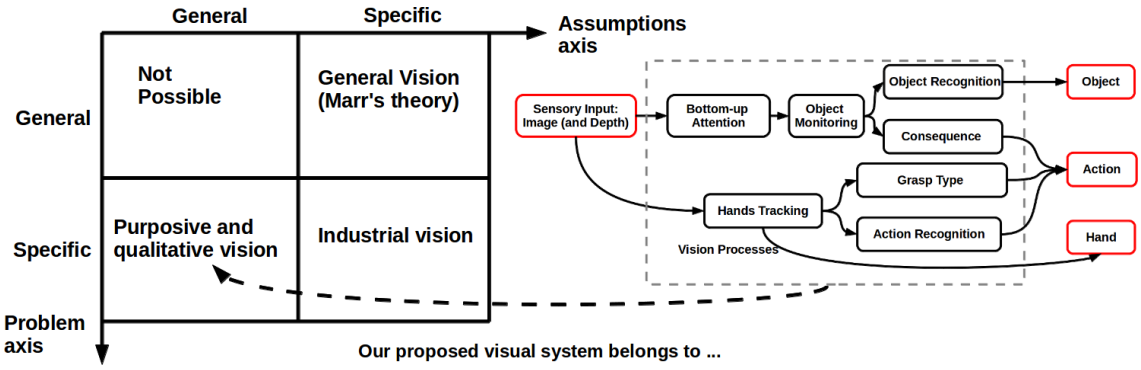


Figure 1.1: How general vision, purposive vision and industrial vision fit together, and where the manipulation action observation locates in the problem space.

The proposed formalism for the modules to visually interpret manipulation actions lies in the category of purposive vision due to the inherited nature of the problem itself. Problems such as object recognition and action recognition, which are considered as general problems in the sense their goal is a complete grounding of the perceived space. For example, a general object recognition problem asks to return all possible object areas from the dynamic scene. The object recognition

problem that we consider under manipulation action setting is specific, in the sense we only focus on the objects that is used, or under manipulation, or touched by human hands. Similarly the problem of action recognition that we consider under manipulation action setting is also a specific problem, because we only care about these actions that lead to physical change of the object, and therefore sometimes not even the actions themselves but only their consequences. However, when we make assumptions towards manipulation actions, they are general assumptions like "an object can only move or change when one or more effectors interact with it", because indeed the world satisfies it. The Figure. 1.1 from [21] shows how general vision, purposive vision and industrial vision fit together, and we show where the problem, visually interpretation of manipulation actions, locates in the problem space.

Hands are the main driving force in human manipulation actions. First of all, during the observation of manipulation actions, human beings pay attention to the hands and their surrounded area, indicating a considerable computational power is devoted to hands. Secondly, the change of grasp types of hands characterize a segmentation of the action into semantically meaningful finer components. From the viewpoint of processing perception data, the grasp type contains information about the action itself, and it can be used for prediction or as a feature for recognition. In this thesis, we investigate two different ways of tracking and analyzing hands. 1) A model based approach for controlled lab setting: a state-of-the-art markerless hand tracking system is used to obtain fine grain skeleton models of both hands. Using this data, the manner in which the human grasps the objects is classified into primitive categories; and 2) A feature based approach for unconstrained input

setting: a convolutional neural network framework is used to recognize a patch of area around each hand into grasp types. Further we show that grasp type can also be used to infer human action intention at higher level.

The second crucial part of manipulation actions is about objects. Here we denote objects as both tools and the objects under manipulation. In this thesis, we also investigate two ways to recognize objects. 1) An attention driven object monitoring process for controlled lab environment: to obtain objects, first a contour based attention mechanism locates the object. Next, the manipulated object is monitored using a process that combines stochastic tracking with active segmentation. Then, the segmented objects are recognized. Finally, with the aid of the monitoring process, the effect of the object during action is checked and classified into four types of "consequences" (which are used in the description of the action). 2) A feature based approach for unconstrained input setting: we train general object detectors from labeled training data and associate candidate object patches with the left or right hand, respectively depending on which has the smaller Euclidean distance.

The third part of manipulation action is the action itself. Unlike general action recognition problem, we are considering these actions that are performed by hands, segmented with grasp type evolution and characterized by the tool and object under manipulation. In this thesis, again we investigate two different action recognition settings. 1) A trajectory based approach for controlled lab data, which makes use of the trajectories of both hands as well as recognized tool and objects. 2) An inference approach for unconstrained scenes that based on large natural language corpus to predict the most likely action happening given the detected subject and patient. We

further show that a natural language description can be generated using a hidden Markov modeling over the detected visual components for both static scenery and manipulation action clips.

1.3.2 The reasoning aspect

Beyond visual modules, our formalism for describing manipulation actions uses a structure similar to natural language. What do we gain from this formal description of action? This is equal to asking what one gains from a formal description of language. Chomsky's contribution to language was the formal description of language through his generative and transformational grammar [22]. This revolutionized language research, opened up new roads for its computational analysis and provided researchers with common, generative language structures and syntactic operations on which language analysis tools were built. A grammar for action would contribute to providing a common framework of the syntax and semantics of action so that basic tools for action understanding can be built. These tools would allow researchers to build on when developing action interpretation systems, without having to start development from scratch.

The vision processes produce a set of symbols: the "Subject", "Action" and "Object" triplets, which serve as input to the reasoning module. However, since perceptual events do not suffice, then how do we determine the beginning and end of action segments, and how do we combine the individual segments into longer segments corresponding to a manipulation action? An essential component in the

description of manipulations is the underlying goal. The goal of a manipulation action is the physical change inducing on the object. To accomplish it the hands need to perform a sequence of sub-actions on the object. A sub-action is referring to a single movement that hand grasps or releases the object, or the hand changes grasp type during a movement. Centered around this idea, we developed a grammatical formalism for parsing and interpreting manipulation action sequences, and investigated the vision modules to obtain from videos the symbolic information used in the grammatical structure. We studied both the syntactic and semantic modeling of manipulation actions.

The syntactic part of our reasoning module is the Manipulation Action Context-Free Grammar (MACFG). This grammar comes with a set of generative rules and a set of parsing algorithms. The parsing algorithms have two main operations: "construction" and "destruction". These algorithms dynamically parse a sequence of tree (or forest) structures made up from the symbols provided by the vision module. The sequence of semantic tree structures could then be used by the cognitive system to perform reasoning and prediction. We investigated the using of action grammar to parse both controlled lab data with depth sensing and unconstrained on-line instructional videos without depth information. Following the well-known penn-tree bank format, we created a manipulation action tree bank for a set of manipulation actions that can serve as a knowledge base for robot execution.

The semantic part of our reasoning module is a Manipulation Action Categorial Combinatory Grammar (MACCG). Here we present an approach for learning the semantic meaning of manipulation action through a probabilistic semantic parsing

framework based on CCG theory. The advantage of our approach is twofold: 1) Learning semantic representations from annotations helps an intelligent agent to enrich automatically its own knowledge about actions; 2) The logic representation of the action could be used to infer the object-wise consequence after a certain manipulation, and can also be used to plan a set of actions to reach a certain action goal. Equipped with the λ representation for manipulation actions, our system is able to reason beyond observation and deduce "hidden" action consequences.

1.3.3 The execution aspect

At the end of the thesis, we conduct experiments on a research humanoid platform (Baxter robot) to have it observe human doing manipulation actions, extract the action representations for reasoning, and then perform the task using its own set of faculties. We adopt widely used humanoid control techniques such as visual servoring and dynamic movement primitives to control the humanoid. We will briefly introduce our implementation and show the performance of the robot.

1.4 Outline of the Thesis: The Road Map

The rest of the thesis starts with a study (Chapter 2) on grasp type recognition for manipulation actions [23]. It is then followed by a study (Chapter 3) on language-guided manipulation action recognition and scene understanding [24, 25]. After action recognition, the study (Chapter 4) continues with checking and monitoring action consequences [26]. In Chapter 5 and Chapter 6, we report the learning of

syntactical [27] and semantics [28] of manipulation actions respectively. At the end, in Chapter 7, we present a system of leaning procedural manipulation knowledge from on-line instructional videos for robot execution [29]. Fig. 1.2 depicts the road map.

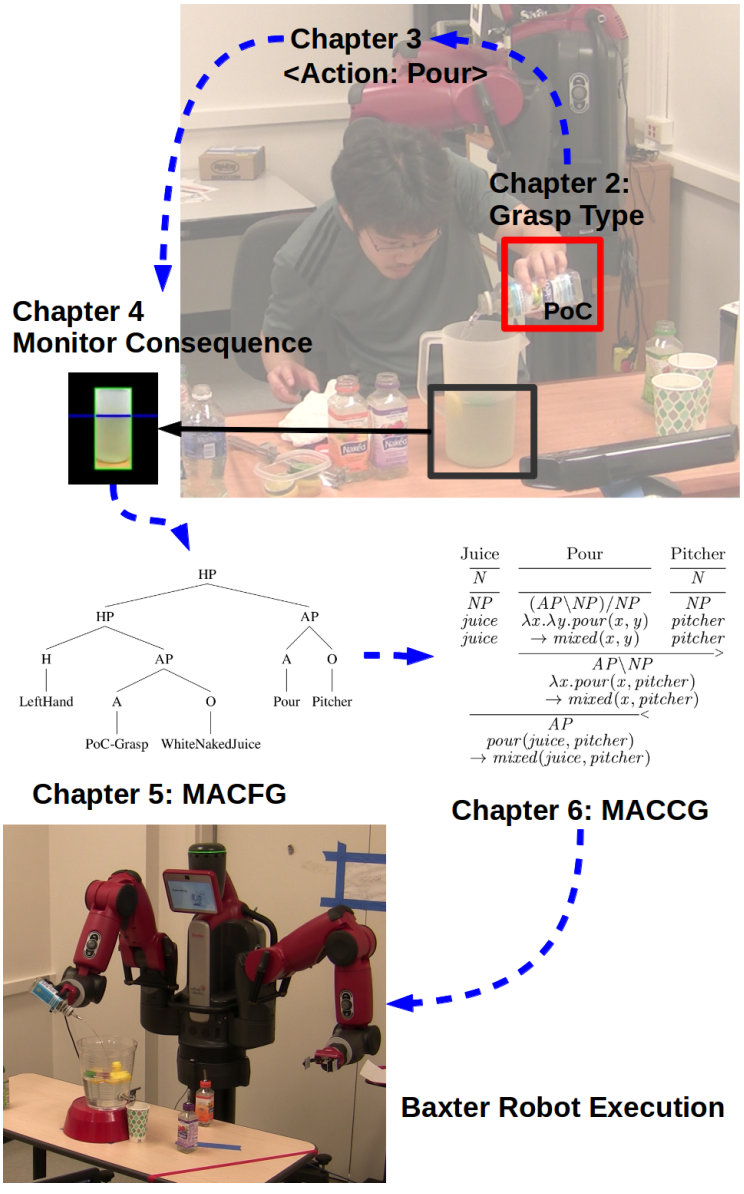


Figure 1.2: The Road Map of the thesis.

Chapter 2: "Grasp it the First Time": A Modern Perspective on Grasp Type for Manipulation Actions

2.1 Introduction

The grasp type contains fine-grain information about human action. Consider the two scenes in Fig. 2.1 from the VOC challenge. Current computer vision systems can easily detect that there is one bicycle and one cyclist (human being) in the image. Through human pose estimation, the system can further confirm that these two cyclists are riding the bike. But humans can tell that the cyclist on the left side literally is not "riding" the bicycle since his hands are posing in a "Rest or Extension" grasp next to the handlebar while the cyclist on the right side is racing because his hands firmly hold the handlebar with a "Power Cylindrical" grasp. In other words, the recognition of grasp type is essential for a more detailed analysis of human action, beyond the processes of current state-of-the-art vision systems.

Moreover, recognizing grasp type can help an intelligent system predict the human action intention. Consider an intelligent agent looking at the two scenes in Fig. 2.2(a) and (b). Current state-of-the-art computer vision techniques can accurately recognize many visual aspects from both of these scenes, such as the fact

(a) (b)

Figure 2.1: (a) Rest or Extension on the handlebar vs. (b) Firmly power cylindrical grasping the handlebar.

that there must be a human being standing in the outdoor garden scene, with a knife in his/her hand. However, we human beings will react dramatically different when experiencing the two different scenes, because of our ability to recognize immediately the different ways the person is handling the knife, i.e., the grasp type. We can effectively infer the possible activity the man is going to do based on his way of grasping the knife. After seeing scene Fig. 2.2(a), we could believe this man is going to cut something hard, or even might be malicious, since he is "Power Hook" grasping the knife. After seeing scene Fig. 2.2(b), we may react with a movement to acquire the knife (shown in Fig. 2.2(c)) since the man is "Precision Lumbrical" grasping the knife indicating a passing action. From this example we can see that the grasp type is a strong cue for us to infer the human action intention.

These are two examples demonstrating how important it is for us to be able to

(a) (b) (c)

Figure 2.2: (a) Power Hook Grasp a knife vs. (b) Precision Lumbrical Grasp a knife. (c) A natural reaction when seeing scene (b) is to open the hand to receive the knife.

recognize grasp types. The grasp type is an essential component in the characterization of human actions of manipulation ([30]). From the viewpoint of processing videos, the grasp contains information about the action itself, and it can be used for prediction or as a feature for recognition. It also contains information about the beginning and end of action segments, and thus it can be used to segment videos in time. If we are to perform the action with an intelligent agent, such as a humanoid robot, grasp is one crucial primitive action ([31]). Knowledge about how to grasp the object is necessary so the robot can arrange its effectors. For example, consider a humanoid with one parallel gripper and one vacuum gripper. When a power grasp is desired, the robot should select the vacuum gripper for a stable grasp, but when a precision grasp is desired, the parallel gripper is a better choice. Thus, knowing the grasp type provides information to plan the configuration of robot's effectors, or even the type of effector to use ([29]).

Here we present a study centered around human grasp type recognition and its applications in computer vision. The goal of this research is to provide intelligent systems with the capability to recognize the human grasp type in unconstrained

static or dynamic scenes. To be specific, our system takes in an unconstrained image patch around the human hand, and outputs which category of grasp type is used (examples are shown in Fig. 2.3). In the rest of the chapter, we show that this capability 1) is very useful for predicting human action intention and 2) helps to further understand human action by introducing a finer layer of granularity. Further experiments on two publicly available dataset empirically support that we can 1) infer human action intention in static scenes and 2) segment videos of human manipulation actions into finer segments based on the grasp type evolution. Additionally, we provide a labeled grasp type image data set and a human intention data set for further research.

Figure 2.3: Sample outputs. PoC: Power Cylindrical; PoS: Power Spherical; PoH: Power Hook; PrP: Precision Pinch; PrT: Precision Tripod; PrL: Precision Lumbrical; RoE: Rest or Extension

2.2 Related Work

Human hand related: One way to recognize grasp type is through model based hand detection and tracking [32]. Based on the estimated articulated hand model, a set of biologically plausible features such as the arches formed by fingers [33] were used to infer the grasp type involved [30]. These approaches normally use RGB Depth data and require a calibration phase, which is not applicable or is too fragile for real world situations. Also a lot of research has been devoted to hand pose or gesture recognition with promising experimental results [34, 35]. The goal of these works is to recognize poses such as "POINT", "STOP" or "YES" and "NO", not considering the interaction with objects. When it comes to recognizing grasp type from unconstrained visual input, inevitably our system has to deal with the additional challenges introduced by the interaction with unknown objects. Later in the chapter we will show that the large variation in the scenery will not allow traditional feature extraction and learning mechanism to work robustly on public available hand patch testing beds.

The robotics community has been studying perception and control problems of grasping for decades [36]. Recently, several learning based systems were reported that infer contact points or how to grasp an object from its appearance [37, 38]. However, the desired grasping type could be different for the same target object, when used for different action goals. The acquisition of grasp information from natural static or dynamic scenes is still considered very difficult because of the large variation in appearance and the occlusions of the hand from objects during

manipulation.

Vision beyond appearance: The very small number of works in computer vision, which aim to reason beyond appearance models, are also related to this chapter. [39] proposed that beyond state-of-the-art computer vision techniques, we could possibly infer implicit information (such as functional objects) from video, and they call them "Dark Matter" and "Dark Energy". [26] used stochastic tracking and graph-cut based segmentation to infer manipulation consequences beyond appearance. [40] used a ranking SVM to predict the persuasive motivation (or the intention) of the photographer who captured an image. More recently, [41] seeks to infer the motivation of the person in the image by mining knowledge stored in a large corpus using natural language processing techniques. Different from these fairly general investigations about reasoning beyond appearance, our chapter seeks to infer human action intention from a unique and specific point of view: the grasp type.

Convolutional neural networks: The recent development of deep neural networks based approaches revolutionized visual recognition research. Different from the traditional hand-crafted features [42, 43], a multi-layer neural network architecture efficiently captures sophisticated hierarchies describing the raw data [44], which has shown superior performance on standard object recognition benchmarks [45, 46] while utilizing minimal domain knowledge. The work presented in this chapter shows that with the recent developments of deep neural networks, we can learn a model to recognize grasp type from unconstrained visual inputs with robustness. We believe we are among the first to apply deep learning on grasp type recognition.

2.3 Our Approach

First, we briefly summarize the basic concepts of Convolutional Neural Networks (CNN), and then we present our implementations for grasp type recognition, human action intention prediction and fine level manipulation action segmentation using the change of grasp type over time.

2.3.1 Human Grasp Types

A number of grasping taxonomies have been proposed in several areas of research, including robotics, developmental medicine, and biomechanics, each focusing on different aspects of action. In a recent survey, Feix et al. [47] reported 45 grasp types in the literature, of which only 33 were found valid. In this work, we use a categorization into seven grasp types. First we distinguish, according to the most commonly used classification (based on functionality), into power and precision grasps [48]. Power grasping is used when the object needs to be held firmly in order to apply force, such as "grasping a knife to cut"; precision grasping is used in order to do fine grain actions that require accuracy, such as "pinch a needle". We then further distinguish among the power grasps, whether they are cylindrical, spherical, or hook. Similarly, we distinguish the precision grasps into pinch, tripodal and lumbrical. Additionally, we also consider a Rest or Extension position (no grasping performed). Fig. 2.4 illustrates the grasp categories.

Humans, when looking at a photograph, can more or less tell what kind of grasp the person in the picture is using. The question becomes, whether using the

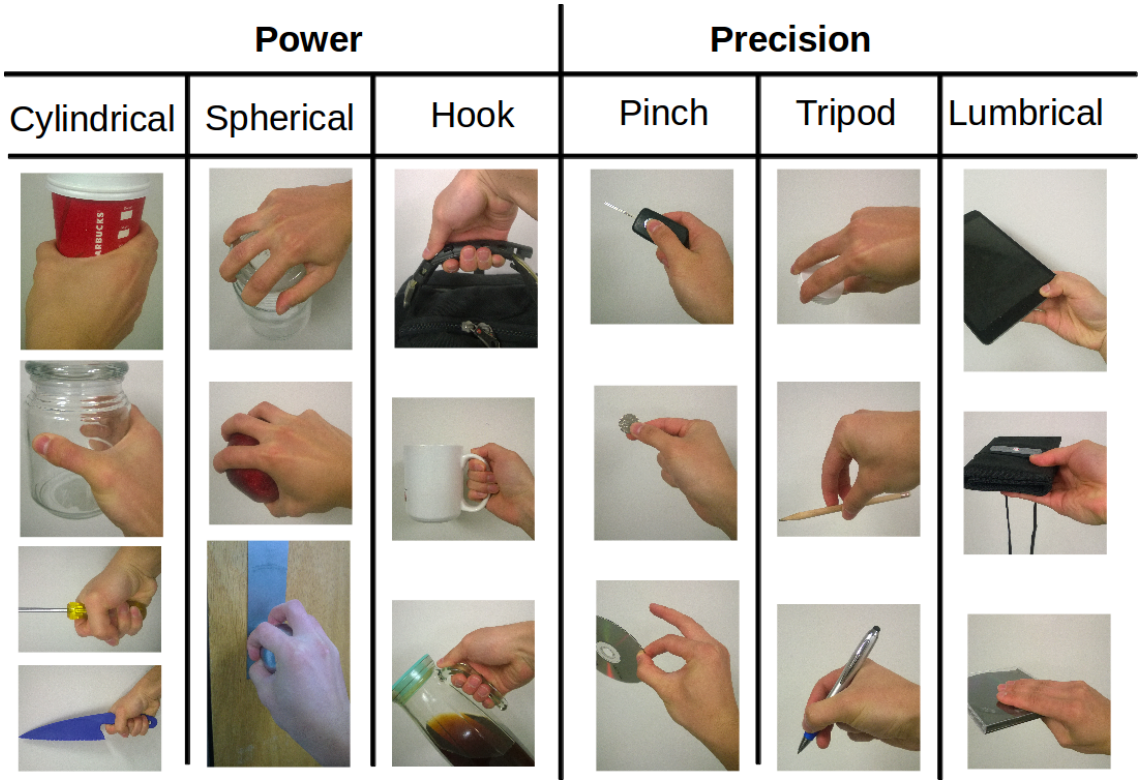


Figure 2.4: The grasp types considered. Grasps which cannot be categorized into the six types here are considered as the "Rest and Extension" (no grasping performed).

current state-of-the-art computer vision technique, whether we can develop a system that learns the pattern from human labeled data and recognizes grasp type from a patch around each hand? In the following section, we present our take and show that a grasp type recognition model with decent robustness can be learned using Convolutional Neural Network (CNN) techniques.

2.3.2 CNN for Grasp Type Recognition

Convolutional Neural Network (CNN) is a multilayer learning framework, which may consist of an input layer, a few convolutional layers and an output layer. The goal of CNN is to learn a hierarchy of feature representations. Response maps in each layer are convolved with a number of filters and further down-sampled by pooling operations. These pooling operations aggregate values in a smaller region by down-sampling functions including max, min, and average sampling. In this work we adopt the softmax loss function which is given by:

$$L(t,y) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{C} t_k^n log(\frac{e^{y_k^n}}{\sum_{m=1}^{C} e^{y_m^n}}) \tag{2.1}$$

where t_k^n is the n-th training example's k-th ground truth output, and y_k^n is the value of the k-th output layer unit in response to the n-th input training sample. N is the number of training samples, and since we consider 7 grasp type categories, $C = 7$. The learning in CNN is based on Stochastic Gradient Descent (SGD), which includes two main operations: Forward and Back Propagation. The learning rate is dynamically lowered as training progresses. Please refer to [49] for details.

We used a five layer CNN (including the input layer and one fully-connected perception layer for regression output). The first convolutional layer has 32 filters of size 5×5 with max pooling, the second convolutional layer has 32 filters of size 5×5 with average pooling, and the third convolutional layer has 64 filters of size 5×5 with average pooling, respectively. Convolutional layer convolves its input with a bank of filters, then applies point-wise non-linearity and max or average pooling

operation.

The final fully-connected perception layer has 7 regression outputs. Fully-connected perception layer applies linear filters to its input, then applies point-wise non-linearity. Our system considers 7 grasp type classes.

For testing, we pass each target hand patch to the trained CNN model, and obtain an output of size 7×1: $P_{GraspType}$. In the action intention and segmentation experiments we use the classification for both hands to obtain $P_{GraspType1}$ for the left hand, and $P_{GraspType2}$ for the right hand, respectively. To have a fully automatic fine level manipulation segmentation approach, we need to localize the input hand patches from videos and then recognize grasp types using CNN. We use the hand detection method of [2] to detect hands in the first frame, and then apply a meanshift algorithm based tracking method [3] on both hands to continuously extract the image patch around each hand.

2.3.3 Human Action Intention

Our ability to interpret other people's actions hinges crucially on predicting their intentionality. Even 18-month-old infants behave altruistically when they observe an adult accidentally dropping a marker on the floor but out of his reach, and they can predict his intention to pick up the marker [50]. From the point view of machine learning for intelligent systems and human-robot collaboration, due to the differences in the embodiment of humans and robots, a direct mapping of action signals is problematic. One solution is that the robot predicts the intent of the ob-

served human activity and implements the same intention using its own sensorimotor apparatus [51].

Previous studies showed that there are several key factors that affect the grasp type [52]. One crucial deciding factor for the selection of the grasp type to use is the intended activity. We choose here a categorization into three human action intentions, closely related to the functional classification discussed above (Fig. 2.4). The first category reflects the intention to apply force onto the physical world, such as for example "cut down a tree with an ax", and we refer to it as "Force-oriented". The second category reflects fine-grained activity where sensitivity and dexterity are needed, such as "tie shoelaces", and we refer to it as "Skill-oriented". The third category has no intention of specific action, such as "showcasing and posing", and we call it "Casual". Fig. 2.5 illustrates the action intention categories by showing one typical example of each. We should note that the three categories: "force-oriented", "skill-oriented" and "casual" are closely related to the three functional categories "power" "precision", and "rest", respectively (Fig. 2.4). We used a different labeling, because we encounter a larger variety of hand poses in the static images used for intention classification than in the videos of human manipulation activities used for functional categorization.

We investigate the causal relation between human grasp type and action intention by training a classifier using grasp types of both hands as input, and the category of action intention as output. As shown next, our experiment demonstrates a strong link. We want to point out that certainly a finer categorization is possible. For example, "Force oriented" intention can be further divided into sub classes such

Figure 2.5: Human action intention categories.

as "Selfish" or "Altruistic" and so on. However, such a classification would require other dynamic observations. Here we show that from the grasp type in a single image a classification into basic intentions (shown in Fig. 2.5) is possible.

2.3.4 From Grasp Type to Action Intention

Our hypothesis is that the grasp type is a strong indicator of human action intention. In order to validate this, we train an additional classifier layer. The procedure is as follows. For each training image, we first pass the target hand patches (left hand and right hand, if present) of the main character in the image to the trained CNN model, and we obtain two belief distributions: $P_{GraspType1}$ and $P_{GraspType2}$. We concatenate these two distributions and use them as our feature vector for training. We train a support vector machine (SVM) classifier f, which

takes as input the grasp type belief distributions and derives as output an action intention distribution P_{Int} of size 3×1:

$$P_{Int} = f(P_{GraspType1}, P_{GraspType2}|\theta), \tag{2.2}$$

where θ are the model parameters learned from labeled pairs. Fig. 2.6 shows a diagram of the approach. We need to point out that in the human action intention recognition we use belief distributions instead of final class labels of the two hands as input feature vectors. Thus, a certain category of grasp type does not directly indicate a certain action intention in our model. A further experiment using detected grasp type labels of both hands (the grasp type with the highest belief score) to infer action intention achieves a slightly worse performance, which confirms our claim here.

2.3.5 Grasp Type Evolution

In manipulation actions involving tools and objects, the details of the small sub actions contain rich semantics. Current computer vision methods do not consider them. Consider a typical kitchen action, as shown in Fig. 2.7. In most approaches the whole sequence would be denoted as "sprinkle the steak", and the whole segment would be considered an atomic part for recognition or analysis. However, within this around 15 second long action, there are several finer segments. The gentleman first "Pinch" grasps the salt to sprinkle the beef, then he "Extends" to point at the oil bottle, and later he "Power Spherical" grasps a pepper bottle to further sprinkle black pepper onto the beef. Here we can see that the dynamic changes of grasp type

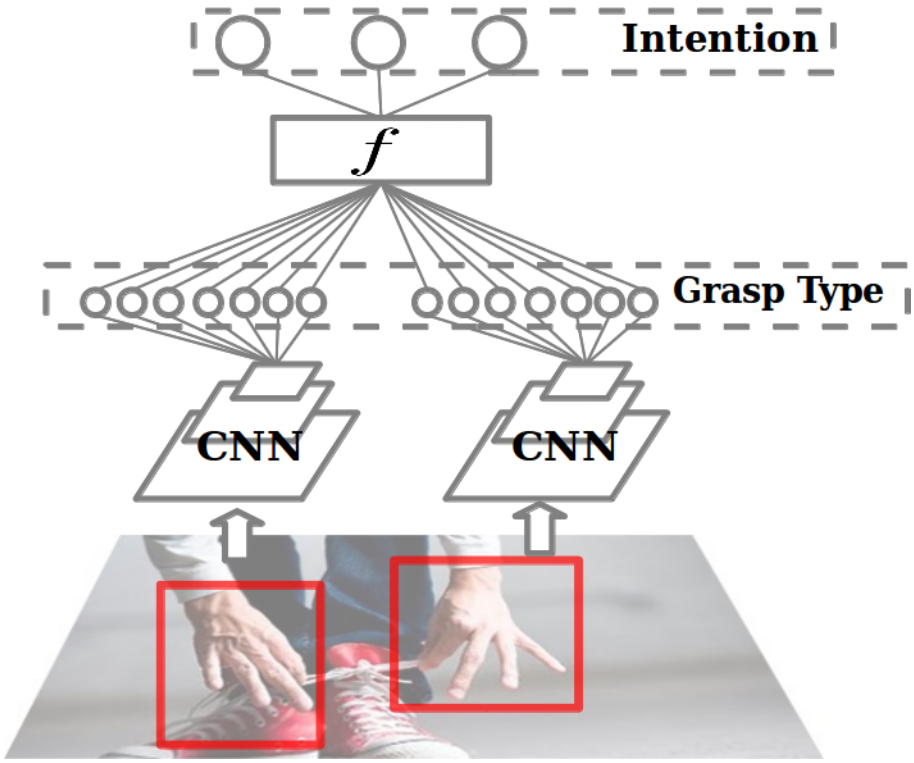


Figure 2.6: Inference of human action intention from grasp type recognition.

characterize the start and end of these finer actions.

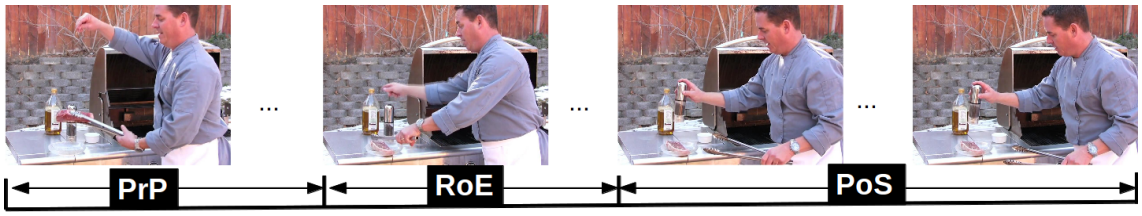


Figure 2.7: Grasp type evolution (right hand) in a manipulation action.

In order to see if grasp type evolution actually can help with a finer segmentation of manipulation actions, we first recognize the grasp type of both hands, frame by frame, and then output a segmentation at the points in time when any of the hands has a change in grasp type. We design a third experiment on a public cooking

video dataset from Youtube for validation.

2.3.6 Finer segment action using grasp type evolution

We adopt a straightforward approach. Let's denote the sets of grasp types along the time-line of an action of length M as $G_l = \{G_l^1, G_l^2 ... G_l^M\}$ for the left hand and as $G_r = \{G_r^1, G_r^2 ... G_r^M\}$ for the right hand. Assuming that during a manipulation action the grasp type evolves gradually, we first apply a one dimensional mode filter to smooth temporally. Each grasp type detection at time t is replaced by its most common neighbor in the window of $[t - \delta/2, t + \delta/2]$, where δ is the window size.

Then, whenever at a time instance $t \in [1, M]$, if $G_l^t \neq G_l^{t+1}$ or $G_r^t \neq G_r^{t+1}$, our system outputs one segment at t, denoted as S_t. The set S_t yields a finer segmentation of the manipulation action clip.

2.4 Experiments

The theoretical framework we presented suggests three hypotheses that deserve empirical tests: (a) the CNN based grasp type recognition module can robustly classify input hand patches into correct categories; (b) hand grasp type is a reliable cognitive feature to infer human action intention; (c) the evolution of hand grasp types is useful for fine-grain segmentation of human manipulation actions.

To test the three hypotheses empirically, we need to define a set of performance variables and how they relate to our predicted results. The first hypothesis relates to visual recognition, and we can empirically test it by comparing the de-

Methods	PoC		PoS		PoH		PrP		PrT		PrL		RoE		Overall
	P	R	P	R	P	R	P	R	P	R	P	R	P	R	Accu
(A)	.44	.46	0	NaN	0	NaN	0	0	0	NaN	0	NaN	.81	.41	.42
(B)	.50	.40	0	NaN	0	NaN	.03	.17	0	0	0	0	.62	.36	.36
(C)	.59	.60	.38	.62	.38	.58	.62	.60	.56	.66	.36	.40	.69	.56	.59

Table 2.1: Precision (P) and Recall (R) for each grasp type category and overall accuracy. (A):HoG+BoW+SVM; (B):HoG+BoW+RF; (C): CNN

tected grasp type labels with the ground truth ones using the precision and recall metrics. We further compare the method with a traditional hand-crafted feature based approaches to show the advantage of our approach. The second hypothesis relates to the inference of human action intention, and we can also empirically test it by comparing the predicted action intention with the ground truth ones on a testing set. The third hypothesis relates to manipulation action segmentation, and we can test it by comparing the computed key segment frames with the ground-truth ones. We used two publicly available datasets: (1) the Oxford hand dataset [2] and (2) a unconstrained cooking video dataset (YouCook) [53].

2.4.1 Grasp Type Recognition in Static Images

Dataset and Experimental protocol

The Oxford hand dataset is a comprehensive dataset of hand images collected from various different public image data set sources with a total of 13050 annotated

hand instances. Hand patches larger than a fixed area of (a bounding box of 1500 sq. pixels) were considered sufficiently 'large' and were used for evaluation. This way we obtained 4672 hand patches from the training set and 660 hand patches from the testing set (VOC images). We then further augmented the dataset with new annotations. We categorized each patch into one of the seven classes by considering its functionality given the image context and its appearance following Fig. 2.4. We followed the training and testing protocol from the dataset.

For training the grasping type, the image patches were resized to 64×64 pixels. The training set contains 4672 image patches and was labeled with the seven grasping types. We used a GPU based CNN implementation [54] to train the neural network, following the structure described above.

We compared our approach with traditional hand-crafted feature based approaches. One was the histogram of oriented gradients (HoG) + Bag of Words (BoW) + SVM classification, the other HoG + BoW + Random Forest. The number of orientations we selected for HoG was 32, and the number of dictionary entries for BoW was 100. The parameters for the baseline methods were tuned to have the best performance.

Experimental results

We achieved an average of 59% classification accuracy using the CNN based method. Table 2.1 shows the performance metrics of each grasp type category and the overall performance in comparison to baseline algorithms. It can be seen that the CNN based approach has a decent advantage. To provide a full picture of our CNN based classification model, we also show the confusion matrix in Fig. 2.8. Our

system mainly confused "Power Cylindrical Grasp" with "Rest or Extension". We believe that this is mostly because the fingers form natural curves when resting and this makes the hand look very similar to a cylindrical grasp with large diameter. Also our model does not perform well on "Precision Lumbrical" grasp due to the relatively small amount of training samples in this category. Fig. 2.9 shows some correct grasp type predictions (denoted by black boxes), and some failure examples (denoted by red and blue bounding boxes). Blue boxes denote a correct prediction of the underlying high-level grasp type in either the "Power" or "Precision" category, but incorrect recognition in finer categories. Red boxes denote a confusion between"Power" and "Precision" grasp. Intuitively, the blue marked errors should be penalized less than the red marked ones.

2.4.2 Inference of Action Intention from Grasp Type

Dataset and Experimental protocol

A subset of 200 images from the Oxford hand dataset serves as testing bed for action intention classification. Since not every image in the test set contains an action intention that falls into one of the three major categories described above, the subset was selected with the following rules: (1) at least one hand of the main character can be seen from the image and (2) the main character has a clear action intention. For example, we can infer that the character from Fig. 2.10(a) is going to perform a skill-oriented actions that requires accuracy, while this is not clear from the character in Fig. 2.10(b) (pull the rope with force or just posing casually?).

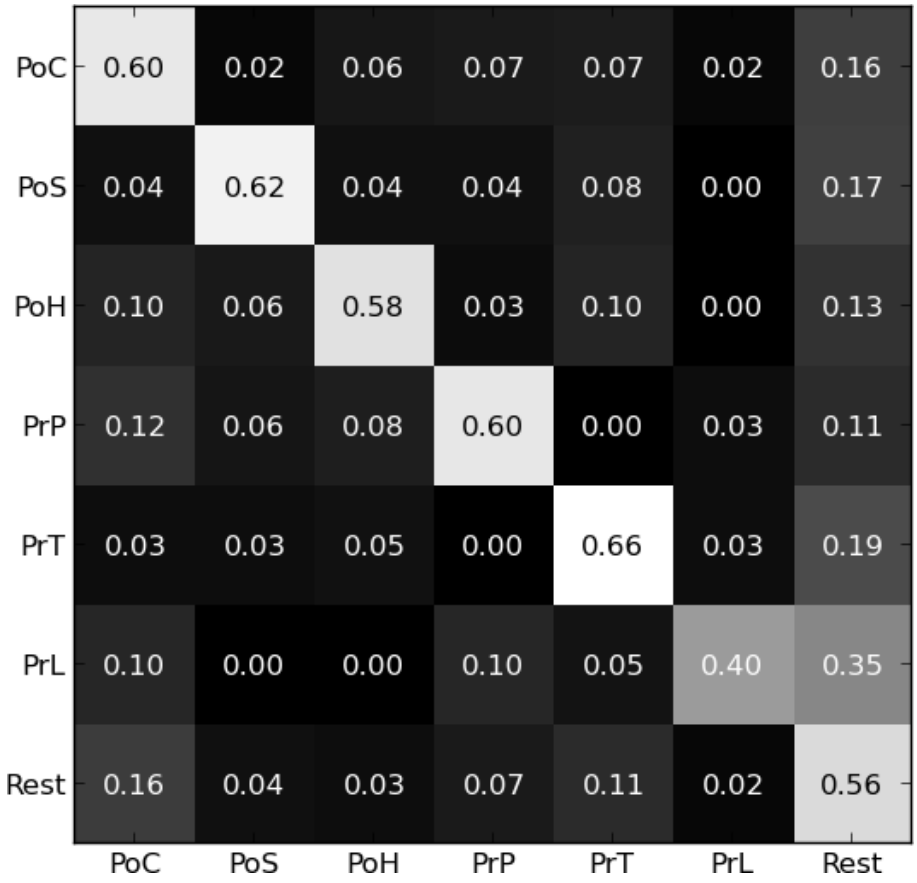


Figure 2.8: Category pairwise confusion matrix for grasp type classification.

We labeled the 200 images into the three major action intention categories and used them as ground truth. The grasp type CNN model was used to extract a 14 dimension belief distribution as grasp type feature (which is due to data from both hands of the main character). A 5 folds cross validation protocol was adopted and we trained each fold using a linear SVM classifier.

Experimental results

We achieved an average 65% prediction accuracy. Table 2.2 reports precision and recall metrics for each category of action intention. We also run the same experiment using grasp type labels instead of belief distributions (GL+SVM). We

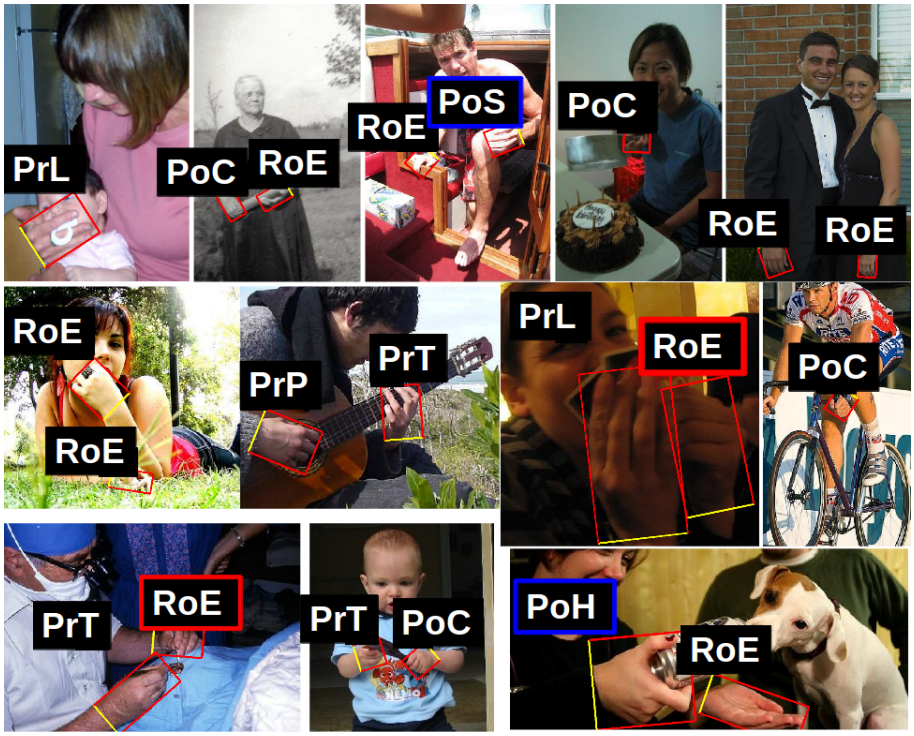


Figure 2.9: Examples of correct and false classification. PoC: Power Cylindrical; PoS: Power Spherical; PoH: Power Hook; PrP: Precision Pinch; PrT: Precision Tripod; PrL: Precision Lumbrical; RoE: Rest or Extension.

(a) (b)

Figure 2.10: Clear action intention vs. an ambigous one

can see that it achieves slightly worse performance than using belief distributions. Fig. 2.11 shows some interesting correct cases, and Fig. 2.12 shows several failure predictions. We believe that the failure cases are mostly due to the wrong grasp

Methods	F-O		S-O		C		Overall
	P	R	P	R	P	R	Accu
GL+SVM	.54	.35	.73	.59	.80	.89	.63
GT+SVM	.61	.35	.82	.71	.82	.83	.65

Table 2.2: Precision (P) and Recall (R) for each intention category and overall accuracy. GL: Grasp type Label; GT: Grasp Type belief distribution.

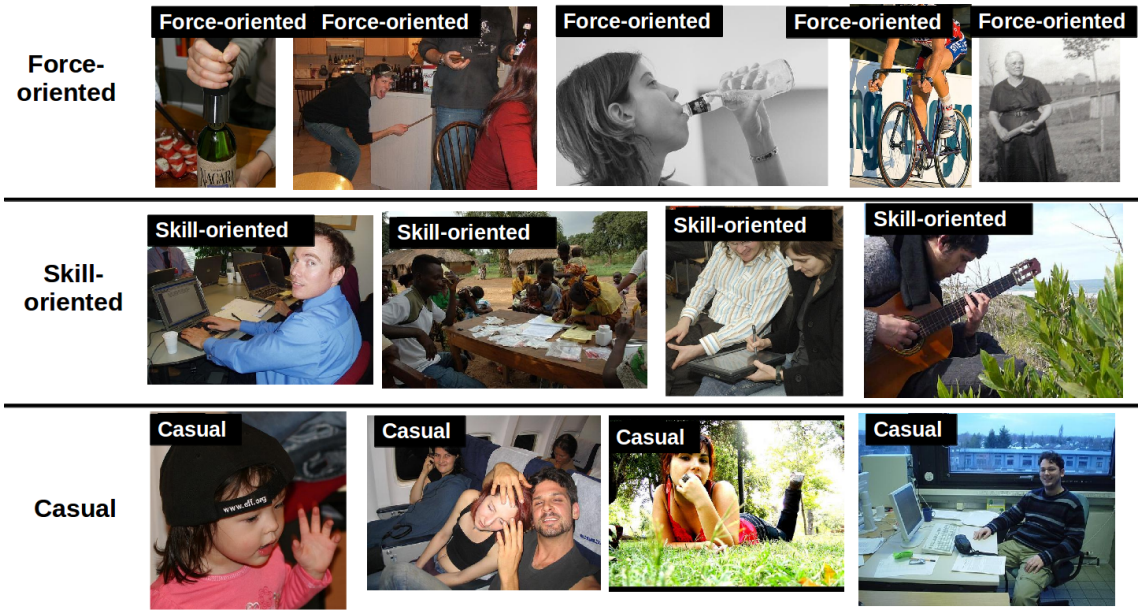


Figure 2.11: Correct examples of predicting action intention.

type recognition inherited from the previous section. Because of the small amount of pairs with ground truth, we were not able to train for comparison a converging CNN model, that would predict action intention directly from hand patches.

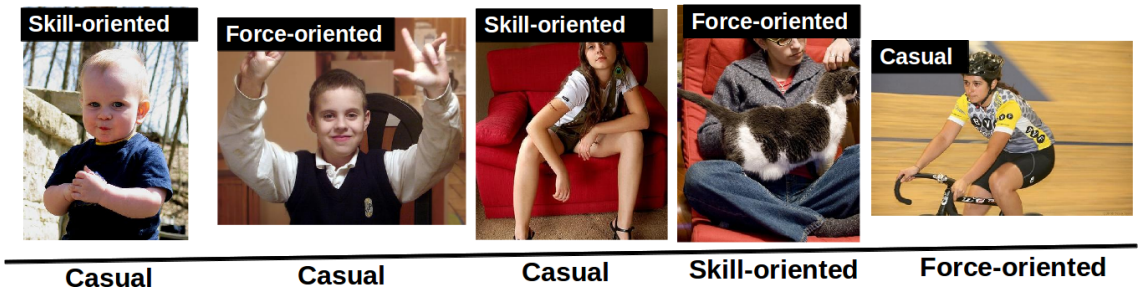


Figure 2.12: Failure cases of predicting action intention. The label at the bottom denotes the human labeling.

2.4.3 Manipulation Action Fine Level Segmentation using Grasp Type Evolution

In this section we want to demonstrate that the change of grasp type is a good feature for fine grain level manipulation action temporal segmentation.

Dataset and Experimental protocol

Cooking is an activity, requiring a variety of grasp types, that intelligent agents most likely need to learn. We conducted our experiments on a publicly available cooking video dataset collected from the world wide web and fully labeled, called the Youtube cooking dataset (YouCook) [53]. The data was prepared from open-source Youtube cooking videos with a third-person view. These features make it a good empirical testing bed for our third hypothesis.

We conducted the experiment using the following protocols: (1) 8 video clips, which contain at least two fine grain activities, were reserved for testing; (2) all other video frames were used for training; (3) we randomly reserved 10% of the training

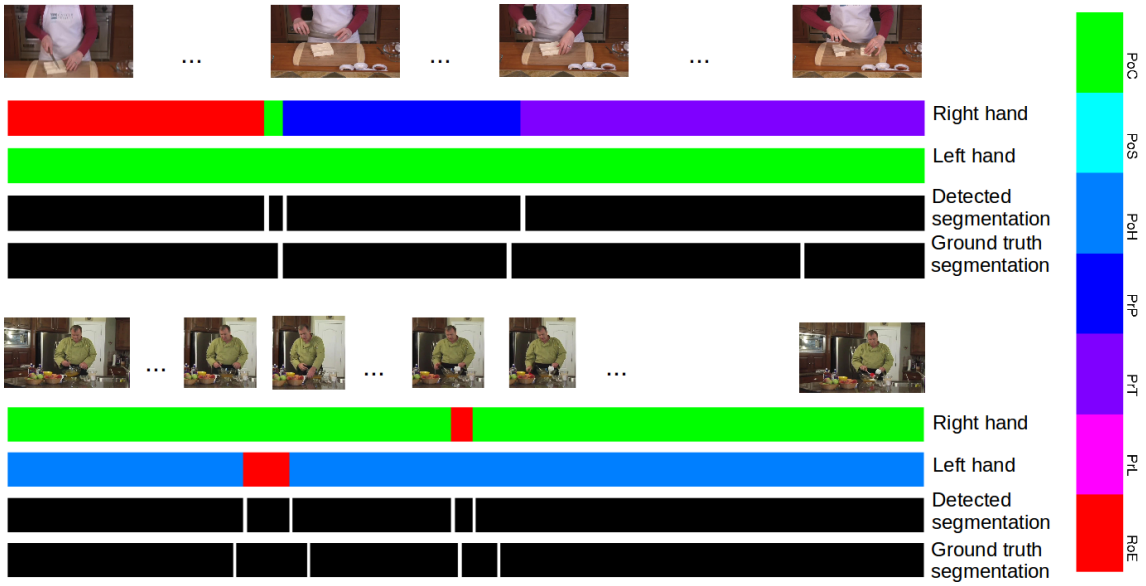


Figure 2.13: Left and right hand grasp type recognition along timeline and video segmentation results compared with ground truth segments.

data as validation set for training the CNNs. For training the grasp type recognition model, we extended the dataset by annotating image patches containing hands in the training frames. The image patches were resized to 64×64 pixels. The training set contains image patches that was labeled with the seven grasp types. We used the same CNN implementation [54] to train the neural network, following the same structures described above.

Action Fine Level Segmentation

For each testing clip, we first picked the top two hand proposals using [2] in the first frame, and then we applied a meanshift algorithm based tracking method [3] on both hands to continuously extract an image patch around each hand (Fig. 2.14). The image patches were further resized to 64×64 and pipelined to the trained CNN

Figure 2.14: 1st row: sample hand localization on first frame using [2]. 2nd to 5th row: two sample sequences of hand patches extracted using meanshift tracking [3].

model. We then labeled each hand with the grasp type of highest belief score in each frame. After applying a one dimensional mode filtering for temporal smoothing, we computed the grasp type evolution for each hand and segmented whenever one hand changes grasp type, as described in Sec. 2.3.6.

Fig. 2.13 shows two examples of intermediate grasp type recognition for the two hands and the detected segmentation. A key frame is considered correct, when a ground truth key frame lies within 10 frames around it. In the first example, the subject's right hand at the beginning holds the tofu using an Extension grasp, and then she cuts the tofu with a Pinch grasp holding the blade. Then using a precision Tripod grasp she separates one piece of tofu from the rest, and at the end using a

Lumbrical grasp she further cuts the smaller piece of tofu. Using the grasp type evolution, our system can successfully detect two key frames out of the three ground truth ones. In the second video, the gentleman using a Cylindrical grasp whisks the bowl at the beginning. Then his left hand extends to reach a small cup, and then using a Hook grasp he holds the cup. After that, his right hand extends to reach a spatula and at the end his right hand scoops food out of the small cup using a Cylindrical grasp. Using the grasp type evolution, our system can successfully detect three key frames out of the four ground truth ones.

In the 8 test clips, there are 18 ground truth segmentation key frames, and 14 of them are successfully detected, which yields a recall of 78%. Among the 20 detected segmentation key frames, 16 are correct, which yields a precision of 80%.

Chapter 3: "Get Your Act Together": Language-Guided Manipulation Action Recognition and Scene Understanding

3.1 Language-Guided Action Recognition

Humans display an uncanny ability to perceive the world that far surpasses current vision only based systems both in terms of precision and accuracy. This is largely due to the vast amount of high-level knowledge that humans have acquired over their lives that enables humans to infer and recognize complex visual inputs. In the same way, service and personal robots of the future must be endowed with such knowledge in order to interact and reason with humans and their environments effectively. This knowledge can be encoded in various forms, of which language is clearly the most predominant. Language manifests itself in the form of text which is also extremely accessible from various large research corpora.

The ability to reason is an essential faculty for service and personal robots. A typical scenario is when the robot needs to understand an action or task which the human performs for the purpose of learning. Such Learning from Demonstration (LfD) [55] paradigm is gaining popularity in robotics as shown by the recently

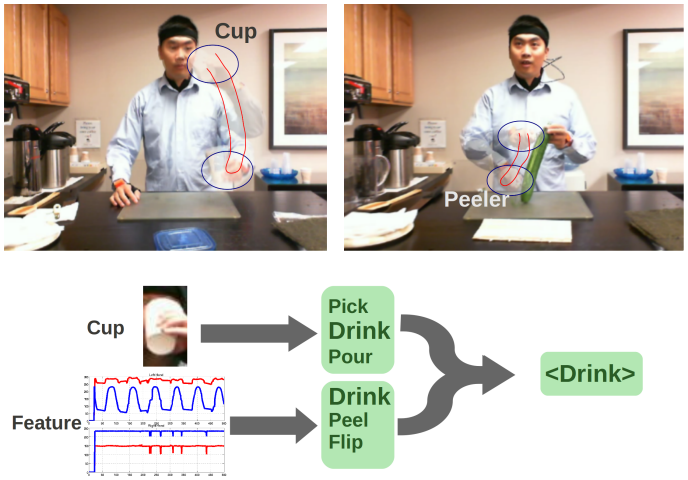


Figure 3.1: (Top) Ambiguities in action recognition: similar trajectories for different actions. Tools considered in isolation can only suggest possible actions. (Below) Language can predict, given the tool and action trajectories, the most likely action label.

concluded Learning by Demonstration Challenge at AAAI 2011[1]. There are two interconnected components in LfD. The first component recognizes what actions occurred and the second component creates an internal representation so that the action can be recreated by the robot. Unlike the LfD challenge where the robot is supposed to recreate the *one* task it was taught (the second component), we address a variant of the first component where the robot labels the correct action associated with a set of *unlabeled* action data (with repetitions) performed by different human teachers (or actors). Posing our problem in terms of unlabeled data over different actors is also closer to reality as the robot needs to learn how to *generalize* the action so that it can discover, in the second component, an optimal representation to recreate this action on its own.

[1] http://www.lfd-challenge.org/

In this chapter, we consider the task of recognizing human activities from videos sequences – specifically, activities that involve hand-tools. Action or activity recognition has remained one of the most difficult problems in computer vision. The main reason is that detections of key objects that define the action in video – tools, objects, hands and humans are still unreliable even using current state of the art object detectors [4, 56]. Descriptions based on tracking trajectories of local features such as STIP [57] and modeling velocity as suggested in [58], are strongly viewpoint dependent and may be confused when similar movements are used for different actions, e.g. drinking from a cup vs. peeling. Both actions involve large up-down hand movements. The main challenge of action recognition is the ambiguity when these two components: objects and trajectories, are considered in isolation. From Fig. 3.1(top), detecting a cup or action trajectory in isolation can only suggest some likely actions. A more reliable prediction can be achieved when we combine both object detection and trajectories.

A missing component in the above approach is how we can combine object detection with actions in a logical and intuitive way. To do this, we propose to model language as a resource of prior knowledge that essentially encodes how actions and objects (tools) are related. This language model allows us to weigh the video detections so that the objects and action features that best explain the video are eventually selected (Fig. 3.1(below)).

3.1.1 Related Work

Action recognition research spans a long history. Comprehensive reviews of recent state of the art can be found in [59–61]. Most of the focus was on studying human actions that were characterized by movement and change of posture, such as walking, running, jumping etc. Many studies represent actions by modeling body poses, body joints and body parts [62]. Depending on the extent of the features used, the literature distinguishes between local and global action models. The former use spatio-temporal interest points and descriptors based on intensity, gradients and flow within spatio-temporal cuboids centered on these interest points [57,63]. The latter compute descriptors over the whole video frame or an extracted human skeleton or silhouette. For example, [64] used histograms of optical flow and Gorelick et al. [65] and Yilmaz et al. [66] represented human activities by 3-D space-time shapes. Another class of approaches model the evolution of actions over time. For example Bissacco et al. [67] used joint-angle as well as joint trajectories from motion-capture data and features extracted from silhouettes to represent action profiles. Chaudhry et al. [68] employed non linear dynamical systems to model the temporal evolution of optical flow histograms.

Our approach is more closely related to the use of language for object detection and image annotation. With advances on textual processing and detection, several works recently focused on using sources of data readily available "in the wild" to analyze static images. The seminal work of Duygulu et al. [69] showed how nouns can provide constraints that improve image segmentation. Gupta et al. [70] (and

references herein) added prepositions to enforce spatial constraints in recognizing objects from segmented images. Berg et al. [71] processed news captions to discover names associated with faces in the images, and Jie et al. [72] extended this work to associate poses detected from images with the verbs in the captions. Some studies also considered dynamic scenes. [73] studied the aligning of screen plays and videos, [74] learned and recognized simple human movement actions in movies, and [75] studied how to automatically label videos using a compositional model based on AND-OR-graphs that was trained on the highly structured domain of baseball videos

.

These recent works had shown that exploiting co-occurring text information from scripts and captions aids in the visual labeling task. This chapter takes this further by using *generic* text obtained from the English Gigaword corpus [76], which is a large corpus of English newswire text from which we learn a language model. As we will show, using general NLP tools, we still can derive interesting relationships to guide the visual task of action recognition.

3.1.2 Our Approach

The input is a set of $|M|$ videos, $M = \{m_d\}, d \in \{1, 2, \cdots, |M|\}$ containing some actions, with each video containing exactly one unique action. The $|V|$ actions are drawn from the set $V = \{v_j\}, j \in \{1, 2, \cdots, |V|\}$. This means that we have assumed that the task of segmenting actions from a long video sequence has been done. In addition, we assume that every action must have an actor that uses a

particular hand-tool. The same tool can be used in multiple actions. The $|N|$ tools comes from the set $N = \{n_i\}, i \in \{1, 2, \cdots, |N|\}$. All labels (both actions and tools) must be known in advance, which is a requirement for learning the appropriate language model. A summary of the approach is shown in Fig. 3.2.

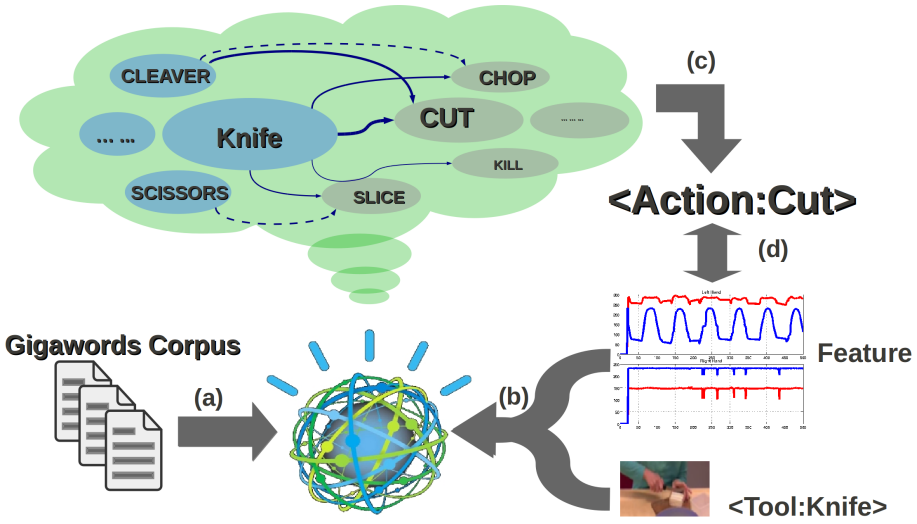


Figure 3.2: Key components of the approach.: (a) Training the language model from a large text corpus. (b) Detected tools are queried into the language model. (c) Language model returns prediction of action. (d) Action features are compared and beliefs updated.

The high level overview of the approach is as follows (see Fig. 3.2): 1) We first detect potential tools from the input video. 2) For each identified tool, we query a trained language model to determine the most likely verbs (actions) associated with the tool. 3) We then confirm the predicted action using the action features obtained from the video to update our confidence on the current action label of the video. This process is repeated until our belief on the action labels is maximized over all tools and action features. Note that our approach is *symmetric*, that is,

we could have started off with action features and inquire the language model in exactly the same way. Our choice of starting with objects is based on the fact that object detectors are better researched and are generally more accurate than action detectors.

For the purpose of this discussion, we shall assume the most general case where we only have unlabeled video data. This means that the best that we can do is to perform some form of clustering to *discover* automatically what is the best action label that describes the cluster. Intuitively, we want to learn the "meaning" of actions by grounding them to visual representations obtained from video data. Hence if we knew this grounding, we can assign action labels to the videos. On the other hand, if we know the action labels of the video data, we can estimate this grounding. This leads naturally to an iterative Expectation-Maximization (EM) formulation where we attempt to determine the optimal grounding parameters that will assign action labels to videos with the highest probability.

More formally, our goal is to label each video with their most likely action, along with the tool that is associated with the action. That is, we want to maximize the likelihood:

$$\begin{aligned} L(\mathcal{D}; A) &= \mathbb{E}_{\mathcal{P}(A)}[L(\mathcal{D}|A)] \\ &= \mathbb{E}_{\mathcal{P}(A)}[log\mathcal{P}(F_M, \mathcal{P}_I(\cdot), \mathcal{P}_L(\cdot)|A)] \end{aligned} \tag{3.1}$$

where A is the current (binary) action label assignments of the videos (see eq. (3.3)). $\mathcal{D}$ is the data computed from the video that consists of: 1) the language model $\mathcal{P}_L(\cdot)$ that predicts an action given the detected tool (sec. 3.1.2.1), 2) the tool detection

model $\mathcal{P}_I(\cdot)$ (sec. 3.1.2.2) and 3) the action features, F_M, associated with the video (sec. 3.1.2.3). We describe how these 3 data components are computed in the following paragraphs and detail how we optimize eq. (3.1) using EM in sec. 3.1.3.1.

3.1.2.1 Language as a predictor of actions

The key component of our approach is the language model that predicts the most likely verb (action) that is associated with a noun (tool) trained from a large text corpus. We view the Gigaword Corpus as a large text resource that contains the information we need to make correct predictions of actions given the detected tools from the video. We do this by training a language model $\mathcal{P}_L(v_j, n_i)$ that returns the maximum likelihood estimates of an action v_j given the tool n_i. This can be done by counting the number of times v_j co-occurs with n_i in a sentence:

$$\mathcal{P}_L(v_j|n_i) = \frac{\#(v_j, n_i)}{\#(n_i)} \tag{3.2}$$

As many English words share common meanings, a simple count of the action words (verbs) defined in $\mathcal{V}$ is likely to grossly underestimate the relationship between the tool and the action it is associated with. For example, in the Gigaword Corpus, counting the number of times `drink` co-occurs with `cup` where the actual words are used will not be significantly larger than `pick` and `cup`. The reason is that `cup` can mean a normal drinking cup or a trophy cup. In order to ensure that $\mathcal{P}_L$ captures the correct sense of the word (nouns, verbs), we use WordNet to determine the synonyms and hyponymns of the tools and actions considered. As illustrated in Fig. 3.3, extending `cup` to include `drinking_glass, wine_glass, mug` captures

the expected action `drink` in a larger part of the corpus.

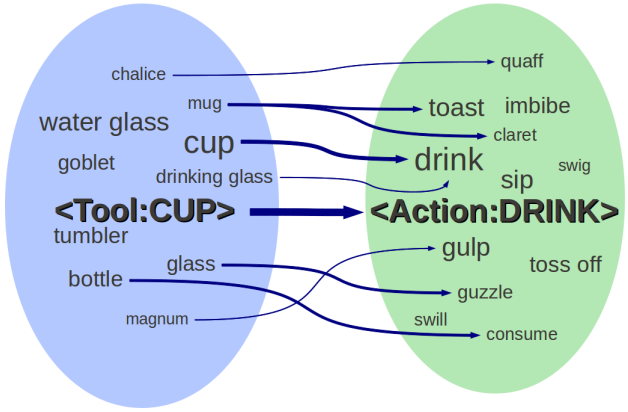


Figure 3.3: Enlarging the word class to contain synonyms yields more reasonable counts: `cup` only connects weakly with `drink`. By clustering other closely related words together, their combined counts increase the desired association between `cup` and `drink`.

We then recompute $\mathcal{P}_L$ using these enlarged word classes to capture more meaningful relationships between the co-occurring words. Fig. 3.4 shows the $|N| \times |V|$ co-occurrence matrix of likelihood scores over all the tools and actions considered in the experiments, denoted as $\mathcal{P}_L(V|N)$.

For most of the tool classes, the predicted actions are correct (large values along the diagonals): for e.g. `peeler` predicts `peeling` with high probability (0.94) and `shaker` predicts `sprinkling` at 0.66. This shows that for tools that have a *unique* use, our approach can predict the expected action easily. However, there are many co-occurrences which we could not anticipate. For e.g, using synonyms of `cup` makes it more selective to `drinking` (0.17) but it is `sprinkling` that has the highest prediction score (0.29). Investigating further reveals that `sprinkling` has

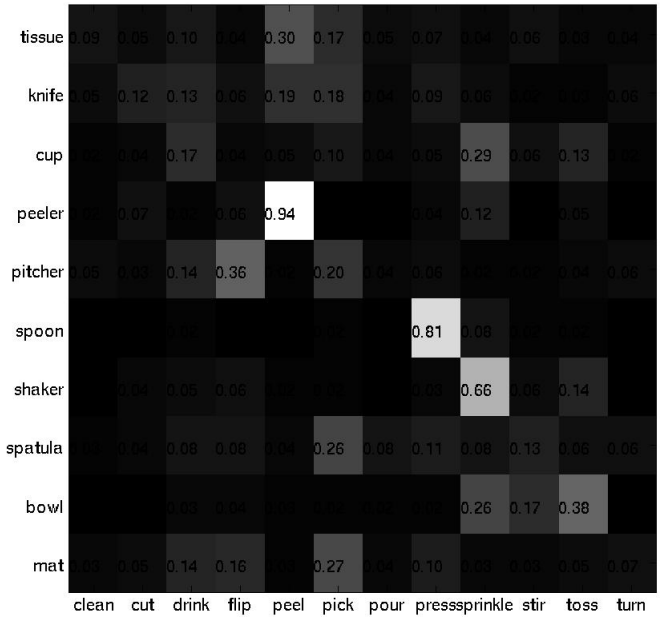


Figure 3.4: Gigaword co-occurrence matrix of tools and predicted actions.

some synonyms such as `drizzle moisten splash splosh` which have uses that are also close to `cup`. Other mis-selected tools-action are also due to the confusion at the synonyms/hyponymns levels. We also notice that more *general* actions such as `picking` have a more uniform distribution across the tools, which is expected. In the same way, the tool `mat` is also very general in its use such that it displays no significant selectivity to any of the actions. Despite this simplistic model, most of the entries in $\mathcal{P}_L$ make sense – and it properly reflects the innate complexity of language. As will be shown in sec. 3.1.4, although the priors from language are weak, they are still helpful for the task of action recognition.

3.1.2.2 Active tool detection strategy

We pursue the following active strategy for detecting, and subsequently recognizing, relevant tools in the video as illustrated in Fig. 3.5. First, a trained person detector [4] is used to determine the location of the human actor in the video frame. The location of the face is also detected using [77]. Optical flow is then computed [78] and we focus on human regions which have the highest flow, indicating the potential locations of the hands. We then apply a variant of a CRF-based color segmentation [79] using a trained skin color+flow model to segment the hand-like regions which are moving. This is justified by the fact that the moving hand is in contact with the tool that we want to identify. In some cases, the face may be detected (since it may be moving) but they are removed using the face detector results. We then apply a trained Partial Least Squares (PLS) object detector similar to [56] near the detected active hand region that returns a detection score at each video frame. Averaging out the detection yields $P_I(n_i|m_d)$, the probability that a tool $n_i \in N$ exists given the video m_d. We denote $P_I(N|M)$ as the set of detection scores (essentially the likelihood) over all tools in N and all videos in M.

This active approach has two important benefits. By focusing our processing only on the relevant regions of the video frame, we dramatically reduce the chance that the tool detector will misfire. At the same time, by detecting the hand locations, we obtain immediately the action trajectory, which is used to describe the action as shown in the next section.

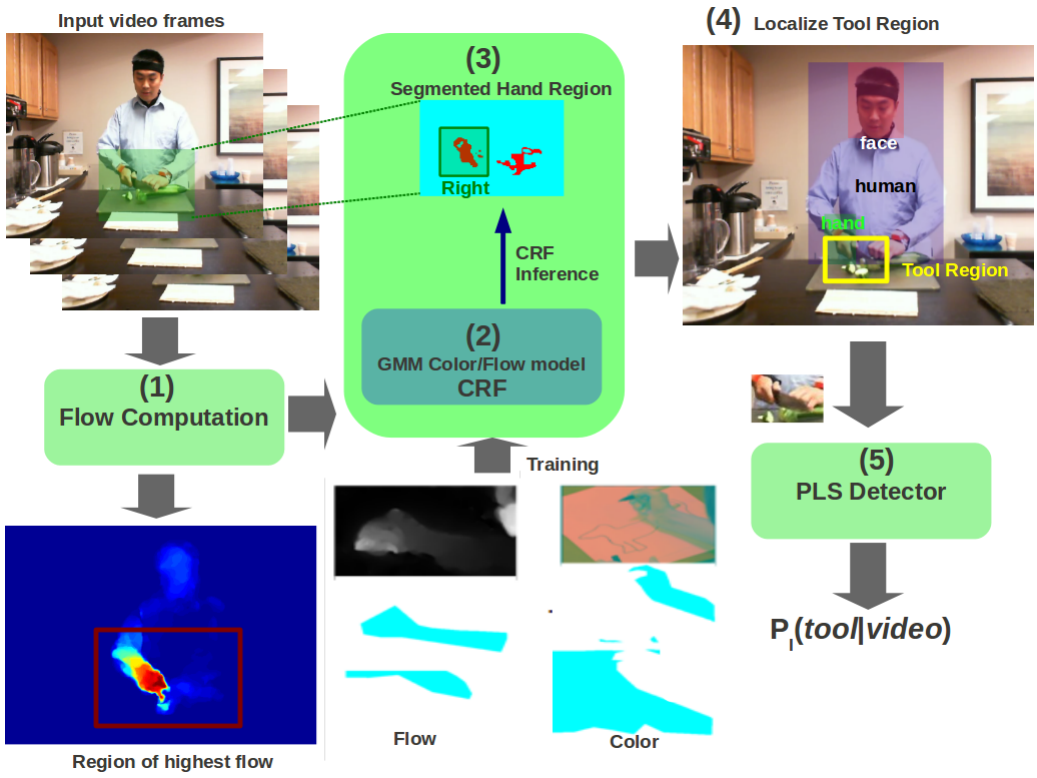


Figure 3.5: *(Best viewed in color)* Overview of the tool detection strategy: (1) Optical flow is first computed from the input video frames. (2) We train a CRF segmentation model based on optical flow + skin color. (3) Guided by the flow computations, we segment out hand-like regions (and removed faces if necessary) to obtain the hand regions that are moving (the active hand that is holding the tool). (4) The active hand region is where the tool is localized. Using the PLS detector (5), we compute a detection score for the presence of a tool.

3.1.2.3 Action features

Tracking the hand regions in the video provides us with two sets of (left and right) hand trajectories as shown in Fig. 3.6. We then construct for every video a feature vector F_d that encodes the hand trajectories. F_d encodes the frequency and

velocity components. Frequency is encoded by using the first 4 real coefficients of the Fourier transform in both the x and y directions, f_x, f_y, which gives a 16-dim vector over both hands. Velocity is encoded by averaging the difference in hand positions between two adjacent frames $\langle \delta x \rangle$, $\langle \delta y \rangle$ which gives a 4-dim vector. These features are then combined to yield a 20-dim vector F_d.

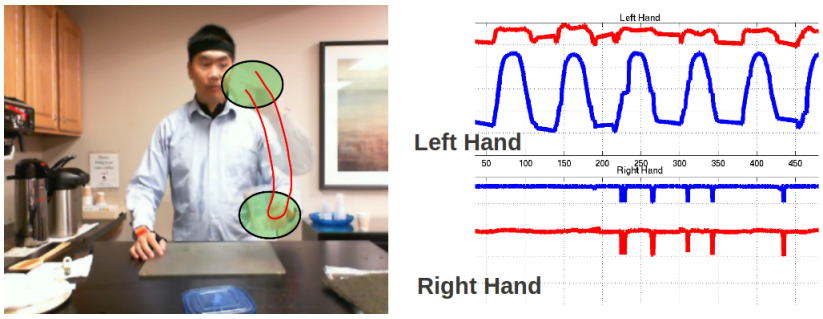


Figure 3.6: Detected hand trajectories. x and y coordinates are denoted as red and blue curves respectively.

We denote F_M as the set of of action features F_d over all videos in M.

3.1.3 Using Language to guide recognition

In this section, we formalize our EM approach to learn a joint tool-action model that assigns the most likely action label associated with a set of unlabeled video. We first derive from eq. (3.1) an expression that allows EM to estimate the parameters of this model, followed by details of the Expectation and Maximization steps. We then show how to use the learned model to perform testing. Finally, we consider the case where we have labeled data in which we formulate a simpler supervised approach.

3.1.3.1 Unsupervised learning of a joint tool-action model

We first define the latent assignment variable A. To simplify our notations, we will use subscripts to denote tools $i = n_i$, actions $j = v_j$ and videos $d = m_d$. For each $i \in N$, $j \in V$, $d \in M$, A_{ijd} indicates whether an action j is performed using tool i during video clip d.

$$A_{ijd} = \begin{cases} 1 & j \text{ is performed using } i \text{ during } d \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

and A is a 3D indicator matrix (or a 3D array) over all tools, actions and videos. Denoting the parameters of the model as $\mathcal{C} = \{\mathcal{C}_j\}$ which specifies the grounding of each action j, we seek to determine from eq. (3.1) the maximum likelihood parameter:

$$\mathcal{C}^* = \arg\max_{\mathcal{C}} \sum_{A} L(\mathcal{D}, A|\mathcal{C}) \tag{3.4}$$

Where,

$$\begin{aligned} L(\mathcal{D}, A|\mathcal{C}) &= \log P(\mathcal{D}, A|\mathcal{C}) \\ &= \log P(A|\mathcal{D}, \mathcal{C}) P(\mathcal{D}|\mathcal{C}) \end{aligned} \tag{3.5}$$

with the data $\mathcal{D}$ comprised of the tool detection likelihoods $\mathcal{P}_I(N|M)$, the tool-action likelihoods $\mathcal{P}_L(V|N)$ and action features F_M under the current model parameters $\mathcal{C}$. Geometrically, we can view $\mathcal{C}$ as the superset of the $|V|$ action label centers that defines our current grounding of each action j in the action feature space.

Using these centers, we can write the assignment given *each* video d, tool i

and action j, $P(A_{ijd}|\mathcal{D},\mathcal{C})$ as:

$$\mathcal{P}(A_{ijd}=1|\mathcal{D},\mathcal{C}) = \mathcal{P}_I(i|d)\mathcal{P}_L(j|i)Pen(d|j) \tag{3.6}$$

where $Pen(d|j)$ is an exemplar-based likelihood function defined between the associated action feature of video d, F_d and the current model parameter for action j, $\mathcal{C}_j$ as:

$$Pen(d|j) = \frac{1}{Z}\exp^{-||F_d-\mathcal{C}_j||^2} \tag{3.7}$$

where Z is a normalization factor. What eq. (3.7) encodes is the penalty that we score against the assignment when there is a large mismatch between F_d and $\mathcal{C}_j$, the cluster center of action j.

Rewriting eq. (3.6) over *all* videos M, tools N and actions V we have:

$$\mathcal{P}(A=1|\mathcal{D},\mathcal{C}) = \mathcal{P}_I(N|M)\mathcal{P}_L(V|N)Pen(F_M|C) \tag{3.8}$$

where we use the set variables to represent the full data and assignment model parameters. In the derivation that follows, we will simplify $\mathcal{P}(A=1|\mathcal{D},\mathcal{C})$ as $\mathcal{P}(A=1)$ and $\mathcal{P}(A=0) = 1-\mathcal{P}(A=1)$. We detail the Expectation and Maximization steps in the following sections.

Expectation step: We compute the expectation of the latent variable A, denoted by $\mathcal{W}$, according to the probability distribution of A given our current model parameters $\mathcal{C}$ and data ($\mathcal{P}_I$, $\mathcal{P}_L$, and F_M):

$$\begin{aligned}\mathcal{W} &= \mathbb{E}_{\mathcal{P}(A)}[A] \\ &= \mathcal{P}(A=1)\times 1 + (1-\mathcal{P}(A=1))\times 0 \\ &= \mathcal{P}(A=1)\end{aligned} \tag{3.9}$$

According to Eq. 3.6, the expectation of $\mathcal{A}$ is:

$$\mathcal{W} = \mathcal{P}(A = 1) \propto \mathcal{P}_I(N|M)\mathcal{P}_L(V|N)Pen(F_M|\mathcal{C}) \tag{3.10}$$

Specifically, for each $i \in N, j \in V, d \in M$:

$$\mathcal{W}_{ijd} \propto \mathcal{P}_I(i)\mathcal{P}_L(j|i)Pen(d|j) \tag{3.11}$$

Here, $\mathcal{W}$ is a $|N| \times |V| \times |M|$ matrix. Note that the constant of proportionality does not matter because it cancels out in the Maximization step.

Maximization step: The maximization step seeks to find the updated parameters $\hat{\mathcal{C}}$ that maximize eq. (3.5) with respect to $\mathcal{P}(A)$:

$$\hat{\mathcal{C}} = \arg\max_{\mathcal{C}} \mathbb{E}_{\mathcal{P}(A)}[\log \mathcal{P}(A|\mathcal{D}, \mathcal{C})\mathcal{P}(\mathcal{D}|\mathcal{C})] \tag{3.12}$$

Where $\mathcal{D} = \mathcal{P}_I, \mathcal{P}_L, F_M$. EM replaces $\mathcal{P}(A)$ with its expectation $\mathcal{W}$. As $A, \mathcal{P}_I, \mathcal{P}_L$ are independent of the model parameters $\mathcal{C}$, we can simplify eq. (3.12) to:

$$\begin{aligned} \hat{\mathcal{C}} &= \arg\max_{\mathcal{C}} \mathcal{P}(F_M|\mathcal{C}) \\ &= \arg\max_{\mathcal{C}} \left(-\sum_{i,j,d} \mathcal{W}_{ijd}||F_d - \mathcal{C}_j||^2 \right) \end{aligned} \tag{3.13}$$

where we had replaced $\mathcal{P}(F_M|\mathcal{C})$ with eq. (3.7) since the relationship between F_M and $\mathcal{C}$ is the penalty function $Pen(F_M|\mathcal{C})$. This enables us to define a target maximization function as $\mathbb{F}(\mathcal{C}) = \sum_{i,j,d} \mathcal{W}_{ijd}||F_d - \mathcal{C}_j||^2$.

According to the Karush-Kuhn-Tucker conditions, we can solve the maximization problem by the following constraint:

$$\frac{\partial \mathbb{F}}{\partial \mathcal{C}} = -2 \sum_{i,j,d} (\mathcal{W}_{ijd}(F_d - \mathcal{C}_j)) = 0 \tag{3.14}$$

Thus, for each $j \in V$, we have:

$$\hat{\mathcal{C}}_j = \frac{\sum_{i \in N, j \in V, d \in M} \mathcal{W}_{ijd} F_d}{\sum_{i \in N, j \in V, d \in M} \mathcal{W}_{ijd}} \tag{3.15}$$

We then update $\mathcal{C} = \hat{\mathcal{C}}$ within each iteration until convergence.

Testing the learned model: The learned model $\mathcal{C}^*$ can be used to classify new videos from a held-out testing set. Denoting the input test video as m_t, we predict the most likely action label, v_t^* by:

$$v_t^* = \arg\max_{j \in V} \sum_{i \in N} \left(\mathcal{P}_I(i|m_t) \mathcal{P}_L(j|i) Pen(F_t|\mathcal{C}_j^*) \right) \tag{3.16}$$

where F_t is the action features extracted from m_t and $\mathcal{C}_j^*$ is the j action center from the learned model.

3.1.3.2 Supervised action classification

If we have labeled video data of actions, a supervised approach will be the most straightforward. Every video, m_d, is represented by a set of features $\mathcal{F}_d$ that combines F_d (action features), $\mathcal{P}_I$ (tool detection) and $\mathcal{P}_L$ together in the following manner:

$$\begin{aligned} \mathcal{F}_d &= [F_d; \mathcal{P}_I(N|m_d); \mathcal{P}_I(N|m_d) \times \mathcal{P}_L(V|N)] \\ &= [F_d; \mathcal{P}_I(N|m_d); \mathcal{P}_L(V|m_d)] \end{aligned} \tag{3.17}$$

where ; means a concatenation of the features vectors. This yields a $20 + |N| + |V|$-dim vector. What eq. (3.17) means is that for every video m_d, we concatenate its F_d together with $\mathcal{P}_I(N|m_d)$, the distribution over all $|N|$ tools, and together

with the verb prediction: $\mathcal{P}_L(V|m_d)$, obtained from the text corpus. Given labeled training videos from all possible actions in $\mathcal{V}$, we can proceed to train discriminative classifiers (SVM, Bayes Net and Naive Bayes) to predict the action in the testing video.

3.1.4 Experiments

We performed a series of experiments using a new dataset of human actions performed with hand-tools to show quantitatively how language aids in action recognition. We first describe the dataset, and report recognition results on both the unsupervised and supervised scenarios.

3.1.5 The UMD Sushi-Making Dataset

The UMD Sushi-Making Dataset[2] consists of 12 actions, performed by 4 actors using 10 different kitchen tools, for the purpose of preparing sushi. This results in 48 video sequences each of around 1000 frames (30 seconds long). Other well known datasets such as the KTH, Weizmann or Human-EVA datasets [65, 80, 81] do not involve hand-tools. The human-object interaction dataset by Gupta et al. [82] has only 4 objects. The dataset by Messing et al. [58] has only 4 actions with tool use. The CMU Kitchen Dataset [83] has many tool interactions for 18 subjects making 5 recipes, but many of the actions are blocked from view due to the placements of the 4 static cameras. The head mounted camera gives a limited and shaky top-down view which cannot be processed easily.

[2] http://www.umiacs.umd.edu/research/POETICON/umd_sushi/

Our Sushi-Making dataset provides a clear view of the action in use with the tools and it simulates an active robotic agent observing the human actor performing the action in a realistic environment: a kitchen, with a real task: making sushi, that is made up of several actions that the robot must identify. See Fig. 3.5 for an example. The 12 actions are: cleaning, cutting, drinking, flipping, peeling, picking (up), pouring, pressing, sprinkling, stirring, tossing, turning. The tools used are: tissue, knife, cup, rolling-mat, fruit-peeler, water-pitcher, spoon, shaker, spatula, mixing-bowl. As was discussed in sec. 3.1.2.1, some of the actions such as `picking` or `flipping` are extremely general and are easily confused. We made this choice to ensure that the language prediction $\mathcal{P}_L$ is *not* perfect and to show that our approach works even under noisy data.

3.1.5.1 Baseline: Vision-only Recognition

As a baseline, we perform experiments without the language component, that is $\mathcal{P}_L$ in eq. (3.17) is not considered as part of $\mathcal{F}_d$. Two experiments: 1) clustering using K-means and 2) supervised classification are performed.

K-means clustering results: For the case of unlabeled videos, we performed K-means clustering with $k = 12$. We used 36 videos in this experiment. As labels, we took the majority ground truth labels from each cluster to be the predicted labels of the cluster. We then counted the number of videos that are correctly placed in the right cluster to derive a measure of accuracy, which is reported in Fig. 3.8(a).

Supervised classification results: From the 48 videos from the UMD Sushi-

Making dataset, we used 36 videos from 3 actors to train a degree 3 polynomial SVM classifier for the 12 actions. We set the cost parameter to 1.0 with a tolerance termination value at 0.001. These parameters were chosen from a separate development set of 8 videos. The remaining 12 videos were then used for testing. 4-fold cross validation was performed and the classification accuracy is reported. In addition, we trained a Naive Bayes (NB) classifier and a Bayes Net (BN) classifier over the training data. The BN is initialized as a NB with at most 1 parent. We then apply a simple estimator to estimate the conditional probability tables. All classifiers are tested using WEKA [84]. We summarize the results in Fig. 3.8(b).

3.1.5.2 Adding Language

In this section, we performed experiments with the aid of the language component $\mathcal{P}_L$. Three separate experiments are performed: 1) Unsupervised EM, 2) Semi-Supervised EM where we initialized the model parameters $\mathcal{C}$ with 12 *known* labels and 3) Supervised classification using trained SVM, Bayes Net and Naive Bayes classifiers. 36 videos were used for training the joint tool-action model using EM and 12 videos were held out for testing. For the supervised part, the same parameters as the baseline were used. We report our unsupervised and supervised results in Figs. 3.8(a) and 3.8(b) respectively. More detailed results (confusion matrices for each action) can be found online[3]. In addition, we show in Fig. 3.7 the improving recognition accuracy of the trained model at each iteration. The evolution of the action labels versus the ground truth is also presented. Testing on the

[3] http://www.umiacs.umd.edu/research/POETICON/umd_sushi/res_ICRA2012

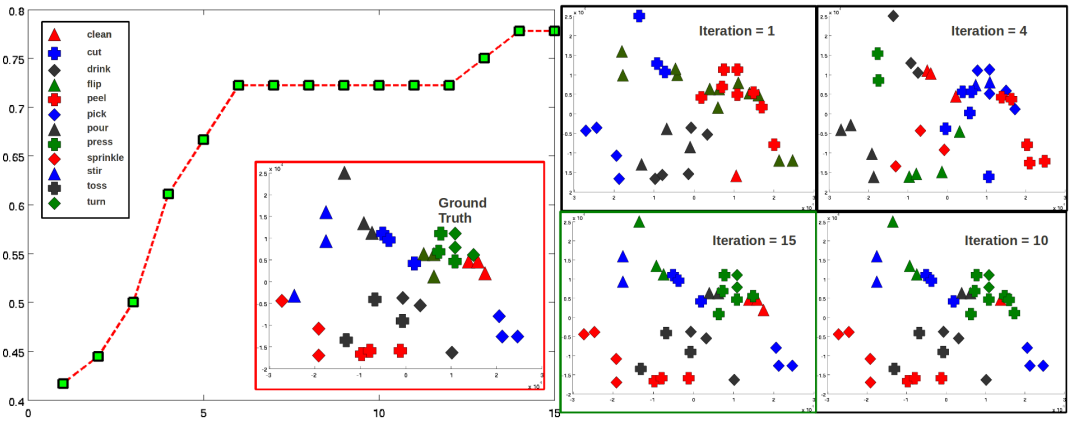


Figure 3.7: *(Best viewed in color)* (Left) Unsupervised EM: accuracy at each iteration. (Right) Scatterplots of action label assignments at selected iterations. We see that with each iteration, the assignment label clusters approaches the ground truth label (boxed in red). Note that we used PCA to reduce the action feature dimensions to 2 for visualization.

held out video set using the trained model yields a recognition accuracy of 83.33%.

3.1.5.3 Comparison with state of art action features

We compared our approach with a bag-of-words (BoW) representation built upon state of the art STIP [57] features clustered using K-means with $k = 50$. We trained three classifiers: SVM, Naive Bayes and Bayesian Net using the same procedure and parameters as the baseline and we summarize the results in Table 3.1. The BoW representation using STIP achieves a maximum classification rate of 77.08% with trained SVM classifiers. Our approach which uses comparatively *simpler* video features: Fourier coefficients + velocity eq. (3.17) outperforms the state of the art significantly. This gain is possible due to the addition of language prediction in the

Feature	Method	Accuracy
STIP+Bag of Words	Naive Bayes	56.25%
	Bayes Net	75%
	SVM	77.08%
Action Features+Language	Naive Bayes	66.67%
	Bayes Net	85.41%
	SVM	91.67%
Action Features+Language	Unsupervised EM	77.78%
	Semi-supervised EM	91.67%

Table 3.1: Classification accuracy: STIP versus our approach.

action feature.

3.1.5.4 Discussion: the effects of adding language

Comparing the unsupervised recognition results, K-means clustering on the action features alone (with $\mathcal{P}_L$) achieves only 69.44% recognition rate. The clustering accuracy, with the addition of $\mathcal{P}_L$ and using the EM formulation described in sec. 3.1.3.1 achieves 77.78% with random initialization of the model $\mathcal{C}$. We show further that with correctly initialized parameters from 12 labeled videos, is enough to increase the accuracy to 91.67%, which is just as good as the SVM classifier (which is supervised). This result shows that once again, even with no or limited labeled data, our proposed EM formulation is able to leverage the predictive power of $\mathcal{P}_L$ to find the optimal action and its corresponding tool that best explains the video.

Fig. 3.9 shows some video frames with the predicted action and corresponding tool using EM.

For the classification results in Fig. 3.8(b) using the three classifiers, we clearly see that the addition of $\mathcal{P}_L$ increases the classification accuracy, with the most dramatic increase when SVM or Naive Bayes are used: from 87.5% to 91.67% (SVM) and 62.5% to 66.67% when language is added. This shows that even with a simple model, $\mathcal{P}_L$ is able to provide additional discriminatory features which improve the classification. The most important result is that these features are estimated directly from a generic text corpus and the method is not limited to a particular domain such as cooking. This fact alone highlights the strength of language in aiding action classification.

Besides improving action recognition accuracy in both supervised and unsupervised scenarios, another key observation from our results is that language is *complementary* in aiding many vision related tasks where the use of high-level knowledge is required. Previous works described in sec. 3.1.1 have shown that language (in a restricted sense) can be used to simplify ill-posed image problems like segmentation and annotation. We showed here that the difficult problem of recognizing actions involves high-level knowledge as well. This is because of the strong relationship between the actions and the tools that were used to perform these actions. Instead of learning from a huge amount of training *image* data on how tools correlate with actions, we showed that it is possible to obtain such information directly from a *text* corpus. Such a text corpus, although noisy, is much easier to obtain than annotated image data; and we showed that with the right EM formulation, the noisy

predictions from $\mathcal{P}_L$ provides us appreciable gains in recognition rates on *unlabeled* video.

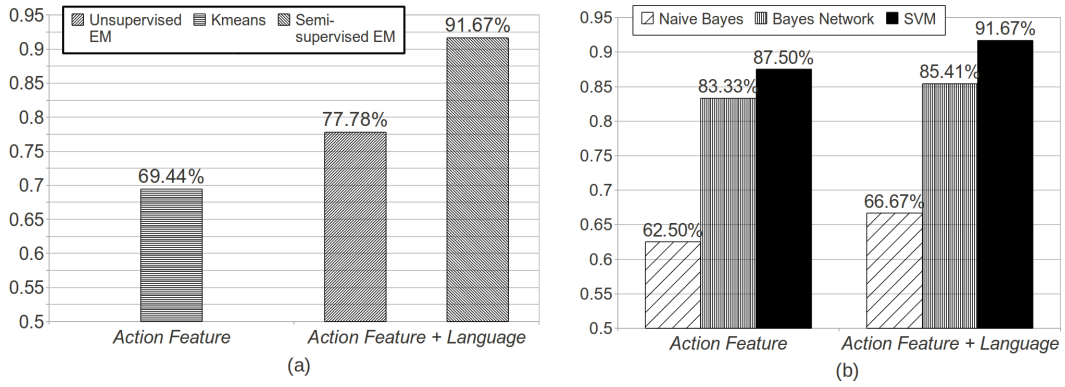


Figure 3.8: (a) Unsupervised recognition accuracy: no language (K-Means) versus language (EM). (b) Classification accuracy: no language versus language. All reported results have variances within $\pm 0.5\%$.

3.2 Language Guided Scene Understanding for Robots

What happens when you see a picture? The most natural thing would be to *describe* it using *words*: using speech or text. This description of an image is the output of an extremely complex process that involves: 1) perception in the Visual space, 2) grounding to World Knowledge in the Language Space and 3) speech/text production (see Fig. 3.10). Each of these components are challenging in their own right and are still considered open problems in the vision and linguistics fields. In this chapter, we introduce a computational framework that attempts to integrate these components together. Our hypothesis is based on the assumption that natural images accurately reflect common everyday scenarios which are captured in

Figure 3.9: Some predicted action and tools using EM. The wrong prediction (in red and italicized) of the `sprinkle` action is due to a high co-occurrence with `bowl` in $P_L(V|N)$.

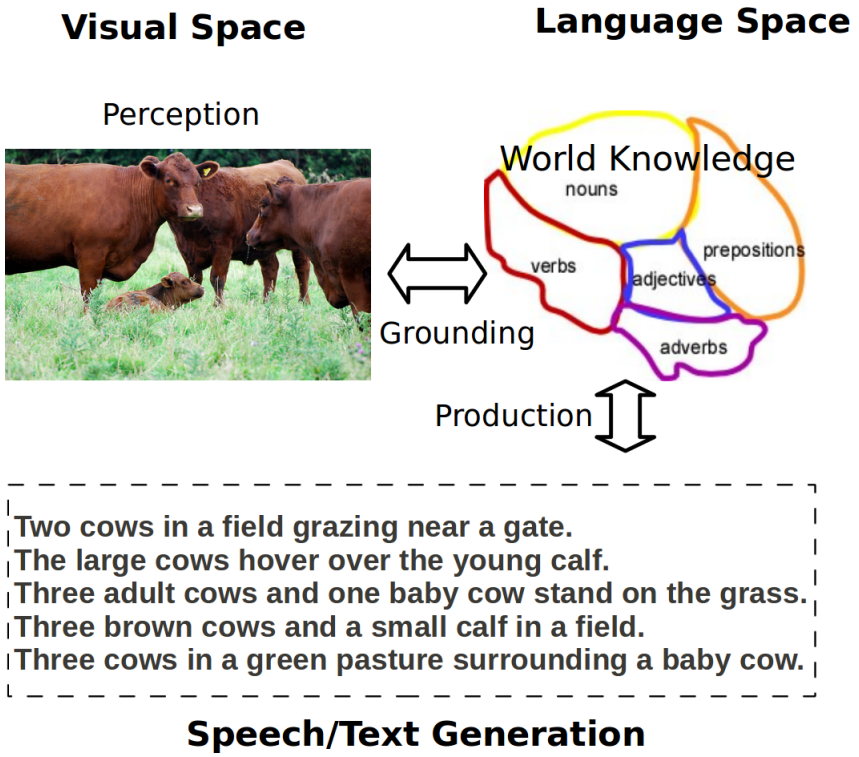


Figure 3.10: The processes involved for describing a scene.

language. For example, knowing that boats usually occur over `water` will enable us to constrain the possible scenes a boat can occur and exclude highly unlikely ones – `street, highway`. It also enables us to predict likely actions (Verbs) given the

current object detections in the image: detecting a dog with a person will likely induce `walk` rather than `swim, jump, fly`. Key to our approach is the use of a large generic corpus such as the English Gigaword [76] as the *semantic grounding* to predict and correct the initial and often noisy visual detections of an image to produce a reasonable sentence that succinctly describes the image.

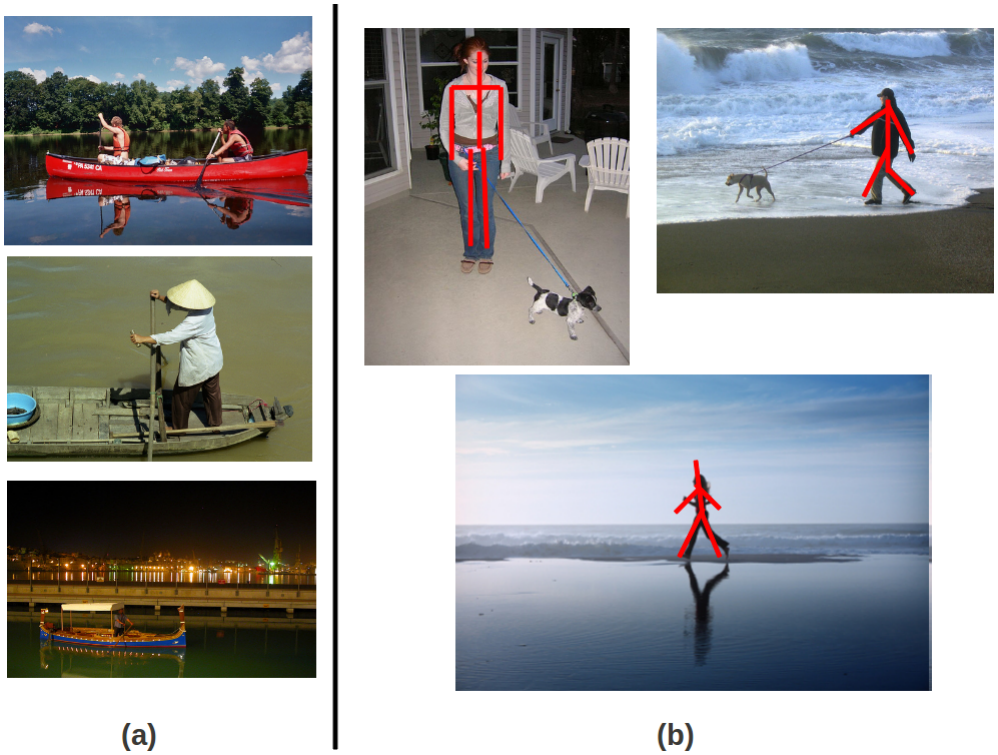


Figure 3.11: Illustration of various perceptual challenges for sentence generation for images. (a) Different images with semantically the same content. (b) Pose relates ambiguously to actions in real images.

In order to get an idea of the difficulty of this task, it is important to first define what makes up a description of an image. Based on our observations of annotated image data (see Fig. 3.13), a descriptive sentence for an image must contain at minimum: 1) the important *objects* (Nouns) that participate in the image,

2) Some description of the *actions* (Verbs) associated with these objects, 3) the *scene* where this image was taken and 4) the *preposition* that relates the objects to the scene. That is, a quadruplet of $\mathcal{T} = \{n, v, s, p\}$ (Noun-Verb-Scene-Preposition) that represents the core sentence structure. Generating a sentence from this quadruplet is obviously a simplification from state of the art generation work, but as we will show in the experimental results (sec. 3.2.4), it is sufficient to describe images. The key challenge is that detecting objects, actions and scenes *directly* from images is often noisy and unreliable. We illustrate this using example images from the Pascal-Visual Object Classes (VOC) 2008 challenge [85]. First, Fig. 3.11(a) shows the *variability* of images in their raw image representations: pixels, edges and local features. This makes it difficult for state of the art object detectors [4,56] to reliably detect important objects in the scene: boat, humans and water – average precision scores reported in [4] manages around 42% for humans and only 11% for boat over a dataset of almost 5000 images in 20 object categories. Yet, these images are *semantically* similar in terms of their high level description. Second, cognitive studies [86, 87] have proposed that inferring the action from static images (known as an "implied action") is often achieved by detecting the *pose* of humans in the image: the position of the limbs with respect to one another, under the assumption that a unique pose occurs for a unique action. Clearly, this assumption is weak as 1) similar actions may be represented by different poses due to the inherent dynamic nature of the action itself: e.g. walking a dog and 2) different actions may have the same pose: e.g. walking a dog versus running (Fig. 3.11(b)). The missing component here is whether the key object (dog) under interaction is considered.

Recent works [88, 89] that used poses for recognition of actions achieved 70% and 61% accuracy respectively under extremely limited testing conditions with only 5-6 action classes each. Finally, state of the art scene detectors [5, 90] need to have enough representative training examples of scenes from pre-defined scene classes for a classification to be successful – with a reported average precision of 83.7% tested over a dataset of 2600 images.

Addressing all these visual challenges is clearly a formidable task which is beyond the scope of this chapter. Our focus instead is to show that with the addition of *language* to ground the noisy initial visual detections, we are able to improve the quality of the generated sentence as a faithful description of the image. In particular, we show that it is possible to avoid predicting actions directly from images – which is still unreliable – and to use the corpus instead to guide our predictions. Our proposed strategy is also *generic*, that is, we make no prior assumptions on the image domain considered. While other works (sec. 3.2.1) depend on strong annotations between images and text to ground their predictions (and to remove wrong sentences), we show that a large generic corpus is also able to provide the same grounding over larger domains of images. It represents a relatively new style of learning: distant supervision [91, 92]. Here, we do not require "labeled" data containing images and captions but only separate data from each side. Another contribution is a computationally feasible way via dynamic programming to determine the most likely quadruplet $\mathcal{T}^* = \{n^*, v^*, s^*, p^*\}$ that describes the image for generating possible sentences.

3.2.1 Related Work

Recently, several works from the Computer Vision domain have attempted to use language to aid image scene understanding. [93] used predefined production rules to describe actions in videos. [71] processed news captions to discover names associated with faces in the images, and [72] extended this work to associate poses detected from images with the verbs in the captions. Both approaches use annotated examples from a limited news caption corpus to learn a joint image-text model so that one can annotate new unknown images with textual information easily. Neither of these works have been tested on complex everyday images where the large variations of objects and poses makes it nearly impossible to learn a more general model. In addition, no attempt was made to generate a descriptive sentence from the learned model. The work of [94] attempts to "generate" sentences by first learning from a set of human annotated examples, and producing the *same* sentence if both images and sentence share common properties in terms of their triplets: (Nouns-Verbs-Scenes). No attempt was made to generate *novel* sentences from images beyond what has been annotated by humans. [95] has recently introduced a framework for parsing images/videos to textual description that requires significant annotated data, a requirement that our proposed approach avoids.

Natural language generation (NLG) is a long-standing problem. Classic approaches [96] are based on three steps: selection, planning and realization. A common challenge in generation problems is the question of: what is the input? Recently, approaches for generation have focused on formal specification inputs, such

as the output of theorem provers [97] or databases [98]. Most of the effort in those approaches has focused on selection and realization. We address a tangential problem that has not received much attention in the generation literature: how to deal with *noisy inputs*. In our case, the inputs themselves are often uncertain (due to misrecognitions by object/scene detectors) and the content selection and realization needs to take this uncertainty into account.

3.2.2 Our Approach

Our approach is summarized in Fig. 3.12. The input is a test image where we detect objects and scenes using trained detection algorithms [4, 5]. To keep the framework computationally tractable, we limit the elements of the quadruplet (Nouns-Verbs-Scenes-Prepositions) to come from a finite set of objects $\mathcal{N}$, actions $\mathcal{V}$, scenes $\mathcal{S}$ and prepositions $\mathcal{P}$ classes that are commonly encountered. They are summarized in Table. 3.2. In addition, the sentence that is generated for each image is limited to at most two objects occurring in a unique scene.

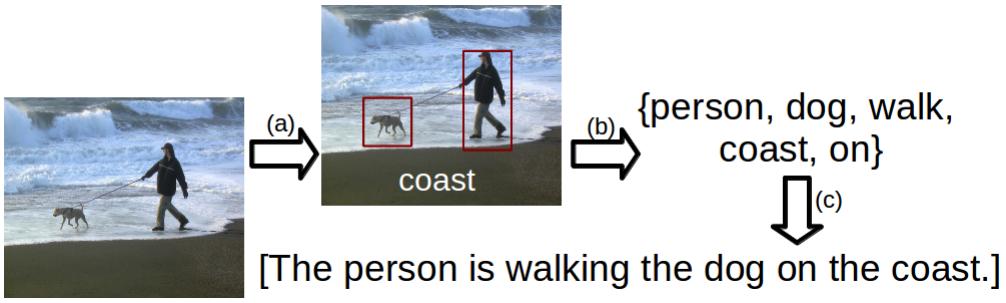


Figure 3.12: Overview of our approach. (a) Detect objects and scenes from input image. (b) Estimate optimal sentence structure quadruplet $\mathcal{T}^*$. (c) Generating a sentence from $\mathcal{T}^*$.

Objects $n \in \mathcal{N}$	Actions $v \in \mathcal{V}$	Scenes $s \in \mathcal{S}$	Preps $p \in \mathcal{P}$
'aeroplane' 'bicycle'	'sit' 'stand' 'park'	'airport'	'in' 'at'
'bird' 'boat' 'bottle'	'ride' 'hold' 'wear'	'field'	'above' 'around'
'bus' 'car' 'cat' 'chair'	'pose' 'fly' 'lie'	'highway'	'behind' 'below'
'cow' 'table' 'dog'	'lay' 'smile' 'live'	'lake'	'beside'
'horse', 'motorbike'	'walk' 'graze' 'drive'	'room' 'sky'	'between'
'person' 'pottedplant'	'play' 'eat' 'cover'	'street'	'before' 'to'
'sheep' 'sofa' 'train'	'train' 'close' ...	'track'	'under' 'on'
'tvmonitor'			

Table 3.2: The set of objects, actions (first 20), scenes and preposition classes considered

Denoting the current test image as I, the initial visual processing first detects objects $n \in \mathcal{N}$ and scenes $s \in \mathcal{S}$ using these detectors to compute $P_r(n|I)$ and $P_r(s|I)$, the probabilities that object n and scene s exist under I. From the observation that an action can often be predicted by its key objects, $N_k = \{n_1, n_2, \cdots, n_i\}, n_i \in \mathcal{N}$ that participate in the action, we use a trained Language model L_m to estimate $P_r(v|N_k)$. L_m is also used to compute $P_r(s|n, v)$, the predicted scene using the corpus given the object and verb; and $P_r(p|s)$, the predicted preposition given the scene. This process is repeated over all n, v, s, p where we used a modified HMM inference scheme to determine the most likely quadruplet: $\mathcal{T}^* = \{n^*, v^*, s^*, p^*\}$ that makes up the core sentence structure. Using the contents and structure of $\mathcal{T}^*$, an appropriate sentence is then generated that describes the image. In the following sections, we first introduce the image dataset used for testing followed by details of how these components are derived.

3.2.2.1 Image Dataset

The cow is grazing in a field.
An ox stands in a field
A yak with a long, camel colored coat standing in a field.
A young highlander cow stands in a pasture.
Closeup of a bull with hair covering its eyes

an Asian woman sitting in a chair on her balcony
A woman smiling.
Smiling Asian woman in floral dress.
The happy lady enjoys her surroundings.
The woman in the floral dress is posing among plants.

A dinner table set for three people.
A Thanksgiving meal with white daisies on a small table.
Dinner sitting on a table and ready to be served.
There is a turkey on the table along with other foods on plates.
The table is set for a turkey dinner and decorated-
with white daisies.

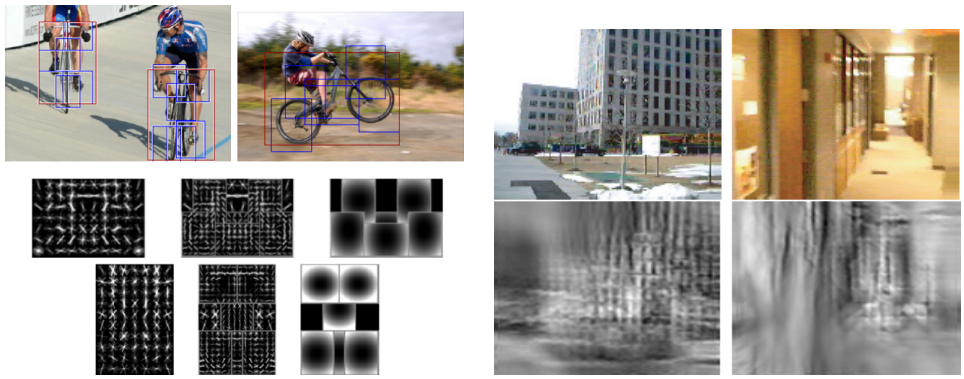
Figure 3.13: Samples of images with corresponding annotations from the UIUC scene description dataset.

We use the *UIUC Pascal Sentence dataset*, first introduced in [94] and available on-line[4]. It contains 1000 images taken from a subset of the Pascal-VOC 2008 challenge image dataset and are hand annotated with sentences that describe the image by paid human annotators using Amazon Mechanical Turk. Fig. 3.13 shows some sample images with their annotations. There are 5 annotations per image, and each annotation is usually short – around 10 words long. We randomly selected 900 images (4500 sentences) as the learning corpus to construct the verb and scene sets, $\{\mathcal{V}, \mathcal{S}\}$ as described in sec. 3.2.3.1, and kept the remaining 100 images for testing

[4] `http://vision.cs.uiuc.edu/pascal-sentences/`

and evaluation.

3.2.3 Object and Scene Detections from Images



(a) (b)

Figure 3.14: (a) [Top] The part based object detector from [4]. [Bottom] The graphical model representation of an object, for e.g. a bike. (b) Examples of GIST gradients: (left) an outdoor scene vs (right) an indoor scene [5].

We use the Pascal-VOC 2008 trained object detectors [99] of 20 common everyday object classes that are defined in $\mathcal{N}$. Each of the detectors are essentially SVM classifiers trained on a large number of the objects' image representations from a large variety of sources. Although 20 classes may seem small, their existence in many natural images (e.g. humans, cars and plants) makes them particularly *important* for our task, since humans tend to describe these common objects as well. As object representations, the part-based descriptor of [4] is used. This representation decomposes any object, e.g. a cow, into its constituent parts: head, torso, legs, which are shared by other objects in a hierarchical manner. At each level, image gradient orientations are computed. The relationship between each parts is modeled

probabilistically using graphical models where parts are the nodes and the edges are the conditional probabilities that relate their spatial compatibility (Fig. 3.14(a)). For example, in a cow, the probability of finding the torso near the head is higher than finding the legs near the head. This model's intuition lies in the assumption that objects can be deformed but the relative position of each constituent parts should remain the same. We convert the object detection scores to probabilities using Platt's method [100] which is numerically more stable to obtain $P_r(n|I)$. The parameters of Platt's method are obtained by estimating the number of positives and negatives from the UIUC annotated dataset, from which we determine the appropriate probabilistic threshold, which gives us approximately 50% recall and precision.

For detecting scenes defined in $\mathcal{S}$, we use the GIST-based scene descriptor of [5]. GIST computes the windowed 2D Gabor filter responses of an input image. The responses of Gabor filters (4 scales and 6 orientations) encode the texture gradients that describe the *local* properties of the image. Averaging out these responses over larger spatial regions gives us a set of *global* image properties. These high dimensional responses are then reprojected to a low dimensional space via PCA, where the number of principal components are obtained empirically from training scenes. This representation forms the GIST descriptor of an image (Fig. 3.14(b)) which is used to train a set of SVM classifiers for each scene class in $\mathcal{S}$. Again, $P_r(s|I)$ is computed from the SVM scores using [100]. The set of common scenes defined in $\mathcal{S}$ is learned from the UIUC annotated data (sec. 3.2.3.1).

3.2.3.1 Corpus-Guided Predictions

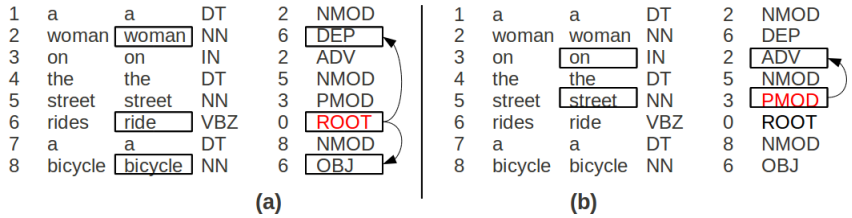


Figure 3.15: (a) Selecting the ROOT verb from the dependency parse `ride` reveals its subject `woman` and direct object `bicycle`. (b) Selecting the head noun (PMOD) as the scene `street` reveals ADV as the preposition `on`

Predicting Verbs: The key component of our approach is the trained language model L_m that predicts the most likely verb v, associated with the objects N_k detected in the image. Since it is possible that different verbs may be associated with varying number of object arguments, we limit ourselves to verbs that take on at most *two* objects (or more specifically two noun phrase arguments) as a simplifying assumption: $N_k = \{n_1, n_2\}$ where n_2 can be NULL. That is, n_1 and n_2 are the subject and direct objects associated with $v \in \mathcal{V}$. Using this assumption, we can construct the set of verbs, $\mathcal{V}$. To do this, we use human labeled descriptions of the training images from the UIUC Pascal-VOC dataset (sec. 3.2.2.1) as a learning corpus that allows us to determine the appropriate target verb set that is amenable to our problem. We first apply the CLEAR parser [101] to obtain a dependency parse of these annotations, which also performs stemming of all the verbs and nouns in the sentence. Next, we process all the parses to select verbs which are marked as ROOT and check the existence of a subject (DEP) and direct object (PMOD,

OBJ) that are linked to the ROOT verb (see Fig. 3.15(a)). Finally, after removing common "stop" verbs such as {`is, are, be`} we rank these verbs in terms of their occurrences and select the top 50 verbs which accounts for 87.5% of the sentences in the UIUC dataset to be in $\mathcal{V}$.

Object class $n \in \mathcal{N}$	Synonyms, $\langle n \rangle$
`bus`	`autobus charabanc double-decker jitney motorbus motorcoach omnibus passenger-vehicle schoolbus trolleybus streetcar ...`
`chair`	`highchair chaise daybed throne rocker armchair wheelchair seat ladder-back lawn-chair fauteuil ...`
`bicycle`	`bike wheel cycle velocipede tandem mountain-bike ...`

Table 3.3: Samples of synonyms for 3 object classes.

Next, we need to explain how n_1 and n_2 are selected from the 20 object classes defined previously in $\mathcal{N}$. Just as the 20 object classes are defined *visually* over several different kinds of specific objects, we expand n_1 and n_2 in their textual descriptions using *synonyms*. For example, the object class n_1=`aeroplane` should include the synonyms {`plane, jet, fighter jet, aircraft`}, denoted as $\langle n_1 \rangle$. To do this, we expand each object class using their corresponding WordNet synsets up to at most three hyponymns levels. Example synonyms for some of the classes are summarized in Table 3.3.

We can now compute from the Gigaword corpus [76] the probability that a

verb exists given the detected nouns, $P_r(v|n_1, n_2)$. We do this by computing the log-likelihood ratio [102] , λ_{nvn}, of *trigrams* $(\langle n_1 \rangle, v, \langle n_2 \rangle)$, computed from each sentence in the English Gigaword corpus [76]. This is done by extracting only the words in the corpus that are defined in $\mathcal{N}$ and $\mathcal{V}$ (including their synonyms). This forms a *reduced* corpus sequence from which we obtain our target trigrams. For example, the sentence:

`the large brown dog chases a small young cat around the messy room, forcing the cat to run away towards its owner.`

will be reduced to the stemmed sequence `dog chase cat cat run owner`[5] from which we obtain the target trigram relationships: {`dog chase cat`}, {`cat run owner`} as these trigrams respect the (n_1, v, n_2) ordering. The log-likelihood ratios, λ_{nvn}, computed for all possible $(\langle n_1 \rangle, v, \langle n_2 \rangle)$ are then normalized to obtain $P_r(v|n_1, n_2)$. An example of ranked λ_{nvn} in Fig. 3.16(a) shows that λ_{nvn} predicts v that makes sense: with the most likely predictions near the top of the list.

Predicting Scenes: Just as an action is strongly related to the objects that participate in it, a scene can be predicted from the objects and verbs that occur in the image. For example, detecting N_k={`boat, person`} with v={`row`} would have predicted the scene s={`coast`}, since boats usually occur in water regions. To learn this relationship from the corpus, we use the UIUC dataset to discover what are the common scenes that should be included in $\mathcal{S}$. We applied the CLEAR dependency parse [101] on the UIUC data and extracted all the head nouns (PMOD) in the PP phrases for this purpose and excluded those nouns with prepositions (marked

[5]stemming is done using [101]

as ADV) such as {`with, of`} which do not co-occur with scenes in general (see Fig. 3.15(b)). We then ranked the remaining scenes in terms of their frequency to select the top 8 scenes used in $\mathcal{S}$.

To improve recall and generalization, we expand each of the 8 scene classes using their WordNet synsets $\langle s \rangle$ (up to a max of three hyponymns levels). Similar to the procedure of predicting the verbs described above, we compute the log-likelihood ratio of ordered *bigrams*, $\{n, \langle s \rangle\}$ and $\{v, \langle s \rangle\}$: λ_{ns} and λ_{vs}, by reducing the corpus sentence to the target nouns, verbs and scenes defined in $\mathcal{N}, \mathcal{V}$ and $\mathcal{S}$. The probabilities $P_r(s|n)$ and $P_r(v|n)$ are then obtained by normalizing λ_{ns} and λ_{vs}. Under the assumption that the priors $P_r(n)$ and $P_r(v)$ are independent and applying Bayes rule, we can compute the probability that a scene co-occurs with the object and action, $P_r(s|n, v)$ by:

$$\begin{aligned} P_r(s|n,v) &= \frac{P_r(n,v|s)P_r(s)}{P_r(n,v)} \\ &= \frac{P_r(n|s)P_r(v|s)P_r(s)}{P_r(n)P_r(v)} \\ &\propto P_r(s|n) \times P_r(s|v) \end{aligned} \tag{3.18}$$

where the constant of proportionality is justified under the assumption that $P_r(s)$ is equiprobable for all s. (3.18) is computed for all nouns in N_k. As shown in Fig. 3.16(b), we are able to predict scenes that co-locate with reasonable correctness given the nouns and verbs.

Predicting Prepositions: It is straightforward to predict the appropriate prepositions associated with a given scene. When we construct $\mathcal{S}$ from the UIUC annotated data, we simply collect and rank all the associated prepositions (ADV)

in the PP phrase of the dependency parses. We then select the top 12 prepositions used to define $\mathcal{P}$. Using $\mathcal{P}$, we then compute the log-likelihood ratio of ordered *bigrams*, $\{p, \langle s \rangle\}$ for prepositions that co-locate with the scene synonyms over the corpus. Normalizing λ_{ps} yields $P_r(p|s)$, the probability that a preposition co-locates with a scene. Examples of ranked λ_{ps} are shown in Fig. 3.16(c). Again, we see that reasonable predictions of p can be found.

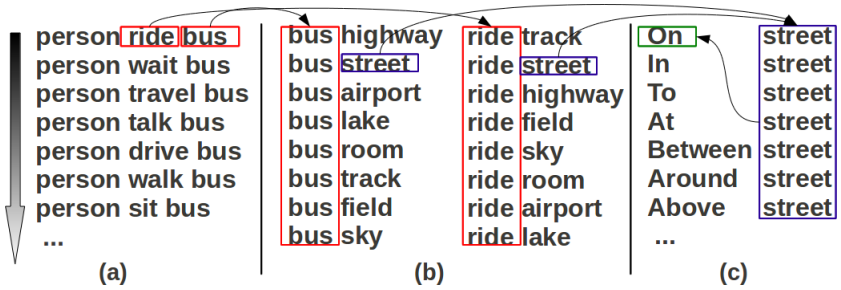


Figure 3.16: Example of how ranked log-likelihood values (in descending order) suggest a possible $\mathcal{T}$: (a) λ_{nvn} for $n_1 = \texttt{person}, n_2 = \texttt{bus}$ predicts $v = \texttt{ride}$. (b) λ_{ns} and λ_{vs} for $n = \texttt{bus}, v = \texttt{ride}$ then jointly predicts $s = \texttt{street}$ and finally (c) λ_{ps} with $s = \texttt{street}$ predicts $p = \texttt{on}$.

3.2.3.2 Determining $\mathcal{T}^*$ using HMM inference

Given the computed conditional probabilities: $P_r(n|I)$ and $P_r(s|I)$ which are observations from an input test image with the parameters of the trained language model, L_m: $P_r(v|n_1, n_2), P_r(s|n, v), P_r(p|s)$, we seek to find the most likely sentence

structure $\mathcal{T}^*$ by:

$$\begin{aligned}\mathcal{T}^* &= \arg\max_{n,v,s,p} P_r(\mathcal{T}|n,v,s,p) \\ &= \arg\max_{n,v,s,p}\{P_r(n_1|I)P_r(n_2|I)P_r(s|I)\times \\ &\qquad P_r(v|n_1,n_2)P_r(s|n,v)P_r(p|s)\} \end{aligned} \tag{3.19}$$

where the last equality holds by assuming independence between the visual detections and corpus predictions. Obviously a brute force approach to try all possible combinations to maximize eq. (3.19) will not be feasible due to the large number of possible combinations: $(20*21*8)*(50*20*20)*(8*20*50)*(12*8) \approx 5\times 10^{13}$. A better solution is needed.

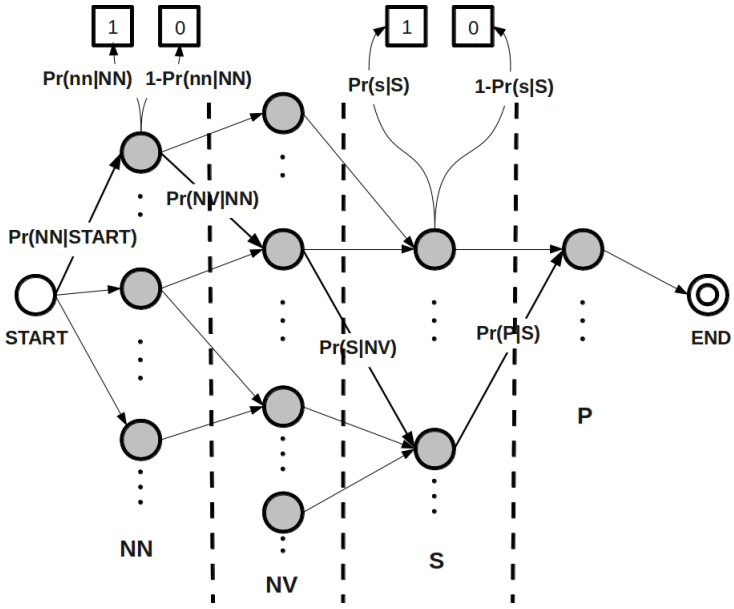


Figure 3.17: The HMM used for optimizing $\mathcal{T}$. The relevant transition and emission probabilities are also shown. See text for more details.

Our proposed strategy is to pose the optimization of $\mathcal{T}$ as a dynamic programming problem, akin to a Hidden Markov Model (HMM) where the hidden states are

related to the (simplified) sentence structure we seek: $\mathcal{T} = \{n_1, n_2, s, v, p\}$, and the emissions are related to the observed detections: $\{n_1, n_2, s\}$ in the image if they exist. To simplify our notations, as we are concerned with object pairs we will write `NN` as the hidden states for all n_1, n_2 pairs and `nn` as the corresponding emissions (detections); and all object+verb pairs as hidden states `NV`. The hidden states are therefore denoted as: $\{\texttt{NN}, \texttt{NV}, \texttt{S}, \texttt{P}\}$ with values taken from their respective word classes from Table 3.2. The emission states are $\{\texttt{nn}, \texttt{s}\}$ with binary values: 1 if the detections occur or 0 otherwise. The full HMM is summarized in Fig. 3.17. The rationale for using a HMM is that we can reuse all previous computation of the probabilities at each level to compute the required probabilities at the current level. From `START`, we assume all object pair detections are equiprobable: $P_r(\texttt{NN}|\texttt{START}) = \frac{1}{|\mathcal{N}|*(|\mathcal{N}|+1)}$ where we have added an additional `NULL` value for objects (at most 1). At each `NN`, the HMM emits a detection from the image and by independence we have: $P_r(\texttt{nn}|\texttt{NN}) = P_r(n_1|I)P_r(n_2|I)$. After `NN`, the HMM transits to the corresponding verb at state `NV` with $P_r(\texttt{NV}|\texttt{NN}) = P_r(v|n_1, n_2)$ obtained from the corpus statistic[6]. As no action detections are performed on the image, `NV` has no emissions. The HMM then transits from `NV` to `S` with $P_r(\texttt{S}|\texttt{NV}) = P_r(s|n, v)$ computed from the corpus which emits the scene detection score from the image: $P_r(\texttt{s}|\texttt{S}) = P_r(s|I)$. From `S`, the HMM transits to `P` with $P_r(\texttt{P}|\texttt{S}) = P_r(p|s)$ before reaching the `END` state.

Comparing the HMM with eq. (3.19), one can see that all the corpus and detection probabilities are accounted for in the transition and emission probabilities respectively. Optimizing $\mathcal{T}$ is then equivalent to finding the best (most likely) path

[6]each verb, v, in `NV` will have 2 entries with the same value, one for each noun.

through the HMM given the image observations using the Viterbi algorithm which can be done in $O(10^5)$ time which is significantly faster than the naive approach. We show in Fig. 3.18 (right-upper) examples of the top viterbi paths that produce $\mathcal{T}^*$ for four test images[7].

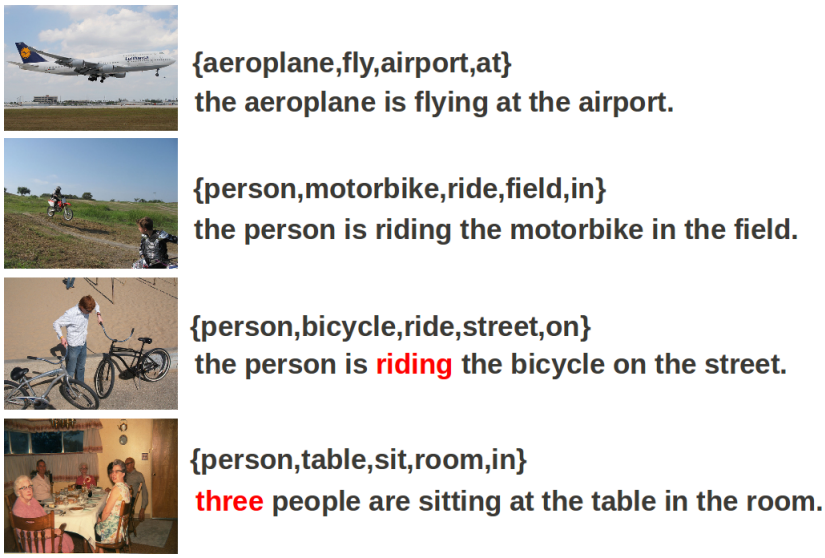


Figure 3.18: Four test images (left) and results. (Right-upper): Sentence structure $\mathcal{T}^*$ predicted using Viterbi and (Right-lower): Generated sentences. Words marked in red are considered to be incorrect predictions.

Note that the proposed HMM is suitable for generating sentences that contain the core components defined in $\mathcal{T}$ which produces a sentence of the form `NP-VP-PP`, which we will show in sec. 3.2.4 is sufficient for the task of generating sentences for describing images. For more complex sentences with more components: such as adjectives or adverbs, the HMM can be easily extended with similar computations derived from the corpus.

[7]Complete results are available at http://www.umiacs.umd.edu/~yzyang/sentence_generateOut.html

3.2.3.3 Sentence Generation

Given the selected sentence structure $\mathcal{T} = \{n_1, n_2, v, s, p\}$, we generate sentences using the following strategy for each component:

1) We add in appropriate determiners and cardinals: `the, an, a, CARD`, based on the content of n_1,n_2 and s. For e.g., if $n_1 = n_2$, we will use `CARD=two`, and modify the nouns to be in the plural form. When several possible choices are available, a random choice is made that depends on the object detection scores: `the` is preferred when we are confident of the detections while `an, a` is preferred otherwise.

2) We predict the most likely preposition inserted between the verbs and nouns learned from the Gigaword corpus via $P_r(p|v, n)$ during sentence generation. For example, our method will pick the preposition `at` between verb `sit` and noun `table`.

3) The verb v is converted to a form that agrees with in number with the nouns detected. The present gerund form is preferred such as `eating, drinking, walking` as it conveys that an action is being performed in the image.

4) The sentence structure is therefore of the form: `NP-VP-PP` with variations when only one object or multiple detections of the same objects are detected. A special case is when *no* objects are detected (below the predefined threshold). No verbs can be predicted as well. In this case, we simply generate a sentence that describes the *scene* only: for e.g. `This is a coast, This is a field`. Such sentences account for 20% of the entire UIUC testing dataset which are scored lower in our evaluation metrics (sec. 3.2.4.1) since they do not fully *describe* the image

content in terms of the objects and actions.

Some examples of sentences generated using this strategy are shown in Fig. 3.18.

3.2.4 Experiments

We performed several experiments to evaluate our proposed approach. The different metrics used for evaluation and comparison are also presented, followed by a discussion of the experimental results.

3.2.4.1 Sentence Generation Results

Three experiments are performed to evaluate the effectiveness of our approach. As a baseline, we simply generated $\mathcal{T}^*$ *directly* from images without using the corpus. There are two variants of this baseline where we seek to determine if listing *all* objects in the image is crucial for scene description. $\mathcal{T}_{b1}$ is a baseline that uses *all* possible objects and scene detected: $\mathcal{T}_{b1} = \{n_1, n_2, \cdots, n_m, s\}$ and our sentence will be of the form: {`Object 1, object 2 and object 3 are IN the scene.`} and we simply selected `IN` as the only admissible preposition. For the second baseline, $\mathcal{T}_{b2}$, we limit the number of objects to just any two: $\mathcal{T}_{b2} = \{n_1, n_2, s\}$ and the sentence generated will be of the form {`Object 1 and object 2 are IN the scene`}. In the second experiment, we applied the HMM strategy described above but made all transition probabilities *equiprobable*, removing the effects of the corpus, and producing a sentence structure which we denote as $\mathcal{T}^*_{eq}$. The third experiment produces the full $\mathcal{T}^*$ with transition probabilities learned from the corpus. All experiments were

performed on the 100 unseen testing images from the UIUC dataset and we used only the most likely (top) sentence generated for all evaluation.

We use two evaluation metrics as a measure of the accuracy of the generated sentences: 1) ROUGE-1 [103] precision scores and 2) *Relevance* and *Readability* of the generated sentences. ROUGE-1 is a recall based metric that is commonly used to measure the effectiveness of text summarization. In this work, the short descriptive sentence of an image can be viewed as summarizing the image content and ROUGE-1 is able to capture how well this sentence can describe the image by comparing it with the human annotated ground truth of the UIUC dataset. Due to the short sentences generated, we did not consider other ROUGE metrics (ROUGE-2, ROUGE-SU4) which captures fluency and is not an issue here.

Experiment	R_1,(length)	Relevance	Readability
Baseline 1, $\mathcal{T}^*_{b1}$	0.35,(8.2)	**2.84** $\pm$ 1.40	3.64 $\pm$ 1.20
Baseline 2, $\mathcal{T}^*_{b2}$	0.39,(6.8)	2.14 $\pm$ 1.13	3.94 $\pm$ 0.91
HMM no corpus, $\mathcal{T}^*_{eq}$	0.42,(6.5)	2.44 $\pm$ 1.25	3.88 $\pm$ 1.18
Full HMM, $\mathcal{T}^*$	**0.44**,(6.9)	2.51 $\pm$ 1.30	**4.10** $\pm$ 1.03
Human Annotation	0.68,(10.1)	4.91 $\pm$ 0.29	4.77 $\pm$ 0.42

Table 3.4: Sentence generation evaluation results with human gold standard. Human R_1 scores are averaged over the 5 sentences using a leave one out procedure. Values in bold are the top scores.

A main shortcoming of using ROUGE-1 is that the generated sentences are compared only to a finite set of human labeled ground truth which obviously does not capture all possible sentences that one can generate. In other words, ROUGE-1

does not take into account the fact that sentence generation is innately a *creative* process, and a better recall metric will be to ask humans to judge these sentences. The second evaluation metric: Relevance and Readability is therefore proposed as an empirical measure of how much the sentence: 1) conveys the image content (relevance) in terms of the objects, actions and scene predicted and 2) is grammatically correct (readability). We engaged the services of Amazon Mechanical Turks (AMT) to judge the generated sentences based on a discrete scale ranging from 1–5 (low relevance/readability to high relevance/readability). The averaged results of ROUGE-1, R_1 and mean length of the sentences with the Relevance+Readability scores for all experiments are summarized in Table 3.4. For comparison, we also asked the AMTs to judge the ground truth sentences as well.

3.2.4.2 Discussion

The results reported in Table 3.4 reveals both the strengths and some shortcomings of the approach which we will briefly discuss here. Firstly, the R_1 scores indicate that based on a purely summarization (unigram-overlap) point of view, the proposed approach of using the HMM to predict $\mathcal{T}^*$ achieves the best results compared to all other approaches with $R_1 = 0.44$. This means that our sentences are the closest in agreement with the human annotated ground truth, correctly predicting the sentence structure components. In addition sentences generated by $\mathcal{T}^*$ are also succinct: with an average length of 6.9 words per sentence. However, we are still some way off the human gold standard since we do not predict other parts-of-speech

such as adjectives and adverbs. Given this fact, our proposed approach performance is comparable to other state of the art summarization work in the literature [104].

Next, we consider the Relevance+Readability metrics based on human judges. Interestingly, the first baseline, $\mathcal{T}_{b1}^{*}$ is considered the most relevant description of the image and the least readable at the same time. This is most likely due to the fact that this recall oriented strategy will almost certainly describe some objects but the lack of any verb description; and longer sentences that average 8.2 words per sentence, makes it less readable. It is also possible that humans tend to penalize less irrelevant objects compared to missing objects, and further evaluations are necessary to confirm this. Since $\mathcal{T}_{b2}^{*}$ is limited to two objects just like the proposed HMM, it is a more suitable baseline for comparison. Clearly, the results show that adding the HMM to predict the optimal sentence structure increases the relevance of the produced sentence. Finally, in terms of *readability*, $\mathcal{T}^{*}$ generates the most readable sentences, and this is achieved by leveraging on the corpus to guide our predictions of the most *reasonable* nouns, verbs, scenes and prepositions that agree with the detections in the image.

Chapter 4: "Can't Make an Omelette without Breaking Eggs": Detection of Manipulation Action Consequences

4.1 Introduction

Visual recognition is the process through which intelligent agents associate a visual observation to a concept from their memory. In most cases, the concept either corresponds to a term in natural language, or an explicit definition in natural language. Most research in Computer Vision has focused on two concepts: objects and actions; humans, faces and scenes can be regarded as special cases of objects. Object and action recognition are indeed crucial since they are the fundamental building blocks for an intelligent agent to semantically understand its observations.

When it comes to understanding actions of manipulation, the movement of the body (especially the hands) is not a very good characteristic feature. There is great variability in the way humans carry out such actions. It has been realized that such actions are better described by involving a number of quantities. Besides the motion trajectories, the objects involved, the hand pose, and the spatial relations between the body and the objects under influence, provide information about the action. In this work we want to bring attention to another concept, the action consequence.

It describes the transformation of the object during the manipulation. For example during a CUT or a SPLIT action an object is divided into segments, during a GLUE or a MERGE action two objects are combined into one, etc.

The recognition and understanding of human manipulation actions recently has attracted the attention of Computer Vision and Robotics researchers because of their critical role in human behavior analysis. Moreover, they naturally relate to both, the movement involved in the action and the objects. However, so far researchers have not considered that the most crucial cue in describing manipulation actions is actually not the movement nor the specific object under influence, but the object centric action consequence. We can come up with examples, where two actions involve the same tool and same object under influence, and the motions of the hands are similar, for example in "cutting a piece of meat" vs. "poking a hole into the meat". Their consequences are different. In such cases, the action consequence is the key in differentiating the actions. Thus, to fully understand manipulation actions, the intelligent system should be able to determine the object centric consequences.

Few researchers have addressed the problem of action consequences due to the difficulties involved. The main challenge comes from the monitoring process, which calls for the ability to continuously check the topological and appearance changes of the object-under-manipulation. Previous studies of visual tracking have considered challenging situations, such as non-rigid objects [3], adaptive appearance model [105], and tracking of multiple objects with occlusions [106], but none can deal with the difficulties involved in detecting the possible changes on objects during

manipulation. In this chapter, for the first time, a system is implemented to conquer these difficulties and eventually achieve robust action consequence detection.

4.2 Why Consequences and Fundamental Types

Recognizing human actions has been an active research area in Computer Vision [107]. Several excellent surveys on the topic of visual recognition are available ([108], [109]). Most work on visual action analysis has been devoted to the study of movement and change of posture, such as walking, running etc. The dominant approaches to the recognition of single actions compute as descriptors statistics of spatio-temporal interest points ([110], [111]) and flow in video volumes, or represent short actions by stacks of silhouettes ([112], [113]). Approaches to more complex, longer actions employ parametric approaches, such as Hidden Markov Models [114], Linear Dynamical Systems [115] or Non-linear Dynamical Systems [116], which are defined on extracted features. There are a few recent studies on human manipulation actions ([117], [11], [118]), but they do not consider action consequences for the interpretation of manipulation actions. Works like [119] emphasize the role of object perception in action or pose recognition, but they focus on object labels, not object-centric consequences.

How do humans understand, recognize, and even replicate manipulation actions? Psychological studies on human manners ([120] etc.) have pointed out the importance of manipulation action consequences for both understanding human cognition and intelligent system research. **Actions, by their very nature, are goal**

oriented. When we perform an action, we always have a goal in mind, and the goal affects the action. Similarly, when we try to recognize an action, we also keep a goal in mind. The close relation between the movement during the action and goal is reflected also in language. For example, the word "CUT" denotes both the action in which hands move up and down or in and out with sharp bladed tools, and the consequence of the action, namely that the object is separated. Very often, we can recognize an action purely by the goal satisfaction, and even neglect the motion or the tools used. For example, we may observe a human carry out movement with a knife, that is "up and down", but if the object remains as one whole, we won't draw the conclusion that a "CUT" action has been performed. Only when the goal of the recognition process, here "DIVIDE", is detected, the goal satisfaction is reached and a "CUT" action is confirmed. An intelligent system should have the ability to detect the consequences of manipulation actions, in order to check the goal of actions.

In addition, experiments conducted in neuronscience [9] show that a monkey's mirror neuron system fires when a hand/object interaction is observed, and it will not fire when a similar movement is observed without hand/object interaction. Recent experiments [10] further showed that the mirror neuron regions responding to the sight of actions responded more during the observation of goal-directed actions than similar movements not directed at goals. These evidences support the idea of goal matching, as well as the crucial role of action consequence in the understanding of manipulation actions.

Taking an object-centric point of view, manipulation actions can be classified

into six categories according how the object is transformed during the manipulation, or in other words what consequence the action has on the object. These categories are: DIVIDE, ASSEMBLE, CREATE, CONSUME, TRANSFER, and DEFORM. Each of these categories is denoted by a term that has a clear semantic meaning in natural language given as follows:

- DIVIDE: one object breaks into two objects, or two attached objects break the attachment;
- ASSEMBLE: two objects merge into one object, or two objects build an attachment between them;
- CREATE: an object is brought to, or emerges in the visual space;
- CONSUME: an object disappears from the visual space;
- TRANSFER: an object is moved from one location to another location;
- DEFORM: an object has an appearance change.

To describe these action categories we need a formalism. We use the visual semantic graph (VSG) inspired from the work of Aksoy et. al [14]. This formalism takes as input computed object segments, their spatial relationship, and temporal relationship over consecutive frames. To provide the symbols for the VSG, an active monitoring process (discussed in sec. 4.4) is required for the purpose of (1) tracking the object to obtain temporal correspondence, and (2) segmenting the object to obtain its topological structure and appearance model. This active monitoring

(consisting of segmentation and tracking) is related to studies on active segmentation [121], and stochastic tracking ([122] etc.).

4.3 Visual Semantic Graph (VSG)

To define object-centric action consequences, a graph representation is used. Every frame in the video is described by a Visual Semantic Graph (VSG), which is represented by an undirected graph $G(V, E, P)$. The vertex set $|V|$ represents the set of semantically meaningful segments, the edge set $|E|$ represents the spatial relations between any of the two segments. Two segments are connected when they share parts of their borders, or when one of the segments is contained in the other. If two nodes $v_1, v_2 \in V$ are connected, $E(v_1, v_2) = 1$, otherwise, $E(v_1, v_2) = 0$. In addition, every node $v \in V$ is associated with a set of properties $P(v)$, that describes the attributes of the segment. This set of properties provides additional information to discriminate the different categories, and in principle many properties are possible. Here we use location, shape, and color.

We need to compute the changes of the object over time. In our formulation this is expressed as the change in the VSG. At any time instance t, we consider two consecutive VSGs, the VSG at time $t-1$, denoted as $G_a(V_a, E_a, P_a)$ and the VSG at time t, denoted as $G_z(V_z, E_z, P_z)$. We then define the following four consequences, where $\rightarrow$ is used to denote the temporal correspondence between two vertices, $\nrightarrow$ is used to denote no correspondence:

- DIVIDE: $\{\exists v_1 \in V_a; v_2, v_3 \in V_z | v_1 \rightarrow v_2, v_1 \rightarrow v_3)\}$ *or* $\{\exists v_1, v_2 \in V_a; v_3, v_4 \in$

$V_z | E_a(v_1, v_2) = 1, E_z(v_3, v_4) = 0, v_1 \to v_3, v_2 \to v_4\}$ **Condition (1)**

- ASSEMBLE: $\{\exists v_1, v_2 \in V_a; v_3 \in V_z | v_1 \to v_3, v_2 \to v_3\}$ *or* $\{\exists v_1, v_2 \in V_a; v_3, v_4 \in V_z | E_a(v_1, v_2) = 0, E_z(v_3, v_4) = 1, v_1 \to v_3, v_2 \to v_4\}$ **Condition (2)**

- CREATE:$\{\forall v \in V_a; \exists v_1 \in V_z | v \nrightarrow v_1\}$ **Condition (3)**

- CONSUME:$\{\forall v \in V_z; \exists v_1 \in V_a | v \nrightarrow v_1\}$ **Condition(4)**

While the above actions can be defined purely on the basis of topological changes, there are no such changes for TRANSFER and DEFORM. Therefore, we have to define them through changes in property. In the following definitions, P^L represents properties of location, and P^S represents properties of appearance (shape, color, etc.).

- TRANSFER:$\{\exists v_1 \in V_a; v_2 \in V_z | P_a^L(v_1) \neq P_z^L(v_2)\}$ **Condition (5)**

- DEFORM: $\{\exists v_1 \in V_a; v_2 \in V_z | P_a^S(v_1) \neq P_z^S(v_2)\}$ **Condition (6)**

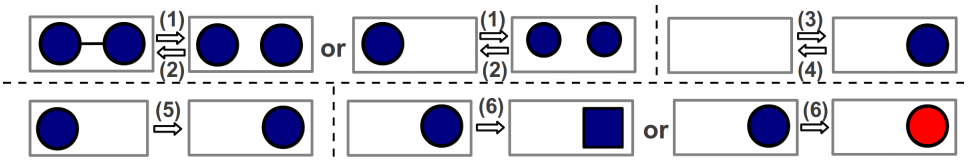


Figure 4.1: Graphical illustration of the changes for **Condition (1-6)**.

A graphical illustration for **Condition (1-6)** is shown in Fig. 4.1. Sec. 4.4 describes how we find the primitives used in the graph. A new active segmentation and tracking method is introduced to 1) find correspondences ($\to$) between V_a and V_z; 2) monitor location property P^L and appearance property P^S in the VSG.

The procedure for computing action consequences, first decides on whether there is a topological change between G_a and G_z. If yes, the system checks whether **Condition (1)** to **Condition (4)** are fulfilled and returns the corresponding consequence. If no, the system then checks whether **Condition (5)** or **Condition (6)** is fulfilled. If both of them are not met, no consequence is detected.

4.4 Active Segmentation and Tracking

Previously, researchers have treated segmentation and tracking as two different problems. Here we propose a new method combining the two tasks to obtain the information necessary to monitor the objects under influence. Our methods combines stochastic tracking [122] with a fixation based active segmentation [121]. The tracking module provides a number of tracked points. The locations of these points are used to define an area of interest and a fixation point for the segmentation, and the color in their immediate surroundings are used in the data term of the segmentation module. The segmentation module segments the object, and based on the segmentation, updates the appearance model for the tracker. Fig 4.2 illustrates the method over time, which is a dynamically closed-loop process. We next describe the attention based segmentation (sec. 4.4.1 - 4.4.4), and then the segmentation guided tracking (sec. 4.4.5).

The proposed method meets two challenging requirements, necessary to detect action consequences: 1) the system is able to track and segment objects when the shape or color (appearance) changes; 2) the system is also able to track and segment

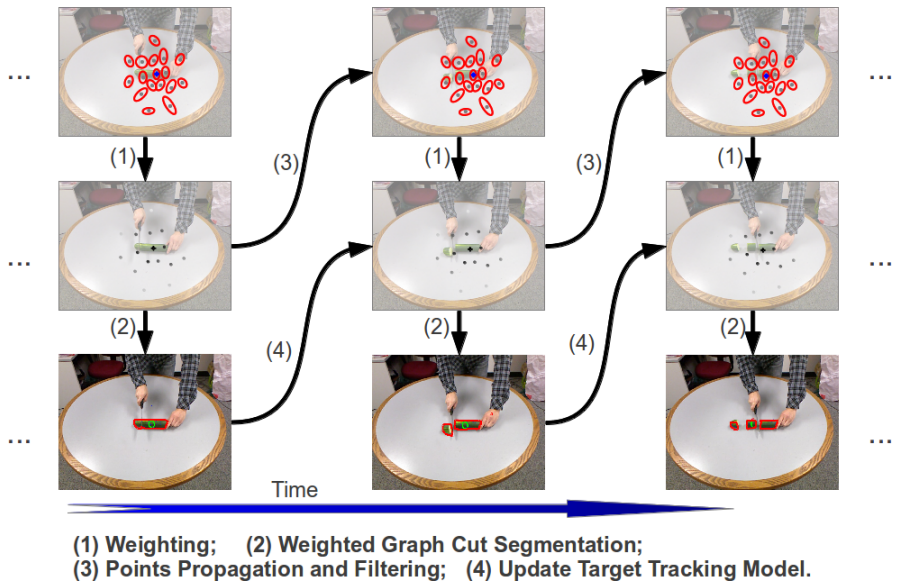


Figure 4.2: Flow chart of the proposed active segmentation and tracking method for object monitoring.

objects when they are divided into pieces. Experiments in sec. 4.5.1 show that our method can handle these requirements, while systems implementing independently tracking and segmentation cannot.

4.4.1 The Attention Field

The idea underlying our approach is, that first a process of visual attention selects an area of interest. Segmentation then is considered the process that separates the area selected by visual attention from background by finding closed contours that best separate the regions. The minimization uses a color model for the data term and edges in the regularization term. To achieve a minimization that is very robust to the length of the boundary, edges are weighted with their distance from the fixation center.

Visual attention, the process of driving an agent's attention to a certain area, is based on both bottom-up processes defined on low level visual features, and top-down processes influenced by the agent's previous experience [123]. Inspired by the work of Yang et al. [124], instead of using a single fixation point in the active segmentation [121], here we use a weighted sample set $S = \{(s^{(n)}, \pi^{(n)}) | n = 1 \dots N\}$ to represent the attention field around the fixation point ($N = 500$ in practice). Each sample consists of an element s from the set of tracked points and a corresponding discrete weight π where $\sum_{n=1}^{N} \pi^{(n)} = 1$.

Generally, any appearance model can be used to represent the local visual information around each point. We choose to use a color histogram with a dynamic sampling area defined by an ellipse. To compute the color distribution, every point is represented by an ellipse, $s = \{x, y, \dot{x}, \dot{y}, H_x, H_y, \dot{H}_x, \dot{H}_y, \}$ where x and y denote the location, $\dot{x}$ and $\dot{y}$ the motion, H_x, H_y the length of the half axes, and $\dot{H}_x$, $\dot{H}_y$ the changes in the axes.

4.4.2 Color Distribution Model

To make the color model invariant to various textures or patterns, a color distribution model is used. A function $h(x_i)$ is defined to create a color histogram, which assigns one of the m-bins to a giving color at location x_i. To make the algorithm less sensitive to lighting conditions, the HSV color space is used with less sensitivity in the V channel ($8 \times 8 \times 4$ bins). The color distribution for each fixation

point $s^{(n)}$ is computed as:

$$p(s^{(n)})^{(u)} = \gamma \sum_{i=1}^{I} k(||y - x_i||)\delta[h(x_i) - u], \tag{4.1}$$

where $u = 1 \ldots m$, $\delta(.)$ is the Kronecker delta function, and γ is the normalization term $\gamma = \frac{1}{\sum_{i=1}^{I} k(||y-x_i||)}$. $k(.)$ is a weighting function designed from the intuition that not all pixels in the sampling region are equally important for describing the color model. Specifically, pixels that are farther away from the point are assigned smaller weights, $k(r) = \begin{cases} 1 - r^2 & \text{if } r < a \\ 0 & \text{otherwise} \end{cases}$, where the parameter a is used to adapt the size of the region, and r is the distance from the fixation point. By applying the weighting function, we increase the robustness of the color distribution by weakening the influence from boundary pixels, which possibly belong to the background or are occluded.

4.4.3 Weights of the Tracked Point Set

In the following weighted graph cut approach, every sample is weighted by comparing its color distribution with the one of the fixation point. Initially a fixation point is selected, later the fixation point is computed as the center of the tracked point set. Let's call the distribution at the fixation point q, and the histogram of the n^{th} tracked point, $p(s^{(n)})$. In assigning weights $\pi^{(n)}$ to the tracked points we want to favor points whose color distribution is similar to the fixation point. We use the Bhattacharyya coefficient $\rho[p, q] = \sum_{u=1}^{m} \sqrt{p^{(u)} q^{(u)}}$ with m the number of bins to weigh points by a Gaussian with variance σ ($\sigma = 0.2$ in practice) and define $\pi^{(n)}$

as:

$$\pi^{(n)} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{d^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1-\rho[p(s^{(n)}),q]}{2\sigma^2}}. \tag{4.2}$$

4.4.4 Weighted Graph Cut

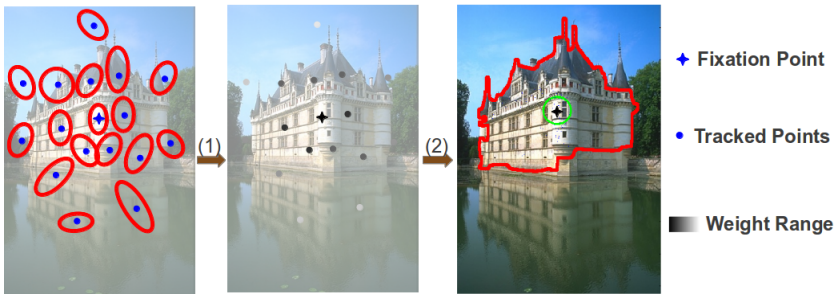
The segmentation is formulated as a minimization that is solved using graph cut. The unary terms are defined on the tracked points on the basis of their color, and the binary terms are defined on all points on the basis of edge information. To obtain the edge information, in each frame, we compute a probabilistic edge map I_E using the Canny edge detector. Consider every pixel $x \in X$ in this edge map as a node in a graph. Denoting the set of all the edges connecting neighboring nodes in the graph as Ω, and using the label set $l = 0, 1$ to indicate whether a pixel x is "inside" ($l_x = 0$) or "outside" ($l_x = 1$), we need to find a labeling $f(X) \longmapsto l$, that minimizes the energy function:

$$Q(f) = \sum_{x \in X} U_x(l_x) + \lambda \sum_{(x,y)\in\Omega} V_{x,y}\delta(l_x, l_y). \tag{4.3}$$

$V_{x,y}$ is the cost of assigning different labels to neighboring pixels x and y, which we defines as $V_{x,y} = e^{-\eta I_{E,xy}} + k$, with $\delta(l_x, l_y) = \begin{cases} 1 & \text{if } l_x \neq l_y \\ 0 & \text{otherwise} \end{cases}$, $\lambda = 1, \eta = 1000, k = 10^{-16}$, $I_{E,xy} = (I_E(x)/R_x + I_E(y)/R_y)/2$, R_x, R_y are the euclidean distances between the x, y and the center of the tracked point set S_t. We use them as weights to make the segmentation robust to the length of the contours.

$U_x(l_x)$ is the cost of assigning label l_x to pixel x. In our system, we have a set of points S_t, and for each sample $s^{(n)}$, there is a weight $\pi^{(n)}$. The weight itself indicates

the likelihood that the area around that fixation point belongs to the "inside" of the object. It becomes straightforward to assign weights $\pi^{(n)}$ to the pixel $s^{(n)}$, which are tracked points as follows: $U_x(l_x) = \begin{cases} N\pi^{(n)} & \text{if } l_x = 1 \\ 0 & \text{otherwise} \end{cases}$. We assume that pixels on the boundary of a frame are "outside" of the object, and assign to them a large weight $W = 10^{10}$: $U_x(l_x) = \begin{cases} W & \text{if } l_x = 0 \\ 0 & \text{otherwise} \end{cases}$. Using this formulation, we run a graph cut algorithm [125] on each frame. Fig. 4.3(a) illustrates the procedure on a texture-rich natural image from the Berkeley segmentation dataset [126].



(a)

(b)

Figure 4.3: **Upper**: (1) Sampling of tracked points sampling and filtering; (2) Weighted graph cut. **Lower**: Segmentation with different initial fixations. Green Cross: initial fixation.

Two critical limitations of the previous active segmentation method [121] in practice are: 1) the segmentation performance largely varies under different initial fixation points; 2) the segmentation performance also is strongly affected by texture

edges, which often leads to a segmentation of object parts. Fig. 4.3(b) shows that our proposed segmentation method is robust to the choice of initial fixation point, and only weakly affected by texture edges.

4.4.5 Active Tracking

At the very beginning of the monitoring process, a Gaussian sampling with mean at the initial fixation point and variances σ_x, σ_y is used to generate the initial point set S_0. When a new frame comes in, the point set is propagated through a stochastic tracking paradigm:

$$s_t = As_{t-1} + w_{t-1}, \tag{4.4}$$

where A denotes the deterministic, and w_{t-1} the stochastic component. In our implementation, we have considered a first order model for A, which assumes that the object is moving with constant velocity. The reader is referred to [127] for details. The complete algorithm is given in Algorithm 1

4.4.6 Incorporating Depth and Optical Flow

It is easy to extend our algorithm to incorporate depth (for example from Kinect) or image motion flow information. Depth information can be used in a straightforward way during two crucial steps. 1) As described in sec. 4.4.2, we can add in depth information as another channel in the distribution model. In preliminary experiments we used 8 bins for the depth, to obtain in RGBD space a model with $8 \times 8 \times 4 \times 8$ bins. 2) Depth can be used to achieve cleaner edge maps,

Algorithm 1 Active tracking and segmentation

Require: Given the tracked point set S_{t-1} and the target model q_{t-1}, perform the following steps:

1. **SELECT** N samples from the set S_{t-1} with probability $\pi_{t-1}^{(n)}$. Fixation points with a high weight may be chosen several times, leading to identical copies, while others with relatively low weights may not be chosen at all. Denote the resulting set as S'_{t-1};

2. **PROPAGATE** each sample from S'_{t-1} by a linear stochastic differential eq. 4.4. Denote the new set as S_t

3. **OBSERVE** the color distributions for each sample of S_t using eq. 4.1. Weigh each sample using eq. 4.2.

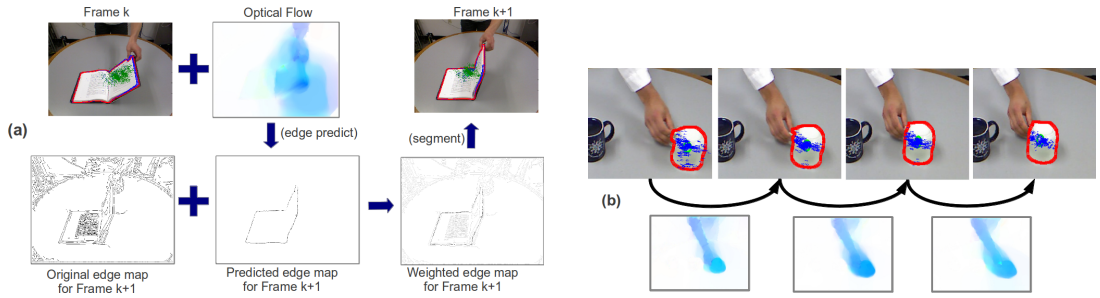
4. **SEGMENTATION** using the weighted sample set. Apply the weighted graph cut algorithm described in sec. 4.4.4. and get the segmented object area M.

5. **UPDATE** the target distribution q_{t-1} by the area M to achieve the new target distribution q_t.

I_E, in the segmentation step 4.4.4.

Optical flow can be incorporated to provide cues for the system to predict the movement of edges to be used for the segmentation step in the next iteration, and the movement of the points in the tracking step. We performed some experiments using the optical flow estimation method proposed by Brox [128] and the improved implementation by Liu [129].

Optical flow was used in the segmentation by first predicting the contour of the object in the next frame, and then fusing it with the next frame's edge map. Fig. 4.4(a) shows an example of an edge map improved by optical flow. Optical flow was incorporated into tracking by replacing the first order velocity components for each tracked point in matrix A (eq. 4.4) by its flow component. Fig. 4.4(b) shows that the optical flow drives the tracked points to move along the flow vectors into the next frame.



(a) (b)

Figure 4.4: (a): Incorporating optical flow into segmentation. (b): Incorporating optical flow into tracking.

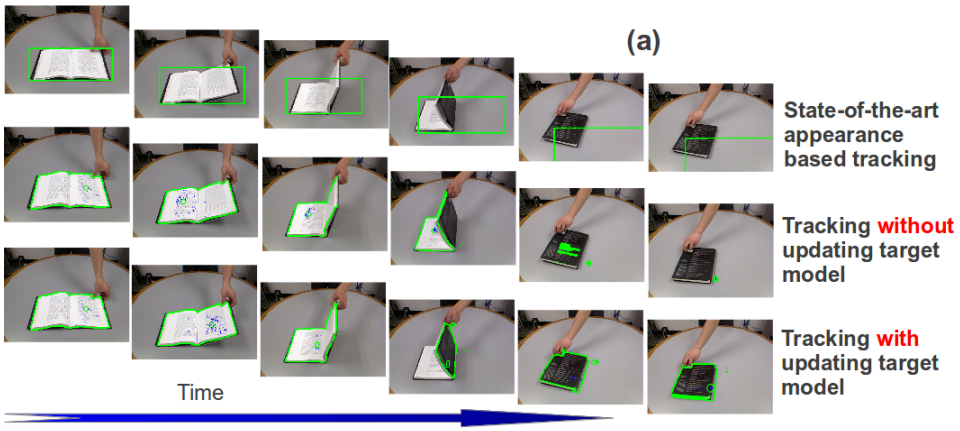
4.5 Experiments

4.5.1 Deformation and Division

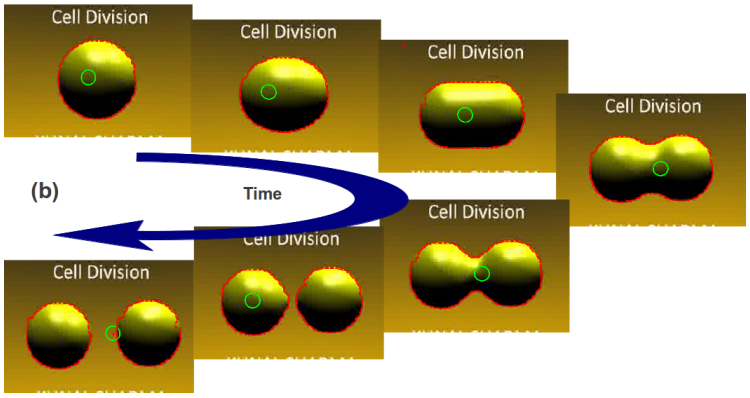
To show that our method can segment challenging cases, we first demonstrate its performance for the case of deforming and dividing objects. Fig. 4.5(a) shows results for a sequence with the main object deforming, and Fig. 4.5(b) for a synthetic sequence with the main object dividing. The ability to handle deformations comes from the updating of the target model using the segmentation of previous frames. The ability to handle division comes from the tracked point set that is used to represent the attention field (sec. 4.4.1), which guides the weighted graph cut algorithm (sec. 4.4.4).

4.5.2 The MAC 1.0 Dataset

To quantitatively test our method, we collected a dataset of several RGB+Depth image sequences of humans performing different manipulation actions. In addition, several sequences from other publicly available datasets ([14], [130] and [131]) were included to increase the variability and make it more challenging. Since the two action consequences CREATE and CONSUME (sec.4.2) relate to the existence of the object and would require a higher level attention mechanism, which is out of this chapter's scope, we did not consider them. For the other four consequences, six sequences were collected each to make the first Manipulation Action Consequence



(a)



(b)

Figure 4.5: (a): Deformation Invariance: upper: state-of-the-art appearance based tracking [6]; middle: tracking without updating target model; lower: updating target model. (b): Division Invariance: synthetic cell division sequence.

(MAC 1.0) dataset.[1].

4.5.3 Consequence Detection on MAC 1.0

We first evaluated the method's ability in detecting the various consequences. Consequences happen in an event based manner. In our description, a consequence is detected using the VSG graph at a point in time, if between two consecutive image frames one of the conditions listed in sec. 4.3 is met. For example, a consequence is detected for the case of DIVIDE, when one segment becomes two segments in the next frame (Fig. 4.6), or for the case of DEFORM, when one appearance model changes to another (Fig. 4.8). We obtained ground truth by asking people not familiar with the purpose to label the sequences in MAC 1.0.

Fig. 4.6, 4.7, 4.8 show typical example active segmentation and tracking, the VSG graph, and the corresponding measures used to identify the different action consequences, as well as the final detection result along the time-line are illustrated. Specifically, for DIVIDE we monitor the change in the number of segments, for ASSEMBLE we monitor the minimum Euclidean distance between the contours of segments, for DEFORM we monitor the change of appearance (color histogram and shape context [132]) of the segment, and for TRANSFER we monitor the velocity of the object. Each of the measurements is normalized to the range of $[0, 1]$ for the ROC analysis. The detection is evaluated, by counting the correct detections over the sequence. For example, for the case of DIVIDE, at any point in time we have either the detection, "not divided" or "divided". For the case of ASSEMBLE

[1]The dataset is available at www.umiacs.umd.edu/~yzyang.

, we have either the detection "two parts assembled" or "nothing assembled", and for DEFORM, we have either "deformed" or "nor deformed". The ROC curves obtained are shown in Fig. 4.9. The graphs indicate that our method is able to correctly detect most of the consequences. Several failures point out the limitations of our method as well. For example, for the PAPER-JHU sequence the method has errors in detecting DIVIDE, because the part that was cut out, connects visually with the rest of the chapter. For the CLOSE-BOTTLE sequence our method fails for ASSEMBLE because the small bottle cap is occluded by the hand. However, our method detects that an ASSEMBLE event happened after the hand move away.

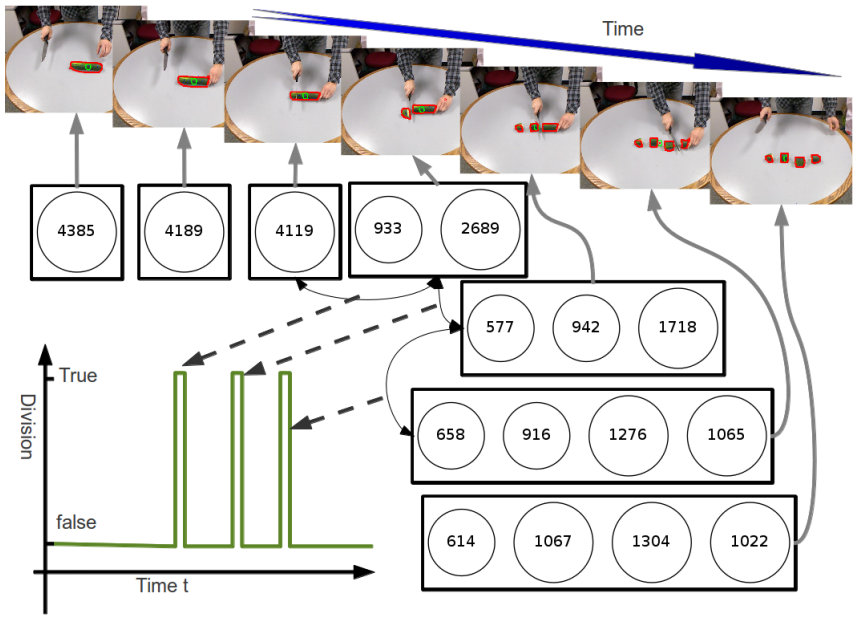


Figure 4.6: "Division" detection on "cut cucumber" sequence. Upper row: Original sequence with segmentation and tracking; Middle and lower right: VSG representations; Lower left: Division consequences detection.

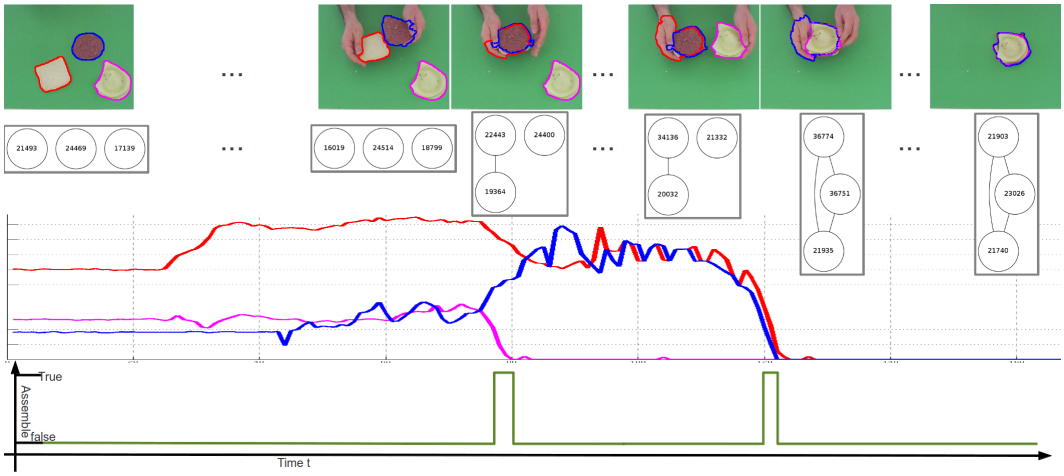


Figure 4.7: "Assemble" detection on "make sandwich 1" sequence; 1st row: Original sequence with segmentation and tracking; 2nd row: VSG representation; 3rd row: Distance between each two segments (red line: bread and cheese, magenta line: bread and meat, blue line: cheese and meat; 4th row: Assemble consequence detection.

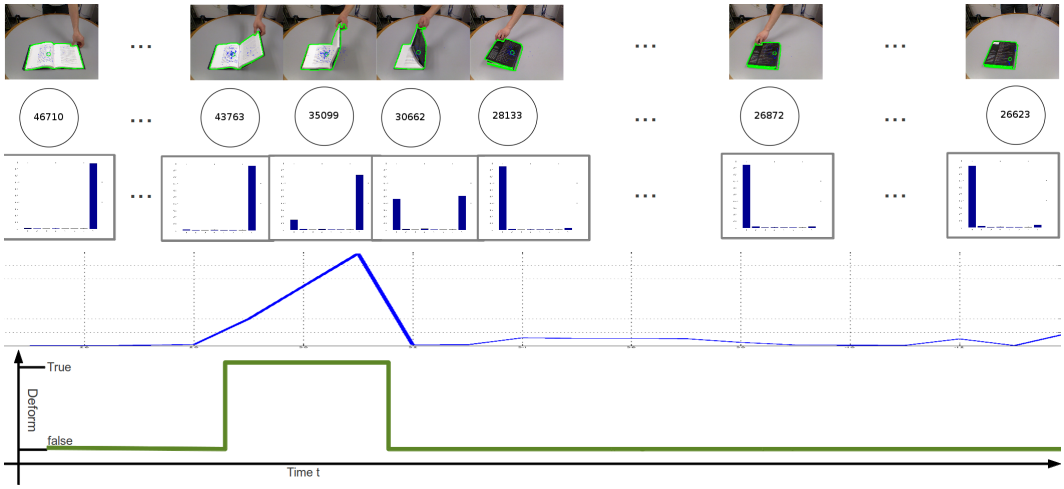


Figure 4.8: "Deformation" detection on "close book 1" sequence; 1st row: Original sequence with segmentation and tracking; 2nd row: VSG representation; 3rd row: appearance description (here color histogram) of each segment; 4th row: measurement of appearance change; 5th row: Deformation consequence detection.

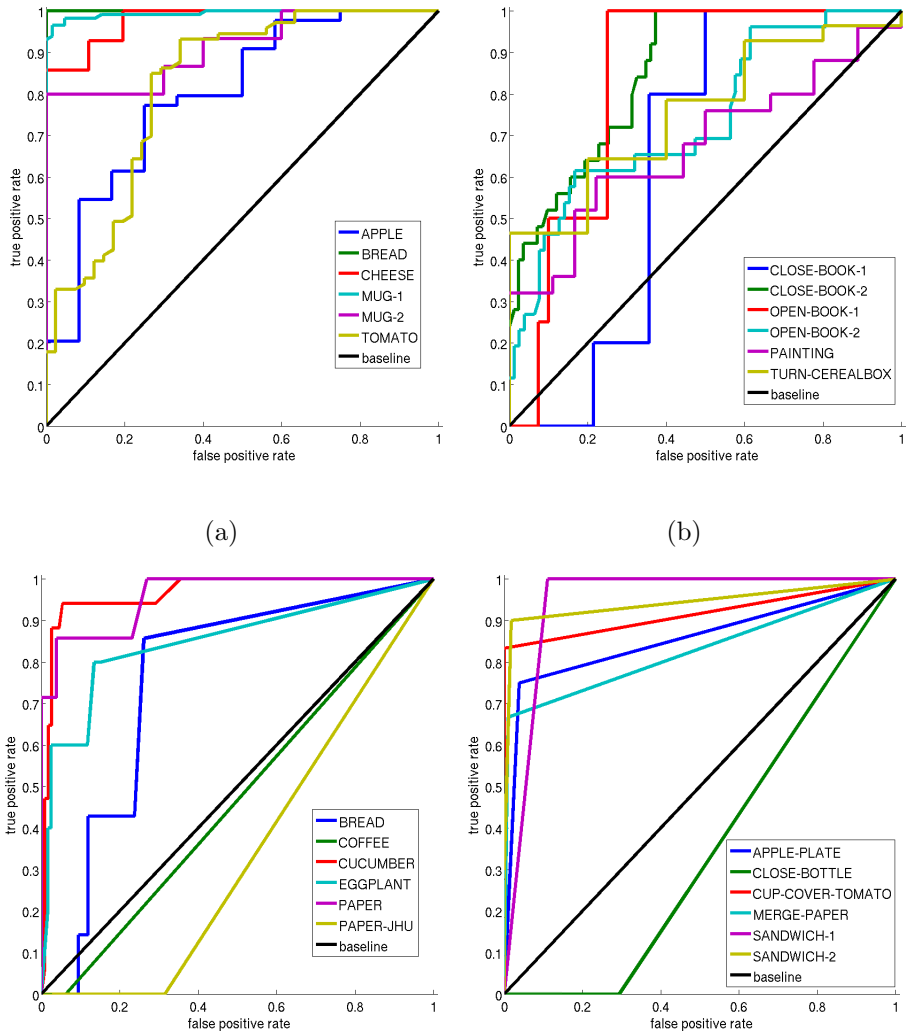


Figure 4.9: ROC curve of each sequence by categories: (a) TRANSFER, (b) DEFORM, (c) DIVIDE, and (d) ASSEMBLE.

4.5.4 Video Classification on MAC 1.0

We also evaluated our method on the problem of classification, although the problem of consequence detection is quite different from the problem of video classification. We compared our method with the state-of-the-art STIP + Bag of Words + classification (SVM or Naive Bayes). The STIP features for each video in the MAC

1.0 dataset were computed using the method described in [110]. For classification we used a bag of words + SVM and a Naive Bayes method. The dictionary codebook size was 1000, and a polynomial kernel SVM with a leave-one-out cross validation setting was used. Fig. 4.10 shows that our method dramatically outperforms the typical STIP + Bag of Words + SVM and the Naive Bayes learning methods. However, this does not come as a surprise. The different video sequences in an action consequence class contain different objects and different actions and thus different visual features, and are therefore not well suited for standard classification. On the other hand, our method has been specifically designed for the detection of manipulation action consequences all the way from low-level signal processing through the mid-level semantic representation to high-level reasoning. Moreover, different from a learning based method, it does not rely on training data. After all, the method stems from the insight of manipulation action consequences.

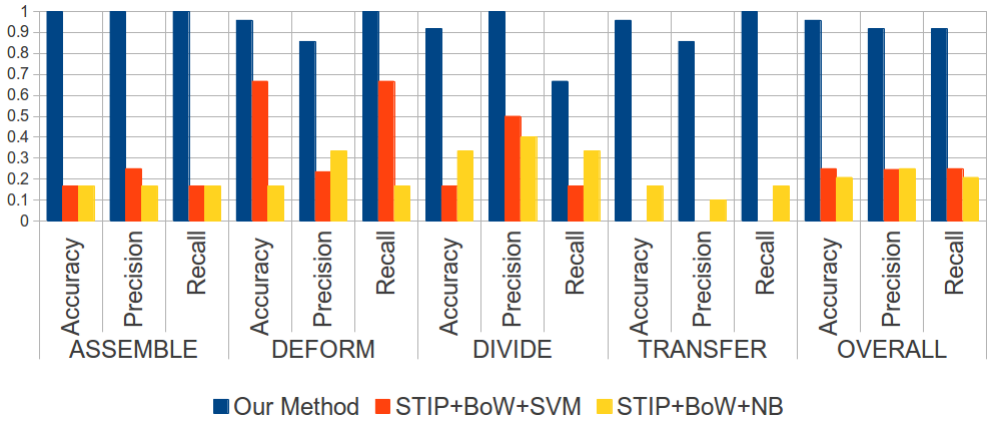


Figure 4.10: Video classification performance comparison.

Chapter 5: The Syntax: A Syntactical Grammar for Understanding Human Manipulation Actions

5.1 Introduction

Cognitive systems that interact with humans must be able to interpret actions. Here we are concerned with manipulation actions. These are actions performed by agents (humans or robots) on objects, which result in some physical change of the object. There has been much work recently on action recognition, with most studies considering short lived actions, where the beginning and end of the sequence is defined. Most efforts have focused on two problems of great interest to the study of perception: the recognition of movements and the recognition of associated objects. However, the more complex an action, the less reliable individual perceptual events are for the characterization of actions. Thus, the problem of interpreting manipulation actions involves many more challenges than simply recognizing movements and objects, due to the many ways that humans can perform them.

Since perceptual events do not suffice, how do we determine the beginning and end of action segments, and how do we combine the individual segments into longer ones corresponding to a manipulation action? An essential component in the

description of manipulations is the underlying goal. The goal of a manipulation action is the physical change induced on the object. To accomplish it, the hands must perform a sequence of sub actions on the object, such as when the hand grasps or releases the object, or when the hand changes the grasp type during a movement. Centered around this idea, we develop a grammatical formalism for parsing and interpreting action sequences, and we also develop vision modules to obtain from dynamic imagery the symbolic information used in the grammatical structure.

Our formalism for describing manipulation actions uses a structure similar to natural language. What do we gain from this formal description of action? This is equal to asking what one gains from a formal description of language. Chomsky's contribution to language was the formal description of language through his generative and transformational grammar [22]. This revolutionized language research, opened up new roads for its computational analysis and provided researchers with common, generative language structures and syntactic operations on which language analysis tools were built. Similarly, a grammar for action provides a common framework of the syntax and semantics of action, so that basic tools for action understanding can be built. Researchers can then use these tools when developing action interpretation systems, without having to start from scratch.

The input into our system for interpreting manipulation actions is perceptual data, specifically sequences of images and depth maps. Therefore, a crucial part of our system is the vision process, which obtains atomic symbols from perceptual data. In Section 5.3, we introduce an integrated vision system with attention, segmentation, hand tracking, grasp classification, and action recognition. The vision

processes produce a set of symbols: the "Subject", "Action", and "Object" triplets, which serve as input to the reasoning module. At the core of our reasoning module is the *manipulation action context-free grammar* (MACFG). This grammar comes with a set of generative rules and a set of parsing algorithms. The parsing algorithms use two main operations – "construction" and "destruction" – to dynamically parse a sequence of tree (or forest) structures made up from the symbols provided by the vision module. The sequence of semantic tree structures could then be used by the cognitive system to perform reasoning and prediction. Figure 5.1 shows the flow chart of our cognitive system.

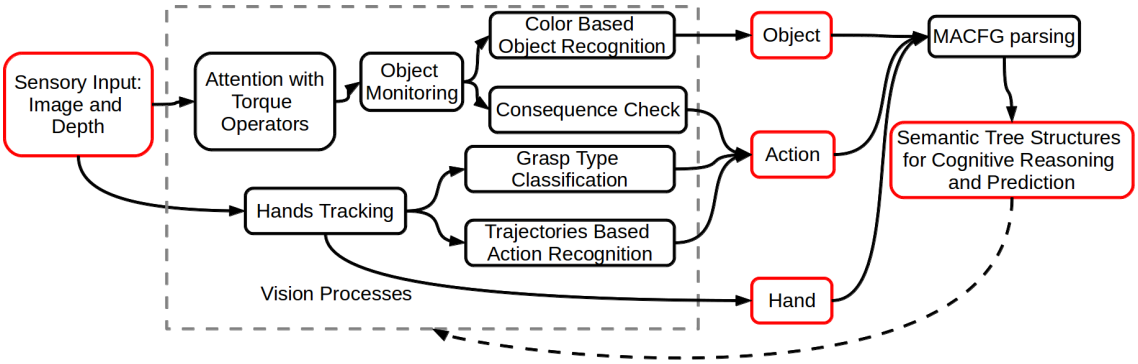


Figure 5.1: Overview of the manipulation action understanding system, including feedback loops within and between some of the modules. The feedback is denoted by the dotted arrow.

5.2 Related Work

The problem of human activity recognition and understanding has attracted considerable interest in computer vision in recent years. Both visual recognition

methods and nonvisual methods using motion capture systems [107, 133] have been used. [108] [109], and [134] provide surveys of the former. There are many applications for this work in areas such as human computer interaction, biometrics, and video surveillance. Most visual recognition methods learn the visual signature of each action from many spatio-temporal feature points (e.g, [110, 111, 135, 136]). Work has focused on recognizing single human actions like walking, jumping, or running ([113, 137]). Approaches to more complex actions have employed parametric models such as hidden Markov models [114] to learn the transitions between image frames (e.g, [14, 115, 116, 138]).

The problem of understanding manipulation actions is also of great interest in robotics, which focuses on execution. Much work has been devoted to learning from demonstration [139], such as the problem of a robot with hands learning to manipulate objects by mapping the trajectory observed for people performing the action to the robot body. These approaches have emphasized signal to signal mapping and lack the ability to generalize. More recently, within the domain of robot control research, [140] have used temporal logic for hybrid controller design, and later [141] suggested a grammatical formal system to represent and verify robot control policies. [142] and [143] created a library of manipulation actions through semantic object-action relations obtained from visual observation.

There have also been many syntactic approaches to human activity recognition that use the concept of context-free grammars, because they provide a sound theoretical basis for modeling structured processes. [144] used a grammar to recognize disassembly tasks that contain hand manipulations. [145] used the context-free

grammar formalism to recognize composite human activities and multi-person interactions. It was a two-level hierarchical approach in which the lower level was composed of hidden Markov models and Bayesian networks while the higher-level interactions were modeled by CFGs. More recent methods have used stochastic grammars to deal with errors from low-level processes such as tracking [146, 147]. This work showed that grammar-based approaches can be practical in activity recognition systems and provided insight for understanding human manipulation actions. However, as mentioned, thinking about manipulation actions solely from the viewpoint of recognition has obvious limitations. In this work, we adopt principles from CFG-based activity recognition systems, with extensions to a minimalist grammar that accommodates not only the hierarchical structure of human activity, but also human-tool-object interactions. This approach lets the system serve as the core parsing engine for manipulation action interpretation.

[148] suggested that a minimalist generative structure, similar to the one in human language, also exists for action understanding. [17] introduced a minimalist grammar of action, which defines the set of terminals, features, non-terminals and production rules for the grammar in the sensorimotor domain. However, this was a purely theoretical description. The first implementation used only objects as sensory symbols [18]. Then [31] proposed a minimalist set of atomic symbols to describe the movements in manipulation actions. In the field of natural language understanding, which traces back to the 1960s, [149] proposed the Conceptual Dependency theory [149] to represent content inferred from natural language input. In this theory, a sentence is represented as a series of diagrams representing both mental and phys-

ical actions that involve agents and objects. These actions are built from a set of primitive acts, which include atomic symbols like GRASP and MOVE. [150] have also discussed the relationship between languages and motion control.

Here we extend the minimalist action grammar of [17] to dynamically parse the observations by providing a set of operations based on a set of context-free grammar rules. Then we provide a set of biologically inspired visual processes that compute from the low-level signals the symbols used as input to the grammar in the form of (Subject, Action, Object). By integrating the perception modules with the reasoning module, we obtain a cognitive system for human manipulation action understanding.

5.3 A Cognitive System For Understanding Human Manipulation Actions

In this section, we first describe the Manipulation Action Context-Free Grammar and the parsing algorithms based on it. Then we discuss the vision methods: the attention mechanism, the hand tracking and action recognition, the object monitoring and recognition, and the action consequence classification.

5.3.1 A Context-Free Manipulation Action Grammar

Our system includes vision modules that generate a sequence of "Subject" "Action" "Patient" triplets from the visual data, a reasoning module that takes in the sequence of triplets and builds them into semantic tree structures. The binary

tree structure represents the parsing trees, in which leaf nodes are observations from the vision modules and the non-leaf nodes are non-terminal symbols. At any given stage of the process, the representation may have multiple tree structures. For implementation reasons, we use a DUMMY root node to combine multiple trees. Extracting semantic trees from observing manipulation actions is the target of the cognitive system.

Before introducing the parsing algorithms, we first introduce the core of our reasoning module: the Manipulation Action Context-Free Grammar (MACFG). In formal language theory, a context-free grammar is a formal grammar in which every production rule is of the form $V \rightarrow w$, where V is a single nonterminal symbol, and w is a string of terminals and/or nonterminals (w can be empty). The basic recursive structure of natural languages, the way in which clauses nest inside other clauses, and the way in which lists of adjectives and adverbs are followed by nouns and verbs, is described exactly by a context-free grammar.

Similarly for manipulation actions, every complex activity is built of smaller blocks. Using linguistics notation, a block consists of a "Subject", "Action" and "Patient" triplet. Here a "Subject" can be either a hand or an object, and the same holds for the "Patient". Furthermore, a complex activity also has a basic recursive structure, and can be decomposed into simpler actions. For example, the typical manipulation activity "sawing a plank" is described by the top-level triplet "handsaw saw plank", and has two lower-level triplets (which come before the top-level action in time), namely "hand grasp saw" and "hand grasp plank". Intuitively, the process of observing and interpreting manipulation actions is syntactically quite similar to

Table 5.1: A Manipulation Action Context-Free Grammar.

AP	$\rightarrow$	$A\ O \mid A\ HP$	(1)
HP	$\rightarrow$	$H\ AP \mid HP\ AP$	(2)
H	$\rightarrow$	h	
A	$\rightarrow$	a	
O	$\rightarrow$	o	(3)

natural language understanding. Thus, the Manipulation Grammar (Table 5.1) is presented to parse manipulation actions.

The nonterminals H, A, and O represent the hand, the action and the object (the tools and objects under manipulation) respectively, and the terminals h, a, and o are the observations. AP stands for Action Phrase and HP for Hand Phrase. They are proposed here as XPs following the X-bar theory, which is used to construct the logical form of the semantic structure [151].

The design of this grammar is motivated by three observations: (i) Hands are the main driving force in manipulation actions, so a specialized nonterminal symbol H is used for their representation; (ii) An Action (A) can be applied to an Object (O) directly or to a Hand Phrase (HP), which in turn contains an Object (O), as encoded in Rule (1), which builds up an Action Phrase (AP); (iii) An Action Phrase (AP) can be combined either with the Hand (H) or a Hand Phrase, as encoded in rule (2), which recursively builds up the Hand Phrase. The rules discussed in Table 5.1 form the syntactic rules of the grammar used in the parsing algorithms.

5.3.2 Cognitive MACFG Parsing Algorithms

Our aim for this project is not only to provide a grammar for representing manipulation actions, but also to develop a set of operations that can automatically parse (create or dissolve) the semantic tree structures. This is crucial for practical purposes, since parsing a manipulation action is inherently an on-line process. The observations are obtained in a temporal sequence. Thus, the parsing algorithm for the grammar should be able to dynamically update the tree structures. At any point, the current leaves of the semantic forest structures represent the actions and objects involved so far. When a new triplet of ("Subject", "Action", "Patient") arrives, the parser updates the tree using the construction or destruction routine.

Theoretically, the non-regular context-free language defined in Table 5.1 can be recognized by a non-deterministic pushdown automaton. However, different from language input, the perception input is naturally a temporal sequence of observations. Thus, instead of simply building a non-deterministic pushdown automaton, it requires a special set of parsing operations.

Our parsing algorithm differentiates between constructive and destructive actions. Constructive actions are the movements that start with the hand (or a tool) coming in contact with an object and usually result in a certain physical change on the object (a consequence), e.g, "Grasp", "Cut", or "Saw". Destructive actions are movements at the end of physical change inducing actions, when the hand (or tool) separates from the object; some examples are "Ungrasp" or "FinishedCut". A constructive action may or may not have a corresponding destructive action, but

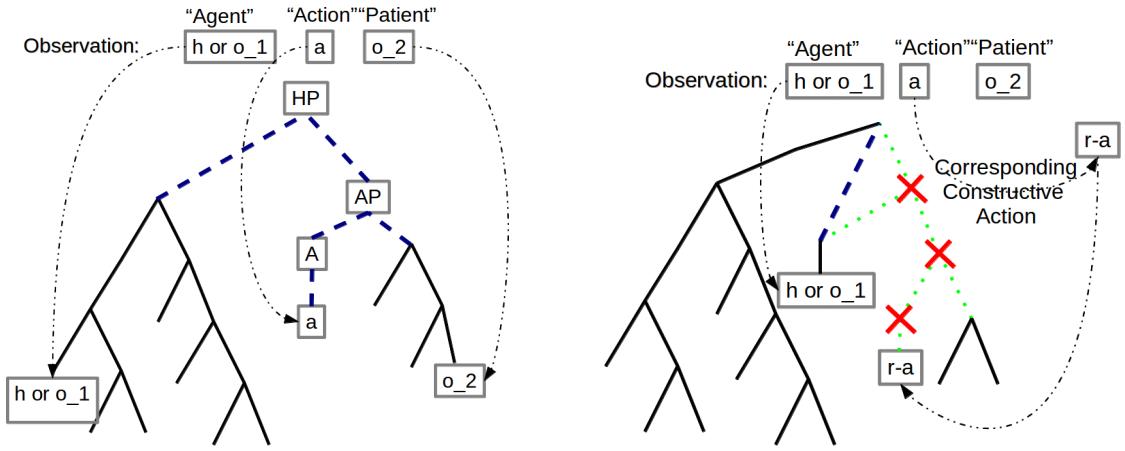


Figure 5.2: The (a) construction and (b) destruction operations. Fine dashed lines are newly added connections, crosses are node deletion, and fine dotted lines are connections to be deleted.

every destructive action must have a corresponding constructive action. Otherwise the parsing algorithm detects an error. In order to facilitate the action recognition, a look-up table that stores the constructive-destructive action pairs is used. This knowledge can be learned and further expanded.

The algorithm builds a tree structure T_s. This structure is updated as new observations are received. Once an observation triplet "Subject", "Action", and "Patient" arrives, the algorithm checks whether the "Action" is constructive or destructive and then follows one of two pathways. If the "Action" is constructive, a construction routine is used. Otherwise a destruction routine is used (Refer to Algorithm 2, Algorithm 3, and Algorithm 4 for details). The process continues till the last observation. Two illustrations in Figure 5.2 demonstrate how the construction and the destruction routines work. The parse operation amounts to a chart

parser [152], which takes in the three nonterminals and performs bottom-up parsing following the context-free rules from Table 5.1.

Algorithm 2 Dynamic manipulation action tree parsing

Initialize an empty tree group (forest) T_s

while New observation (subject s, action a, patient p) **do**

if a is a constructive action **then**

construction(T_s, s, a, p)

end if

if a is a destructive action **then**

destruction(T_s, s, a, p)

end if

end while

Figure 5.3 shows a typical manipulation action example. The parsing algorithm takes as input a sequence of key observations: "LeftHand Grasp Knife; RightHand Grasp Eggplant; Knife Cut Eggplant; Knife FinishedCut Eggplant; RightHand Ungrasp Eggplant; LeftHand Ungrasp Knife". Then a sequence of six tree structures is parsed up or dissolved along the time line. We provide more examples in Section 5.4, and a sample implementation of the parsing algorithm at http://www.umiacs.umd.edu/~yzyang/MACFG. For clarity, Figure 5.3 uses a dummy root node to create a tree structure from a forest and numbers the nonterminal nodes.

Algorithm 3 construction(T_s, s, a, p)

Previous tree group (forest) T_s and new observation (subject s, action a and patient p)

if s is Hand h, and p is an object o **then**

Find the highest subtrees T_h and T_o from T_s containing h and o. If h or o is not in the current forest, create new subtrees T_h and T_o, respectively.

parse(T_h,a,T_o), attach it to update T_s.

end if

if s is an object o_1 and p is another object o_2 **then**

Find the highest subtree T_o^1 and T_o^2 from T_s containing o_1 and o_2 respectively. If either o_1 or o_2 is not in the current forest, create new subtree T_o^1 or T_o^2. If both o_1 and o_2 are not in the current forest, raise an error.

parse(T_o^1,a,T_o^2), attach it to update T_s.

end if

5.3.3 Attention Mechanism with the Torque Operator

It is essential for our cognitive system to have an effective attention mechanism, because the amount of information in real world images is vast. Visual attention, the process of driving an agent's attention to a certain area, is based on both bottom-up processes defined on low-level visual features and top-down processes influenced by the agent's previous experience and goals [123]. Recently, [153] have provided a vision tool, called the image torque, that captures the concept of closed contours using bottom-up processes. Basically, the torque operator takes simple edges as

Algorithm 4 destruction(T_s, s, a, p)

Previous tree structures T_s and new observation (subject s, action a and patient p)

Find corresponding constructive action of a from the look-up table and denote it as a'

if There exists a lowest subtree T'_a that contains both s and a' **then**

Remove every node on the path that starts from root of T'_a to a'.

if T'_a has a parent node **then**

Connect the highest subtree that contains s with T'_a's parent node.

end if

Leave all the remaining subtrees as individual trees.

end if

Set the rest of T_s as new T_s.

input and computes, over regions of different sizes, a measure of how well the edges are aligned to form a closed, convex contour.

The underlying motivation is to find object-like regions by computing the "coherence" of the edges that support the object. Edge coherence is measured via the cross-product between the tangent vector of the edge pixel and a vector from a center point to the edge pixel, as shown in Figure 5.4(a). Formally, the value of torque, τ_{pq} of an edge pixel q within a discrete image patch with center p is defined as

$$\tau_{pq} = ||\vec{r}_{pq}||sin\theta_{pq} \ , \tag{5.1}$$

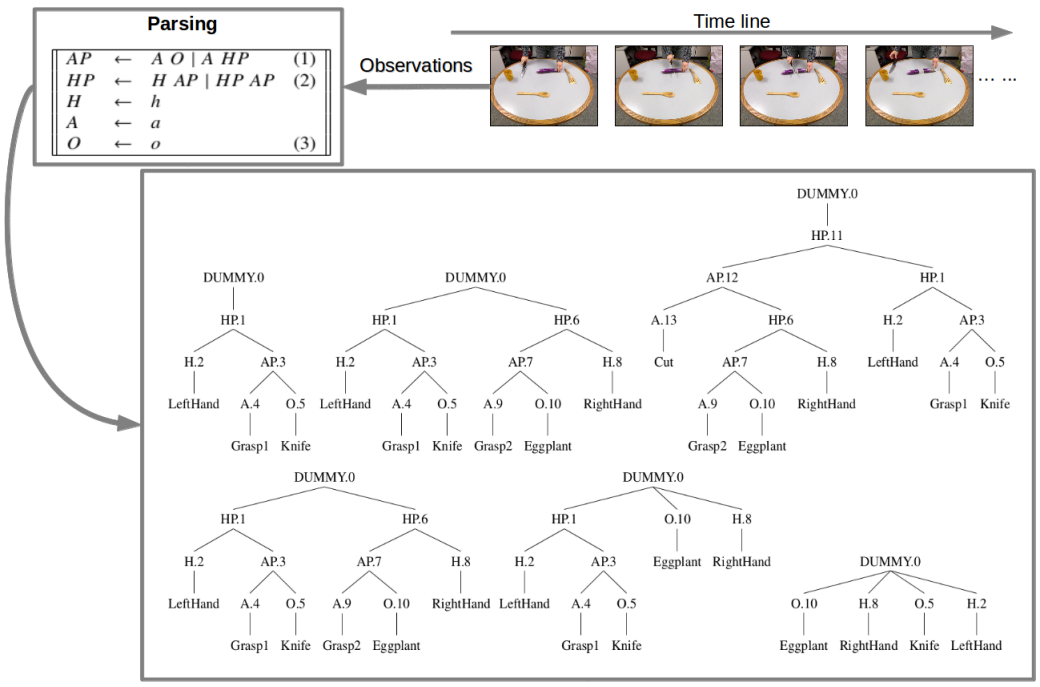


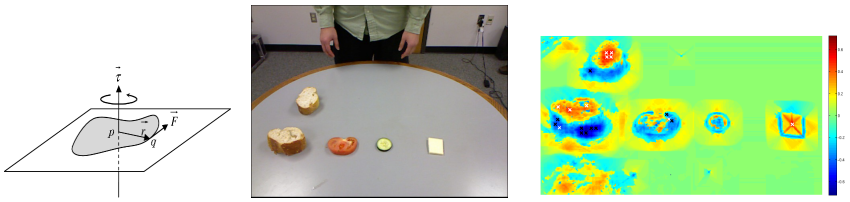
Figure 5.3: Here the system observes a typical manipulation action example, "Cut an eggplant", and builds a sequence of six action trees.

where $\vec{r}_{pq}$ is the displacement vector from p to q, and θ_{pq} is the angle between $\vec{r}_{pq}$ and the tangent vector at q. The sign of τ_{pq} depends on the direction of the tangent vector and for this work, our system computes the direction based on the intensity contrast along the edge pixel. The torque of an image patch, P, is defined as the sum of the torque values of all edge pixels, $E(P)$, within the patch as

$$\tau_P = \frac{1}{2|P|} \sum_{q \in E(P)} \tau_{pq} \ . \tag{5.2}$$

In this work, our system processes the testing sequences by applying the torque operators to obtain possible initial fixation points for the object monitoring process. Figure 5.4 shows an example of the application of the torque operator.

The system also employs a top-down attention mechanism; it uses the hand



(a) (b) (c)

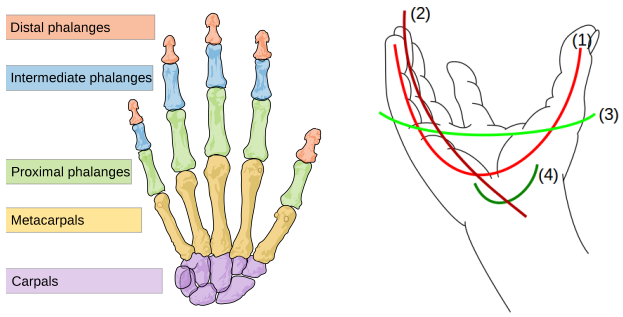
Figure 5.4: (a) Torque for images, (b) a sample input frame, and (c) torque operator response. Crosses are the pixels with top extreme torque values that serve as the potential fixation points.

location to guide the attention. Here, we integrate the bottom-up torque operator output with hand tracking. Potential objects under manipulation are found when one of the hand regions intersects a region with high torque responses, after which the object monitoring system (Section 5.3.5) monitors it.

5.3.4 Hand Tracking, Grasp Classification and Action Recognition

With the recent development of a vision-based, markerless, fully articulated model-based human hand tracking system [32] (http://cvrlcode.ics.forth.gr/handtracking/), the system is able to track a 26 degree of freedom model of hand. It is worth noting, however, that for a simple classification of movements into a small number of actions, the location of the hands and objects would be sufficient. Moreover, with the full hand model, a finer granularity of description can be achieved by classifying the tracked hand-model into different grasp types.

We collected training data from different actions, which then was processed. A set of bio-inspired features, following hand anatomy [33], were extracted. Intuitively,



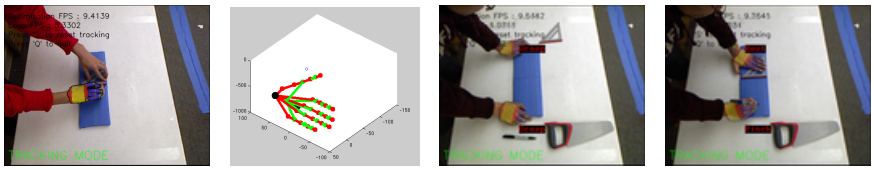
(a) (b)

Figure 5.5: (a) Bones of the human hand. (b) Arches of the hand: (1) one of the oblique arches; (2) one of the longitudinal arches of the digits; (3) transverse metacarpal arch; (4) transverse carpal arch. Source: [7].

the arches formed by the fingers are crucial to differentiate different grasp types. Figure 5.5 shows that the fixed and mobile parts of the hand adapt to various everyday tasks by forming bony arches: longitudinal arches (the rays formed by finger bones and associated metacarpal bones), and oblique arches (between the thumb and four fingers).

In each image frame, our system computed the oblique and longitudinal arches to obtain an eight parameter feature vector, as in Figure 5.6(a). We further reduced the dimensionality of the feature space by Principle Component Analysis and then applied k-means clustering to discover the underlying four general types of grasp, which are Rest, Firm Grasp, Delicate Grasp (Pinch) and Extension Grasp. To classify a given test sequence, the data was processed as described above and then the grasp type was computed using a naive Bayesian classifier. Figure 5.6(c) and (d) show examples of the classification result.

The grasp classification is used to segment the image sequence in time and also



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

Figure 5.6: (a) One example of fully articulated hand model tracking, (b) a 3-D illustration of the tracked model, and (c-d) examples of grasp type recognition for both hands.

serves as part of the action description. In addition, our system uses the trajectory of the mass center of the hands to classify the actions. The hand-tracking software provides the hand trajectories (of the given action sequence between the onset of grasp and release of the object), from which our system computed global features of the trajectory, including the frequency and velocity components. Frequency is encoded by the first four real coefficients of the Fourier transform in all the x, y and z directions, which gives a 24 dimensional vector over both hands. Velocity is encoded by averaging the difference in hand positions between two adjacent timestamps, which gives a six dimensional vector. These features are then combined to yield a 30 dimensional vector that the system uses for action recognition [24].

5.3.5 Object Monitoring and Recognition

Manipulation actions commonly involve objects. In order to obtain the information necessary to monitor the objects being worked on, our system applies a new method combining segmentation and tracking [26]. This method combines stochastic tracking [122] with a fixation-based active segmentation [121]. The track-

ing module provides a number of tracked points. The locations of these points define an area of interest and a fixation point for the segmentation. The data term of the segmentation module uses the color immediately surrounding the fixation points. The segmentation module segments the object and updates the appearance model for the tracker. Using this method, the system tracks objects as they deform and change topology (two objects can merge, or an object can be divided into parts.).

For object recognition, our system simply uses color information. The system uses a color distribution model to be invariant to various textures or patterns. A function $h(x_i)$ is defined to create a color histogram, which assigns one of the m-bins to a given color at location x_i. To make the algorithm less sensitive to lighting conditions, the system uses the Hue-Saturation-Value color space with less sensitivity in the V channel ($8 \times 8 \times 4$ bins). The color distribution for segment $s^{(n)}$ is denoted as

$$p(s^{(n)})^{(u)} = \gamma \sum_{i=1}^{I} k(||y - x_i||)\delta[h(x_i) - u] \ , \tag{5.3}$$

where $u = 1 \ldots m$, $\delta(.)$ is the Kronecker delta function and γ is the normalization term $\gamma = \frac{1}{\sum_{i=1}^{I} k(||y-x_i||)}$. $k(.)$ is a weighting function designed from the intuition that not all pixels in the sampling region are equally important for describing the color model. Specifically, pixels that are farther away from the fixation point are assigned smaller weights,

$$k(r) = \begin{cases} 1 - r^2 & \text{if } r < a \\ 0 & \text{otherwise} \end{cases} \ , \tag{5.4}$$

where the parameter a is used to adapt the size of the region, and r is the distance

from the fixation. By applying the weighting function, we increase the robustness of the distribution by weakening the influence of boundary pixels that may belong to the background or are occluded.

Since the objects in our experiments have distinct color profiles, the color distribution model used was sufficient to recognize the segmented objects. We manually labelled several examples from each object class as training data and used a nearest k-neighbours classifier. Figure 5.7 shows sample results.

5.3.6 Detection of Manipulation Action Consequences

Taking an object-centric point of view, manipulation actions can be classified into six categories according to how the manipulation transforms the object, or, in other words, what consequence the action has on the object. These categories are Divide, Assemble, Create, Consume, Transfer, and Deform.

To describe these action categories we need a formalism. We use the visual semantic graph (VSG) inspired by the work of [14]. This formalism takes as input computed object segments, their spatial relationship, and the temporal relationship over consecutive frames. Please refer to Chapter 4 for details.

Our system integrated visual modules in the following manner. Since hands are the most important components in manipulation actions, a state-of-the-art markerless hand tracking system obtains skeleton models of both hands. Using this data, our system classifies the manner in which the human grasps the objects into four primitive categories. On the basis of the grasp classification, our system finds the

start and end points of action sequences. Our system then classifies the action from the hand trajectories, the hand grasp, and the consequence of the object (as explained above). To obtain objects, our system monitors the manipulated object using a process that combines stochastic tracking with active segmentation, then recognizes the segmented objects using color. Finally, based on the monitoring process, our system checks and classifies the effect on the object into one of four fundamental types of "consequences". The final output are sequences of "Subject" "Action" "Patient" triplets, and the manipulation grammar parser takes them as input to build up semantic structures.

5.4 Experiments

The theoretical framework we have presented suggests two hypotheses that deserve empirical tests: (a) manipulation actions performed by single human agents obey a manipulation action context-free grammar that includes manipulators, actions, and objects as terminal symbols; (b) a variant on chart parsing that includes both constructive and destructive actions, combined with methods for hand tracking and action and object recognition from RGBD data, can parse observed human manipulation actions.

To test the two hypotheses empirically, we need to define a set of performance variables and how they relate to our predicted results. The first hypothesis relates to representations, and we can empirically test if it is possible to manually generate target trees for each manipulation action in the test set. The second hypothesis has

to do with processes, and can be tested by observing how many times our system builds a parse tree successfully from observed human manipulation actions. The theoretical framework for this consists of two parts: (1) a visual system to detect (subject, action, object) triplets from input sensory data and (2) a variant of the chart parsing system to transform the sequence of triplets into tree representations. Thus, we further separate the test for our second hypothesis into two sub-tasks: (1) we measure the precision and recall metrics by comparing the detected triplets from our visual system with human-labelled ground truth data and (2) given the ground truth triplets as input, we measure the success rate using our variant of chart parsing system by comparing it with the target trees for each action. We consider a parse successful if the generated tree is identical with the manually generated target parse. Therefore, we consider the second hypothesis supported when (1) our visual system achieves high precision and recall, and (2) our parsing system achieves a high success rate. We use the ground truth triplets as input instead of the detected ones because we cannot expect the visual system to generate the triplets with 100% precision and recall, due to occlusions, shadows, and unexpected events. As a complete system, we expect the visual module to have high precision and recall, thus the detected triplets can be used as input to the parsing module in practice.

We designed our experiments under the setting with one RGBD camera in a fixed location (we used a Kinect sensor). We asked human subjects to perform a set of manipulation actions in front of the camera while both objects and tools were presented within the view of the camera during the activity. We collected RGBD sequences of manipulation actions being performed by one human, and to

Category	*Hand*	*Object*	*Constructive*	*Destructive*
Kitchen	*LeftHand*	*Bread, Cheese, Tomato*	*Grasp, Cut*	*Ungrasp, FinishedCut*
	RightHand	*Eggplant, Cucumber*	*Assemble*	*FinishedAssemble*
Manu	*LeftHand*	*Plank*	*Grasp, Saw*	*Ungrasp, FinishedSaw*
−facturing	*RightHand*	*Saw*	*Assemble*	*FinishedAssemble*

Table 5.2: "Hands", "Objects" and "Actions" involved in the experiments.

ensure some diversity, we collected these from two domains, namely the kitchen and the manufacturing environments. The kitchen action set included "Making a sandwich", "Cutting an eggplant", and "Cutting bread", and the manufacturing action set included "Sawing a plank into two pieces" and "Assemble two pieces of the plank". To further diversify the data set, we adopted two different viewpoints for each action set. For the kitchen actions, we used a front view setting; for manufacturing actions, a side view setting. The five sequences have 32 human-labelled ground truth triplets. Table 5.4 gives a list of "Subjects", "Objects", and "Actions" involved in our experiments.

To evaluate the visual system, we applied the vision techniques introduced in Sections 5.3.3 to 5.3.6. To be specific, the grasp type classification module provides a "Grasp" signal when the hand status changes from "Rest" to one of the three other types, and an "Ungrasp" signal when it changes back to "Rest". At the same time, the object monitoring and the segmentation-based object recognition module provides the "Object" symbol when either of the hands touch an object. Also, the hand tracking module provides trajectory profiles that enable the trajectory-based action recognition module to produce "Action" symbols such as "Cut" and "Saw". The action "Assemble" did not have a distinctive trajectory

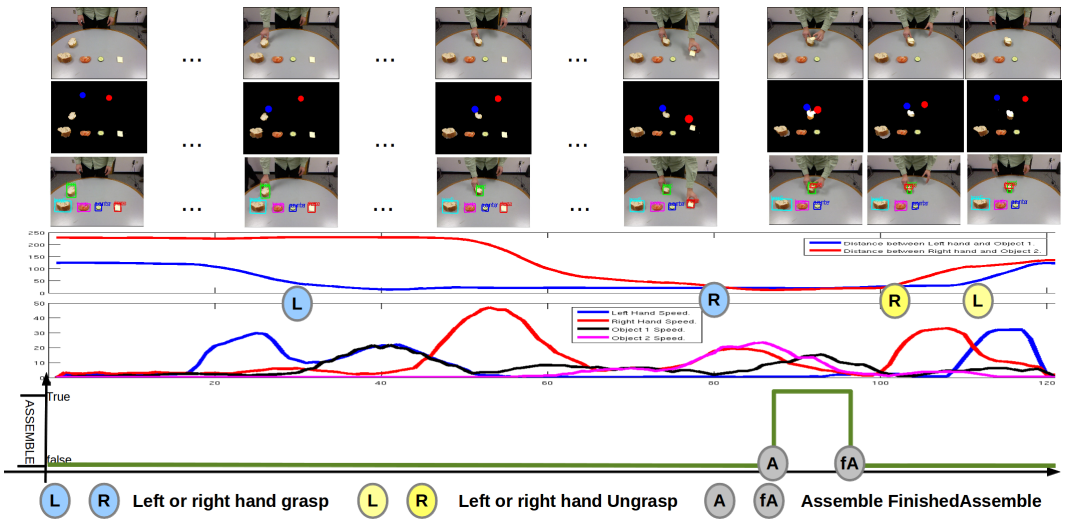


Figure 5.7: The second row shows the hand tracking and object monitoring. The third row shows the object recognition result, where each segmentation is labelled with an object name and a bounding box in different color. The fourth and fifth rows depict the hand speed profile and the Euclidean distances between hands and objects. The sixth row shows the consequence detection.

profile, so we simply generated it when the "Cheese" merged with the "Bread" based on the object monitoring process. At the end of each recognized action, a corresponding destructive symbol, as defined in Table 5.4, is produced, and the consequence checking module is called to confirm the action consequence. Figure 5.7 shows intermediate and final results of vision modules from a sequence of a person making a sandwich. In this scenario, our system reliably tracks, segments and recognizes both hands and objects, recognizes "grasp", "ungrasp" and "assemble" events, and generates a sequence of triplets along the time-line. To evaluate the parsing system, given the sequence of ground truth triplets as inputs, a sequence of trees (or forests) is created or dissolved dynamically using the manipulation action

context free grammar parser (Section 5.3.1, 5.3.2).

Our experiments produced three results: (i) we were able to manually generate a sequence of target tree representations for each of the five sequences in our data set; (ii) our visual system detected 34 triplets, of which 29 were correct (compared with the 32 ground truth labels), and yielded a precision of 85.3% and a recall of 90.6%; (iii) given the sequence of ground truth triplets, our parsing system successfully parsed all five sequences in our data set into tree representations comparing with the target parses. Figure 5.8 shows the tree structures built from the sequence of triplets of the "Making a sandwich" sequence. More results for the rest of manipulation actions in the data set can be found at http://www.umiacs.umd.edu/~yzyang/MACFG. Overall, (i) supports our first hypothesis that human manipulation actions obey a manipulation action context-free grammar that includes manipulators, actions, and objects as terminal symbols, while (ii) and (iii) support our second hypothesis that the implementation of our cognitive system can parse observed human manipulation actions. [1]

The experimental results support our hypotheses, but we have not tested our system on a large data set with a variety of manipulation actions. We are currently testing the system on a larger set of kitchen actions and checking to see if our hypotheses are still supported.

[1] As the experiments demonstrate, the system was robust enough to handle situations that involve hesitation, in which the human grasps a tool, finds that it is not the desired one, and ungrasps it (as in Figure 5.9).

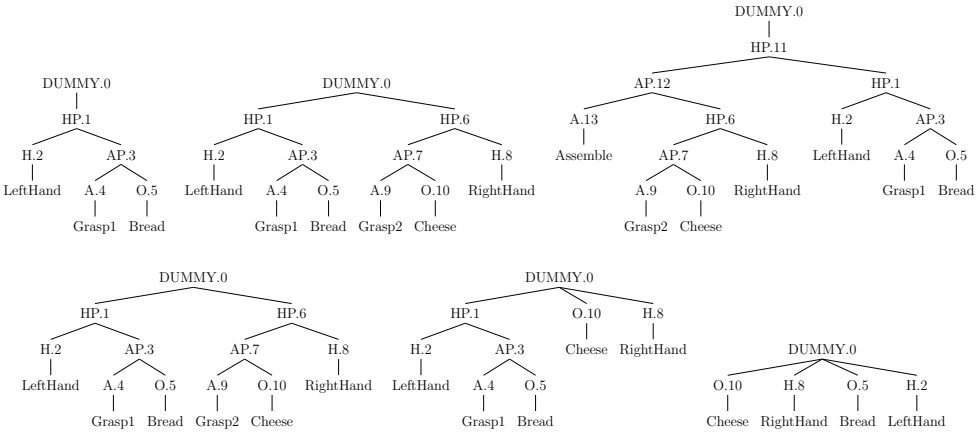


Figure 5.8: The tree structures generated from the "Make a Sandwich" sequence. Figure 5.7 depicts the corresponding visual processing. Since our system detected six triplets temporally from this sequence, it produced a set of six trees. The order of the six trees is from left to right.

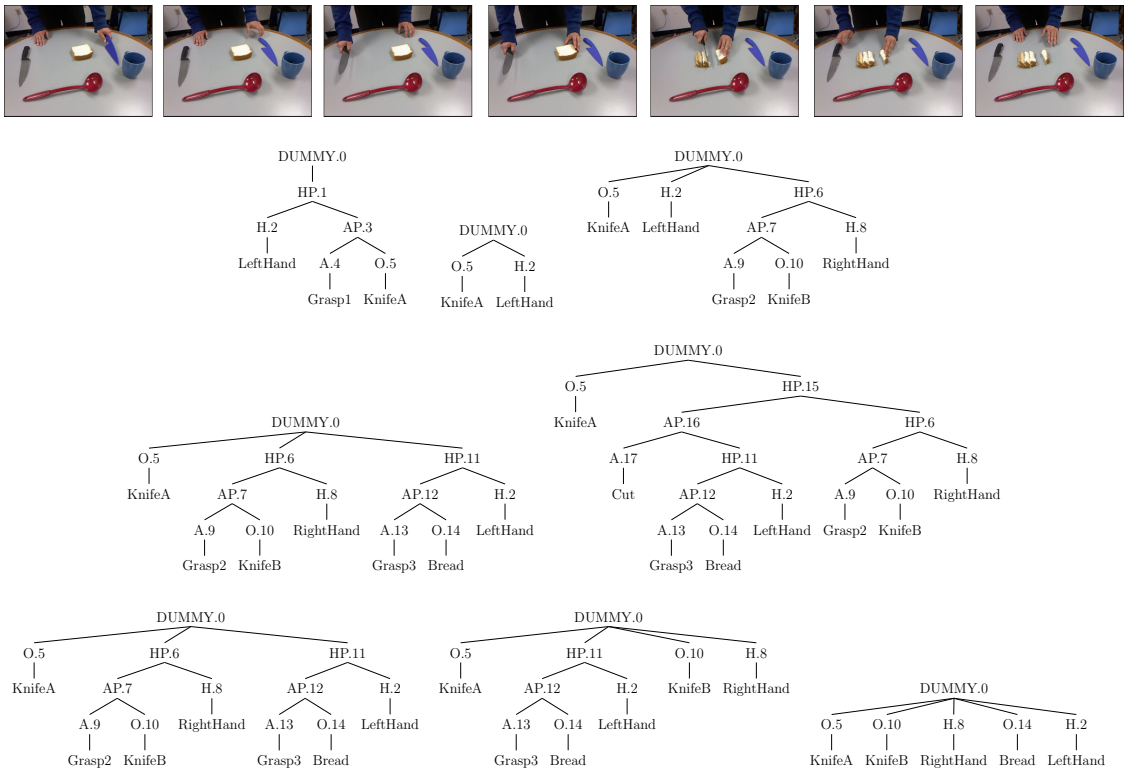


Figure 5.9: Example of the grammar that deals with hesitation. This figure shows key frames of the input visual data and the semantic tree structures.

Chapter 6: The Semantics: Learning Manipulation Action Semantics through Probabilistic Combinatory Categorial Grammar Parsing

6.1 Introduction

Autonomous robots will need to learn the actions that humans perform. They will need to recognize these actions when they see them and they will need to perform these actions themselves. This requires a formal system to represent the action semantics. This representation needs to store the semantic information about the actions, be encoded in a machine readable language, and inherently be in a programmable fashion in order to enable reasoning beyond observation. A formal representation of this kind has a variety of other applications such as intelligent manufacturing, human robot collaboration, action planning and policy design, etc.

In this chapter, we are concerned with manipulation actions, that is actions performed by agents (humans or robots) on objects, resulting in some physical change of the object. However most of the current AI systems require manually defined semantic rules. In this work, we propose a computational linguistics framework, which is based on probabilistic semantic parsing with Combinatory Categorial

Grammar (CCG), to learn manipulation action semantics (lexicon entries) from annotations. We later show that this learned lexicon is able to make our system reason about manipulation action goals beyond just observation. Thus the intelligent system can not only imitate human movements, but also imitate action goals.

Understanding actions by observation and executing them are generally considered as dual problems for intelligent agents. The sensori-motor bridge connecting the two tasks is essential, and a great amount of attention in AI, Robotics as well as Neurophysiology has been devoted to investigating it. Experiments conducted on primates have discovered that certain neurons, the so-called mirror neurons, fire during both observation and execution of identical manipulation tasks [9, 10]. This suggests that the same process is involved in both the observation and execution of actions. From a functionalist point of view, such a process should be able to first build up a semantic structure from observations, and then the decomposition of that same structure should occur when the intelligent agent executes commands.

Additionally, studies in linguistics [154] suggest that the language faculty develops in humans as a direct adaptation of a more primitive apparatus for planning goal-directed action in the world by composing affordances of tools and consequences of actions. It is this more primitive apparatus that is our major interest in this chapter. Such an apparatus is composed of a "syntax part" and a "semantic part". In the syntax part, every linguistic element is categorized as either a function or a basic type, and is associated with a syntactic category which either identifies it as a function or a basic type. In the semantic part, a semantic translation is attached following the syntactic category explicitly.

Combinatory Categorial Grammar (CCG) introduced by [155] is a theory that can be used to represent such structures with a small set of combinators such as functional application and type-raising. What do we gain though from such a formal description of action? This is similar to asking what one gains from a formal description of language as a generative system. Chomskys contribution to language research was exactly this: the formal description of language through the formulation of the Generative and Transformational Grammar [22]. It revolutionized language research opening up new roads for the computational analysis of language, providing researchers with common, generative language structures and syntactic operations, on which language analysis tools were built. A grammar for action would contribute to providing a common framework of the syntax and semantics of action, so that basic tools for action understanding can be built, tools that researchers can use when developing action interpretation systems, without having to start development from scratch. The same tools can be used by robots to execute actions.

In this chapter, we propose an approach for learning the semantic meaning of manipulation action through a probabilistic semantic parsing framework based on CCG theory. For example, we want to learn from an annotated training action corpus that the action "Cut" is a function which has two arguments: a subject and a patient. Also, the action consequence of "Cut" is a separation of the patient. Using formal logic representation, our system will learn the semantic representations of "Cut":

$$Cut := (AP\backslash NP)/NP : \lambda x.\lambda y.cut(x, y) \rightarrow divided(y)$$

Here $cut(x, y)$ is a primitive function. We will further introduce the representation in Sec. 6.3. Since our action representation is in a common calculus form, it enables naturally further logical reasoning beyond visual observation.

The advantage of our approach is twofold: 1) Learning semantic representations from annotations helps an intelligent agent to enrich automatically its own knowledge about actions; 2) The formal logic representation of the action could be used to infer the object-wise consequence after a certain manipulation, and can also be used to plan a set of actions to reach a certain action goal. We further validate our approach on a large publicly available manipulation action dataset (MANIAC) from [15], achieving promising experimental results. Moreover, we believe that our work, even though it only considers the domain of manipulation actions, is also a promising example of a more closely intertwined computer vision and computational linguistics system. The diagram in Fig.6.1 depicts the framework of the system.

6.2 Related Works

Manipulation Action Grammar: As mentioned before, [148] suggested that a minimalist generative grammar, similar to the one of human language, also exists for action understanding and execution. The works closest related to this chapter are [17,18,31]. [17] first discussed a Chomskyan grammar for understanding complex actions as a theoretical concept, and [18] provided an implementation of such a grammar using as perceptual input only objects. More recently, [30] proposed a set of context-free grammar rules for manipulation action understanding, and [29]

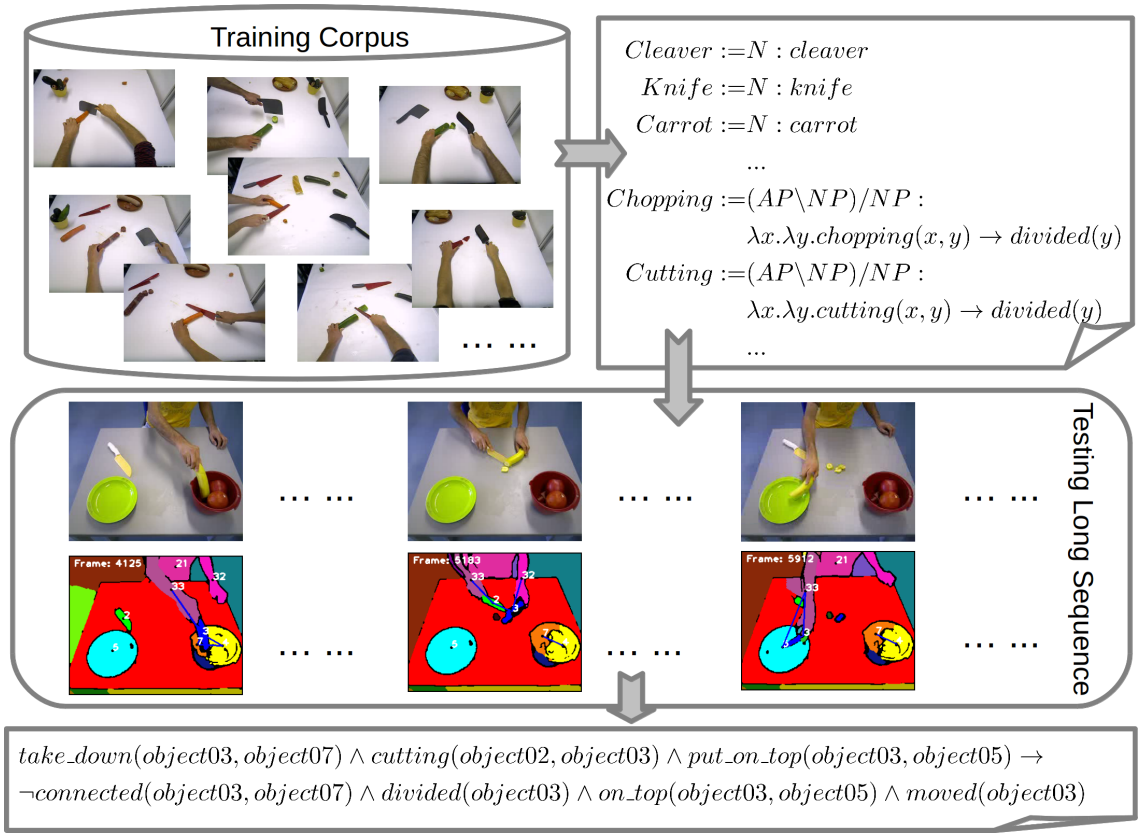


Figure 6.1: A CCG based semantic parsing framework for manipulation actions.

applied it on unconstrained instructional videos. However, these approaches only consider the syntactic structure of manipulation actions without coupling semantic rules using λ expressions, which limits the capability of doing reasoning and prediction.

Combinatory Categorial Grammar and Semantic Parsing: CCG based semantic parsing originally was used mainly to translate natural language sentences to their desired semantic representations as λ-calculus formulas [156, 157]. [158] presented a framework of grounded language acquisition: the interpretation of language entities into semantically informed structures in the context of perception

and actuation. The concept has been applied successfully in tasks such as robot navigation [159], forklift operation [160] and of human-robot interaction [161]. In this work, instead of grounding natural language sentences directly, we ground information obtained from visual perception into semantically informed structures, specifically in the domain of manipulation actions.

6.3 A CCG Framework for Manipulation Actions

Before we dive into the semantic parsing of manipulation actions, a brief introduction to the Combinatory Categorial Grammar framework in Linguistics is necessary. We will only introduce related concepts and formalisms. For a complete background reading, we would like to refer readers to [155]. We will first give a brief introduction to CCG and then introduce a fundamental combinator, i.e., functional application. The introduction is followed by examples to show how the combinator is applied to parse actions.

6.3.1 Manipulation Action Semantics

The semantic expression in our representation of manipulation actions uses a typed λ-calculus language. The formal system has two basic types: entities and functions. Entities in manipulation actions are Objects or Hands, and functions are the Actions. Our *lambda*-calculus expressions are formed from the following items:

Constants: Constants can be either entities or functions. For example, *Knife* is an entity (i.e., it is of type N) and *Cucumber* is an entity too (i.e., it is of type

N). *Cut* is an action function that maps entities to entities. When the event *Knife Cut Cucumber* happened, the expression *cut(Knife, Cucumber)* returns an entity of type AP, aka. Action Phrase. Constants like *divided* are status functions that map entities to truth values. The expression $divided(cucumber)$ returns a true value after the event (*Knife Cut Cucumber*) happened.

Logical connectors: The λ-calculus expression has logical connectors like conjunction ($\wedge$), disjunction ($\vee$), negation($\neg$) and implication($\rightarrow$).

For example, the expression

$$connected(tomato, cucumber) \wedge$$

$$divided(tomato) \wedge divided(cucumber)$$

represents the joint status that the sliced *tomato* merged with the sliced *cucumber*. It can be regarded as a simplified goal status for "making a cucumber tomato salad". The expression $\neg connected(spoon, bowl)$ represents the status after the *spoon* finished stirring the *bowl*.

$$\lambda x.cut(x, cucumber) \rightarrow divided(cucumber)$$

represents that if the *cucumber* is cut by x, then the status of the *cucumber* is divided.

λ **expressions**: *lambda* expressions represent functions with unknown arguments. For example, $\lambda x.cut(knife, x)$ is a function from entities to entities, which is of type NP after any entities of type N that is cut by *knife*.

6.3.2 Combinatory Categorial Grammar

The semantic parsing formalism underlying our framework for manipulation actions is that of combinatory categorial grammar (CCG) [155]. A CCG specifies one or more logical forms for each element or combination of elements for manipulation actions. In our formalism, an element of Action is associated with a syntactic "category" which identifies it as functions, and specifies the type and directionality of their arguments and the type of their result. For example, action "Cut" is a function from patient object phrase (NP) on the right into predicates, and into functions from subject object phrase (NP) on the left into a sub action phrase (AP):

$$Cut := (AP\backslash NP)/NP. \tag{6.1}$$

As a matter of fact, the pure categorial grammar is a conext-free grammar presented in the accepting, rather than the producing direction. The expression (6.1) is just an accepting form for Action "Cut" following the context-free grammar. While it is now convenient to write derivations as follows, they are equivalent to conventional tree structure derivations in Figure. 6.3.2.

$$\dfrac{\dfrac{\text{Knife}}{N} \quad \dfrac{\text{Cut}}{(AP\backslash NP)/NP} \quad \dfrac{\text{Cucumber}}{N}}{\dfrac{\dfrac{}{NP} \quad \dfrac{}{AP\backslash NP}\,>}{AP}\,<}$$

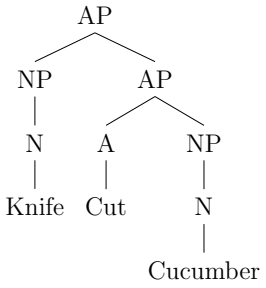


Figure 6.2: Example of conventional tree structure.

The semantic type is encoded in these categories, and their translation can be made explicit in an expanded notation. Basically a λ-calculus expression is attached with the syntactic category. A colon operator is used to separate syntactical and semantic expressions, and the right side of the colon is assumed to have lower precedence than the left side of the colon. Which is intuitive as any explanation of manipulation actions should first obey syntactical rules, then semantic rules. Now the basic element, Action "Cut", can be further represented by:

$$Cut := (AP \backslash NP)/NP : \lambda x.\lambda y.cut(x, y) \rightarrow divided(y).$$

$(AP \backslash NP)/NP$ denotes a phrase of type AP, which requires an element of type NP to specify what object was cut, and requires another element of type NP to further complement what effector initiates the cut action. $\lambda x.\lambda y.cut(x, y)$ is the λ-calculus representation for this function. Since the functions are closely related to the state update, $\rightarrow divided(y)$ further points out the status expression after the action was performed.

A CCG system has a set of combinatory rules which describe how adjacent syntatic categories in a string can be recursively combined. In the setting of ma-

nipulation actions, we want to point out that similar combinatory rules are also applicable. Especially the functional application rules are essential in our system.

6.3.3 Functional application

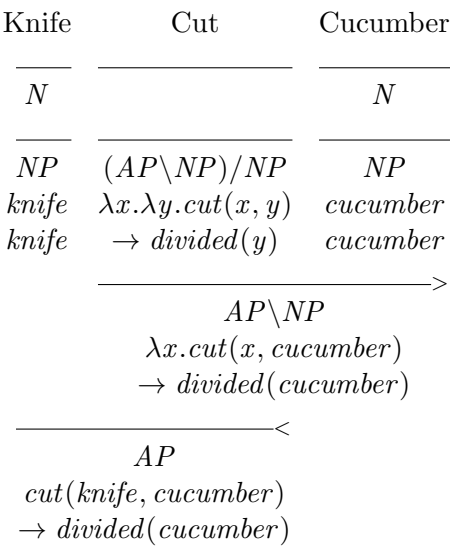
The functional application rules with semantics can be expressed in the following form:

$$A/B : f \quad B : g => A : f(g) \tag{6.2}$$

$$B : g \quad A\backslash B : f => A : f(g) \tag{6.3}$$

Rule. (6.2) says that a string with type A/B can be combined with a right-adjacent string of type B to form a new string of type A. At the same time, it also specifies how the semantics of the category A can be compositionally built out of the semantics for A/B and B. Rule. (6.3) is a symmetric form of Rule. (6.2).

In the domain of manipulation actions, following derivation is an example CCG parse. This parse shows how the system can parse an observation ("Knife Cut Cucumber") into a semantic representation ($cut(knife, cucumber) \rightarrow divided(cucumber)$) using the functional application rules.



6.4 Learning Model and Semantic Parsing

After having defined the formalism and application rule, instead of manually writing down all the possible CCG representations for each entity, we would like to apply a learning technique to derive them from the paired training corpus. Here we adopt the learning model of [156], and use it to assign weights to the semantic representation of actions. Since an action may have multiple possible syntactic and semantic representations assigned to it, we use the probabilistic model to assign weights to these representations.

6.4.1 Learning Approach

First we assume that complete syntactic parses of the observed action are available, and in fact a manipulation action can have several different parses. The parsing uses a probabilistic combinatorial categorial grammar framework similar to the one given by [157]. We assume a probabilistic categorial grammar (PCCG)

based on a log linear model. M denotes a manipulation task, L denotes the semantic representation of the task, and T denotes its parse tree. The probability of a particular syntactic and semantic parse is given as:

$$P(L,T|M;\Theta) = \frac{e^{f(L,T,M)\cdot\Theta}}{\sum_{(L,T)} e^{f(L,T,M)\cdot\Theta}} \tag{6.4}$$

where f is a mapping of the triple (L,T,M) to feature vectors $\in R^d$, and the $\Theta \in R^d$ represents the weights to be learned. Here we use only lexical features, where each feature counts the number of times a lexical entry is used in T. Parsing a manipulation task under PCCG equates to finding L such that $P(L|M;\Theta)$ is maximized:

$$\begin{aligned} & argmax_L P(L|M;\Theta) \\ = & argmax_L \sum_T P(L,T|M;\Theta). \end{aligned} \tag{6.5}$$

We use dynamic programming techniques to calculate the most probable parse for the manipulation task. In this chapter, the implementation from [162] is adopted, where an inverse-λ technique is used to generalize new semantic representations. The generalization of lexicon rules are essential for our system to deal with unknown actions presented during the testing phase.

6.5 Experiments

6.5.1 Manipulation Action (MANIAC) Dataset

[15] provides a manipulation action dataset with 8 different manipulation actions (cutting, chopping, stirring, putting, taking, hiding, uncovering, and pushing),

each of which consists of 15 different versions performed by 5 different human actors[1]. There are in total 30 different objects manipulated in all demonstrations. All manipulations were recorded with the Microsoft Kinect sensor and serve as **training** data here.

The MANIAC data set contains another 20 long and complex chained manipulation sequences (e.g. "making a sandwich") which consist of a total of 103 different versions of these 8 manipulation tasks performed in different orders with novel objects under different circumstances. These serve as **testing** data for our experiments.

[8, 15] developed a semantic event chain based model free decomposition approach. It is an unsupervised probabilistic method that measures the frequency of the changes in the spatial relations embedded in event chains, in order to extract the subject and patient visual segments. It also decomposes the long chained complex testing actions into their primitive action components according to the spatio-temporal relations of the manipulator. Since the visual recognition is not the core of this work, we omit the details here and refer the interested reader to [8, 15]. All these features make the MANIAC dataset a great testing bed for both the theoretical framework and the implemented system presented in this work.

[1]Dataset available for download at https://fortknox.physik3.gwdg.de/cns/index.php?page=maniac-dataset.

6.5.2 Training Corpus

We first created a training corpus by annotating the 120 training clips from the MANIAC dataset, in the format of observed triplets (subject action patient) and a corresponding semantic representation of the action as well as its consequence. The semantic representations in λ-calculus format are given by human annotators after watching each action clip. A set of sample training pairs are given in Table.6.1 (one from each action category in the training set). Since every training clip contains one single full execution of each manipulation action considered, the training corpus thus has a total of 120 paired training samples.

Snapshot	triplet	semantic representation
	cleaver chopping carrot	$chopping(cleaver, carrot)$ $\rightarrow divided(carrot)$
	spatula cutting pepper	$cutting(spatula, pepper)$ $\rightarrow divided(pepper)$
	spoon stirring bucket	$stirring(spoon, bucket)$
	cup take_down bucket	$take\_down(cup, bucket) \rightarrow$ $\neg connected(cup, bucket) \wedge moved(cup)$
	cup put_on_top bowl	$put\_on\_top(cup, bowl) \rightarrow$ $on\_top(cup, bowl) \wedge moved(cup)$
	bucket hiding ball	$hiding(bucket, ball) \rightarrow$ $contained(bucket, ball) \wedge moved(bucket)$
	hand pushing box	$pushing(hand, box) \rightarrow moved(box)$
	box uncover apple	$uncover(box, apple) \rightarrow$ $appear(apple) \wedge moved(box)$

Table 6.1: Example annotations from training corpus, one per manipulation action category.

We also assume the system knows that every "object" involved in the corpus is an entity of its own type, for example:

$$Knife := N : knife$$

$$Bowl := N : bowl$$

.....

Additionally, we assume the syntactic form of each "action" has a main type of $(AP\backslash NP)/NP$ (see Sec. 6.3.2). These two sets of rules form the initial seed lexicon for learning.

6.5.3 Learned Lexicon

We applied the learning technique mentioned in Sec. 6.4, and we used the NL2KR implementation from [162]. The system learns and generalizes a set of lexicon entries (syntactic and semantic) for each action categories from the training corpus accompanied with a set of weights. We list the one with the largest weight

for each action here respectively:

$$Chopping := (AP\backslash NP)/NP : \lambda x.\lambda y.chopping(x,y)$$
$$\rightarrow divided(y)$$

$$Cutting := (AP\backslash NP)/NP : \lambda x.\lambda y.cutting(x,y)$$
$$\rightarrow divided(y)$$

$$Stirring := (AP\backslash NP)/NP : \lambda x.\lambda y.stirring(x,y)$$

$$Take\_down := (AP\backslash NP)/NP : \lambda x.\lambda y.take\_down(x,y)$$
$$\rightarrow \neg connected(x,y) \wedge moved(x)$$

$$Put\_on\_top := (AP\backslash NP)/NP : \lambda x.\lambda y.put\_on\_top(x,y)$$
$$\rightarrow on\_top(x,y) \wedge moved(x)$$

$$Hiding := (AP\backslash NP)/NP : \lambda x.\lambda y.hiding(x,y)$$
$$\rightarrow contained(x,y) \wedge moved(x)$$

$$Pushing := (AP\backslash NP)/NP : \lambda x.\lambda y.pushing(x,y)$$
$$\rightarrow moved(y)$$

$$Uncover := (AP\backslash NP)/NP : \lambda x.\lambda y.uncover(x,y)$$
$$\rightarrow appear(y) \wedge moved(x).$$

The set of seed lexicon and the learned lexicon entries are further used to probabilistically parse the detected triplet sequences from the 20 long manipulation activities in the testing set.

6.5.4 Deducing Semantics

Using the decomposition technique from [8, 15], the reported system is able to detect a sequence of action triplets in the form of (Subject Action Patient) from each of the testing sequence in MANIAC dataset. Briefly speaking, the event chain representation [14] of the observed long manipulation activity is first scanned to estimate the main manipulator, i.e. the hand, and manipulated objects, e.g. knife, in the scene without employing any visual feature-based object recognition method. Solely based on the interactions between the hand and manipulated objects in the scene, the event chain is partitioned into chunks. These chunks are further fragmented into sub-units to detect parallel action streams. Each parsed Semantic Event Chain (SEC) chunk is then compared with the model SECs in the library to decide whether the current SEC sample belongs to one of the known manipulation models or represents a novel manipulation. SEC models, stored in the library, are learned in an on-line unsupervised fashion using the semantics of manipulations derived from a given set of training data in order to create a large vocabulary of single atomic manipulations.

For the different testing sequence, the number of triplets detected ranges from two to seven. In total, we are able to collect 90 testing detections and they serve as the testing corpus. However, since many of the objects used in the testing data are not present in the training set, an object model-free approach is adopted and thus "subject" and "patient" fields are filled with segment IDs instead of a specific object name. Fig. 6.3 and 6.4 show several examples of the detected triplets accompanied

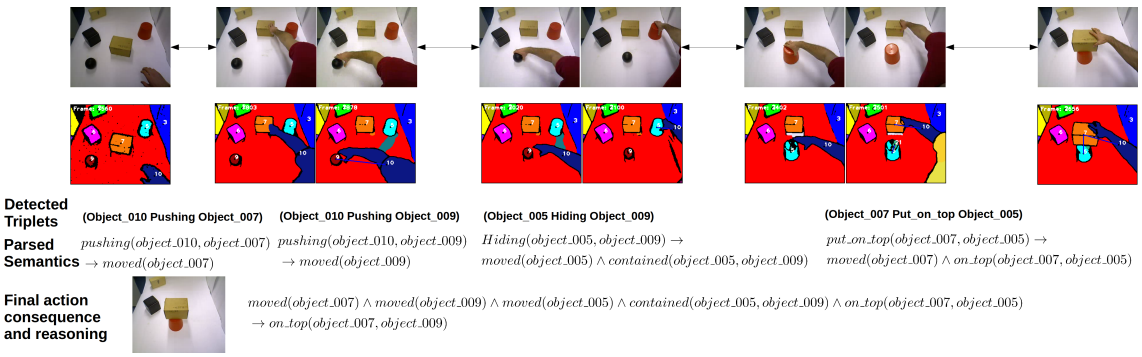


Figure 6.3: System output on complex chained manipulation testing sequence one. The segmentation output and detected triplets are from [8]

.

with a set of key frames from the testing sequences. Nevertheless, the method we used here can 1) generalize the unknown segments into the category of object entities and 2) generalize the unknown actions (those that do not exist in the training corpus) into the category of action function. This is done by automatically generalizing the following two types of lexicon entries using the inverse-λ technique from [162]:

$$Object\_[ID] := N : object\_[ID]$$

$$Unknown := (AP \backslash NP)/NP : \lambda x.\lambda y.unknown(x, y)$$

Among the 90 detected triplets, using the learned lexicon we are able to parse all of them into semantic representations. Here we pick the representation with the highest probability after parsing as the individual action semantic representation. The "parsed semantics" rows of Fig. 6.3 and 6.4 show several example action semantics on testing sequences. Taking the fourth sub-action from Fig. 6.4 as an example, the visually detected triplets based on segmentation and spatial decom-

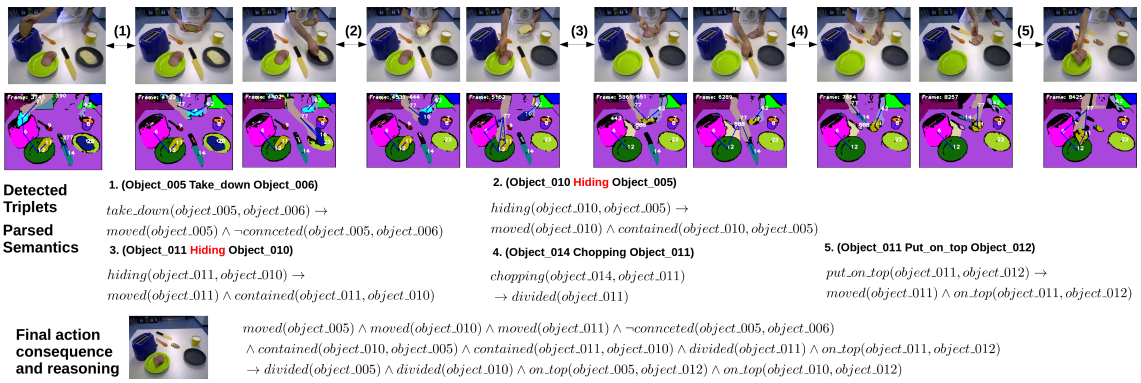


Figure 6.4: System output on the 18th complex chained manipulation testing sequence. The segmentation output and detected triplets are from [8]

.

position is $(Object\_014, Chopping, Object\_011)$. After semantic parsing, the system predicts that $divided(Object\_011)$. The complete training corpus and parsed results of the testing set will be made publicly available for future research.

6.5.5 Reasoning Beyond Observations

As mentioned before, because of the use of λ-calculus for representing action semantics, the obtained data can naturally be used to do logical reasoning beyond observations. This by itself is a very interesting research topic and it is beyond this chapter's scope. However by applying a couple of common sense Axioms on the testing data, we can provide some flavor of this idea.

Case study one: See the "final action consequence and reasoning" row of Fig. 6.3 for case one. Using propositional logic and axiom schema, we can represent the common sense statement ("if an object x is contained in object y, and object z

is on top of object y, then object z is on top of object x") as follows:

Axiom (1): $\exists x, y, z, contained(y, x) \wedge on\_top(z, y) \rightarrow on\_top(z, x)$.

Then it is trivial to deduce an additional final action consequence in this scenario that $(on\_top(object\_007, object\_009))$. This matches the fact: the yellow box which is put on top of the red bucket is also on top of the black ball.

Case study two: See the "final action consequence and reasoning" row of Fig. 6.4 for a more complicated case. Using propositional logic and axiom schema, we can represent three common sense statements:

1) "if an object y is contained in object x, and object z is contained in object y, then object z is contained in object x";

2) "if an object x is contained in object y, and object y is divided, then object x is divided";

3) "if an object x is contained in object y, and object y is on top of object z, then object x is on top of object z" as follows:

Axiom (2): $\exists x, y, z, contained(y, x) \wedge contained(z, y) \rightarrow contained(z, x)$.

Axiom (3): $\exists x, y, contained(y, x) \wedge divided(y) \rightarrow divided(x)$.

Axiom (4): $\exists x, y, z, contained(y, x) \wedge on\_top(y, z) \rightarrow on\_top(x, z)$.

With these common sense Axioms, the system is able to deduce several additional final action consequences in this scenario:

$$divided(object\_005) \wedge divided(object\_010)$$

$$\wedge on\_top(object\_005, object\_012)$$

$$\wedge on\_top(object\_010, object\_012).$$

From Fig. 6.4, we can see that these additional consequences indeed match the facts: 1) the bread and cheese which are covered by ham are also divided, even though from observation the system only detected the ham being cut; 2) the divided bread and cheese are also on top of the plate, even though from observation the system only detected the ham being put on top of the plate.

We applied the four Axioms on the 20 testing action sequences and deduced the "hidden" consequences from observation. To evaluate our system performance quantitatively, we first annotated all the final action consequences (both obvious and "hidden" ones) from the 20 testing sequences as ground-truth facts. In total there are 122 consequences annotated. Using perception only [8], due to the decomposition errors (such as the red font ones in Fig. 6.4) the system can detect 91 consequences correctly, yielding a 74% correct rate. After applying the four Axioms and reasoning, our system is able to detect 105 consequences correctly, yielding a 86% correct rate. Overall, this is a 15.4% of improvement.

Here we want to mention a caveat: there are definitely other common sense Axioms that we are not able to address in the current implementation. However, from the case studies presented, we can see that using the presented formal framework, our system is able to reason about manipulation action goals instead of just observing what is happening visually. This capability is essential for intelligent agents to imitate action goals from observation.

Chapter 7: Procedural Learning: Robot Learning Manipulation Action Plans by "Watching" Unconstrained Videos from the World Wide Web

7.1 Introduction

The ability to learn actions from human demonstrations is one of the major challenges for the development of intelligent systems. Particularly, manipulation actions are very challenging, as there is large variation in the way they can be performed and there are many occlusions.

Our ultimate goal is to build a self-learning robot that is able to enrich its knowledge about fine grained manipulation actions by "watching" demo videos. In this work we explicitly model actions that involve different kinds of grasping, and aim at generating a sequence of atomic commands by processing unconstrained videos from the World Wide Web (WWW).

The robotics community has been studying perception and control problems of grasping for decades [36]. Recently, several learning based systems were reported that infer contact points or how to grasp an object from its appearance [37, 38]. However, the desired grasping type could be different for the same target object,

when used for different action goals. Traditionally, data about the grasp has been acquired using motion capture gloves or hand trackers, such as the model-based tracker of [32]. The acquisition of grasp information from video (without 3D information) is still considered very difficult because of the large variation in appearance and the occlusions of the hand from objects during manipulation.

Our premise is that actions of manipulation are represented at multiple levels of abstraction. At lower levels the symbolic quantities are grounded in perception, and at the high level a grammatical structure represents symbolic information (objects, grasping types, actions). With the recent development of deep neural network approaches, our system integrates a CNN based object recognition and a CNN based grasping type recognition module. The latter recognizes the subject's grasping type directly from image patches.

The grasp type is an essential component in the characterization of manipulation actions. Just from the viewpoint of processing videos, the grasp contains information about the action itself, and it can be used for prediction or as a feature for recognition. It also contains information about the beginning and end of action segments, thus it can be used to segment videos in time. If we are to perform the action with a robot, knowledge about how to grasp the object is necessary so the robot can arrange its effectors. For example, consider a humanoid with one parallel gripper and one vacuum gripper. When a power grasp is desired, the robot should select the vacuum gripper for a stable grasp, but when a precision grasp is desired, the parallel gripper is a better choice. Thus, knowing the grasping type provides information for the robot to plan the configuration of its effectors, or even the type

of effector to use.

In order to perform a manipulation action, the robot also needs to learn what tool to grasp and on what object to perform the action. Our system applies CNN based recognition modules to recognize the objects and tools in the video. Then, given the beliefs of the tool and object (from the output of the recognition), our system predicts the most likely action using language, by mining a large corpus using a technique similar to [25]. Putting everything together, the output from the lower level visual perception system is in the form of (LeftHand GraspType1 Object1 Action RightHand GraspType2 Object2). We will refer to this septet of quantities as *visual sentence*.

At the higher level of representation, we generate a symbolic command sequence. [30] proposed a context-free grammar and related operations to parse manipulation actions. However, their system only processed RGBD data from a controlled lab environment. Furthermore, they did not consider the grasping type in the grammar. This work extends [30] by modeling manipulation actions using a probabilistic variant of the context free grammar, and explicitly modeling the grasping type.

Using as input the belief distributions from the CNN based visual perception system, a Viterbi probabilistic parser is used to represent actions in form of a hierarchical and recursive tree structure. This structure innately encodes the order of atomic actions in a sequence, and forms the basic unit of our knowledge representation. By reverse parsing it, our system is able to generate a sequence of atomic commands in predicate form, i.e. as $Action(Subject, Patient)$ plus the temporal

information necessary to guide the robot. This information can then be used to control the robot effectors [139].

Our contributions are twofold. (1) A convolutional neural network (CNN) based method has been adopted to achieve state-of-the-art performance in grasping type classification and object recognition on unconstrained video data; (2) a system for learning information about human manipulation action has been developed that links lower level visual perception and higher level semantic structures through a probabilistic manipulation action grammar.

7.2 Related Works

Most work on learning from demonstrations in robotics has been conducted in fully controlled lab environments [14]. Many of the approaches rely on RGBD sensors [18], motion sensors [107, 133] or specific color markers [19]. The proposed systems are fragile in real world situations. Also, the amount of data used for learning is usually quite small. It is extremely difficult to learn automatically from data available on the internet, for example from unconstrained cooking videos from Youtube. The main reason is that the large variation in the scenery will not allow traditional feature extraction and learning mechanism to work robustly.

At the high level, a number of studies on robotic manipulation actions have proposed ways on how instructions are stored and analyzed, often as sequences. Work by [163], among others, investigates how to compare sequences in order to reason about manipulation actions using sequence alignment methods, which bor-

row techniques from informatics. This chapter proposes a more detailed representation of manipulation actions, the *grammar trees*, extending earlier work. Chomsky in [148] suggested that a minimalist generative grammar, similar to the one of human language, also exists for action understanding and execution. The works closest related to this chapter are [17, 18, 30, 31]. [17] first discussed a Chomskyan grammar for understanding complex actions as a theoretical concept, [18] provided an implementation of such a grammar using as perceptual input only objects. [30] proposed a set of context-free grammar rules for manipulation action understanding. However, their system used data collected in a lab environment. Here we process unconstrained data from the internet. In order to deal with the noisy visual data, we extend the manipulation action grammar and adapt the parsing algorithm.

The recent development of deep neural networks based approaches revolutionized visual recognition research. The work presented in this chapter shows that with the recent developments of deep neural networks in computer vision, it is possible to learn manipulation actions from unconstrained demonstrations using CNN based visual perception.

7.3 Our Approach

We developed a system to learn manipulation actions from unconstrained videos. The system takes advantage of: (1) the robustness from CNN based visual processing; (2) the generality of an action grammar based parser. Figure7.1 shows our integrated approach.

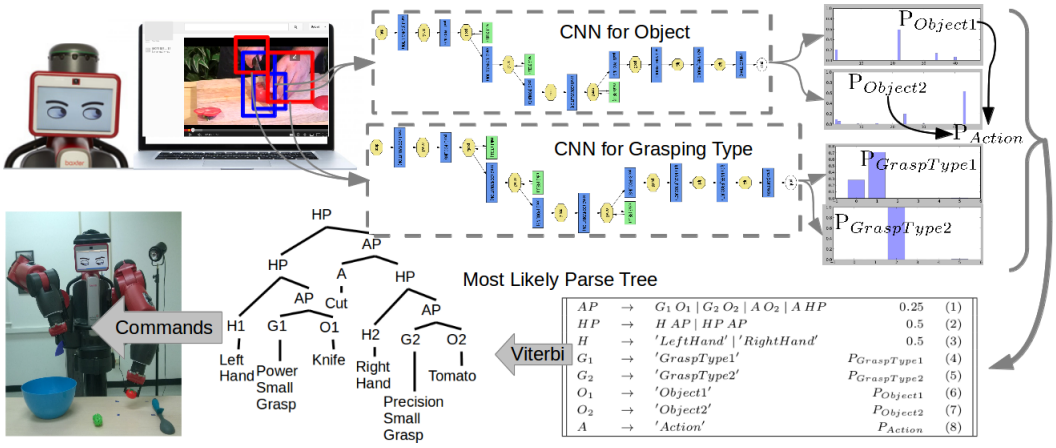


Figure 7.1: The integrated system reported in this work.

7.3.1 CNN based visual recognition

The system consists of two visual recognition modules, one for classification of grasping types and the other for recognition of objects. In both modules we used convolutional neural networks as classifiers. First, we briefly summarize the basic concepts of Convolutional Neural Networks, and then we present our implementations.

7.3.1.1 Convolutional Neural Network

(CNN) is a multilayer learning framework, which may consist of an input layer, a few convolutional layers and an output layer. The goal of CNN is to learn a hierarchy of feature representations. Response maps in each layer are convolved with a number of filters and further down-sampled by pooling operations. These pooling operations aggregate values in a smaller region by downsampling functions including max, min, and average sampling. The learning in CNN is based on Stochastic

Gradient Descent (SGD), which includes two main operations: Forward and Back-Propagation. Please refer to [49] for details.

We used a seven layer CNN (including the input layer and two perception layers for regression output). The first convolution layer has 32 filters of size 5×5, the second convolution layer has 32 filters of size 5×5, and the third convolution layer has 64 filters of size 5×5, respectively. The first perception layer has 64 regression outputs and the final perception layer has 6 regression outputs. Our system considers 6 grasping type classes.

7.3.1.2 Grasping Type Recognition

A number of grasping taxonomies have been proposed in several areas of research, including robotics, developmental medicine, and biomechanics, each focusing on different aspects of action. In a recent survey [47] reported 45 grasp types in the literature, of which only 33 were found valid. In this work, we use a categorization into six grasping types. First we distinguish, according to the most commonly used classification (based on functionality) into power and precision grasps [48]. Power grasping is used when the object needs to be held firmly in order to apply force, such as "grasping a knife to cut"; precision grasping is used in order to do fine grain actions that require accuracy, such as "pinch a needle". We then further distinguish among the power grasps, whether they are spherical, or otherwise (usually cylindrical), and we distinguish the latter according to the grasping diameter, into large diameter and small diameter ones. Similarly, we distinguish the precision grasps

into large and small diameter ones. Additionally, we also consider a Rest position (no grasping performed). Table 7.1 illustrates our grasp categories. We denote the list of these six grasps as G in the remainder of the chapter.

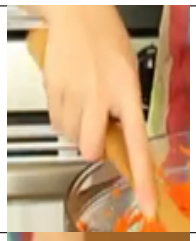
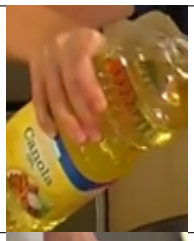
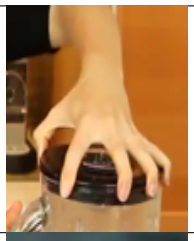
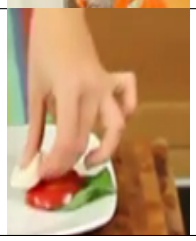
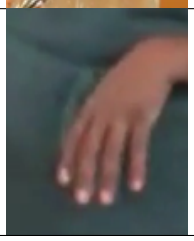
Grasping Types	Small Diameter	Large Diameter	Spherical & Rest
Power			
Precision			

Table 7.1: The list of the grasping types.

The input to the grasping type recognition module is a gray-scale image patch around the target hand performing the grasping. We resize each patch to 32×32 pixels, and subtract the global mean obtained from the training data.

For each testing video with M frames, we pass the target hand patches (left hand and right hand, if present) frame by frame, and we obtain an output of size $6 \times M$. We sum it up along the temporal dimension and then normalize the output. We use the classification for both hands to obtain (GraspType1) for the left hand, and (GraspType2) for the right hand. For the video of M frames the grasping type recognition system outputs two belief distributions of size 6×1: $P_{GraspType1}$ and $P_{GraspType2}$.

7.3.1.3 Object Recognition and Corpus Guided Action Prediction

The input to the object recognition module is an RGB image patch around the target object. We resize each patch to $32 \times 32 \times 3$ pixels, and we subtract the global mean obtained from the training data.

Similar to the grasping type recognition module, we also used a seven layer CNN. The network structure is the same as before, except that the final perception layer has 48 regression outputs. Our system considers 48 object classes, and we denote this candidate object list as O in the rest of the chapter. Table 7.2 lists the object classes.

apple, blender, bowl, bread, brocolli, brush, butter, carrot, chicken, chocolate, corn, creamcheese, croutons, cucumber, cup, doughnut, egg, fish, flour, fork, hen, jelly, knife, lemon, lettuce, meat, milk, mustard, oil, onion, pan, peanutbutter, pepper, pitcher, plate, pot, salmon, salt, spatula, spoon, spreader, steak, sugar, tomato, tongs, turkey, whisk, yogurt.

Table 7.2: The list of the objects considered in our system.

For each testing video with M frames, we pass the target object patches frame by frame, and get an output of size $48 \times M$. We sum it up along the temporal dimension and then normalize the output. We classify two objects in the image: (Object1) and (Object2). At the end of classification, the object recognition system outputs two belief distributions of size 48×1: $P_{Object1}$ and $P_{Object2}$.

We also need the 'Action' that was performed. Due to the large variations

in the video, the visual recognition of actions is difficult. Our system bypasses this problem by using a trained language model. The model predicts the most likely verb (Action) associated with the objects (Object1, Object2). In order to do prediction, we need a set of candidate actions V. Here, we consider the top 10 most common actions in cooking scenarios. They are (Cut, Pour, Transfer, Spread, Grip, Stir, Sprinkle, Chop, Peel, Mix). The same technique, used here, was used before on a larger set of candidate actions [25].

We compute from the Gigaword corpus [76] the probability of a verb occurring, given the detected nouns, $P(Action|Object1, Object2)$. We do this by computing the log-likelihood ratio [102] of trigrams (Object1, Action, Object2), computed from the sentence in the English Gigaword corpus [76]. This is done by extracting only the words in the corpus that are defined in O and V (including their synonyms). This way we obtain a reduced corpus sequence from which we obtain our target trigrams. The log-likelihood ratios computed for all possible trigrams are then normalized to obtain $P(Action|Object1, Object2)$. For each testing video, we can compute a belief distribution over the candidate action set V of size 10×1 as :

$$P_{Action} = \sum_{Object1 \in O} \sum_{Object2 \in O} P(Action|Object1, Object2) \times P_{Object1} \times P_{Object2}. \quad (7.1)$$

7.3.2 From Recognitions to Action Trees

The output of our visual system are belief distributions of the object categories, grasping types, and actions. However, they are not sufficient for executing actions. The robot also needs to understand the hierarchical and recursive structure of the

action. We argue that grammar trees, similar to those used in linguistics analysis, are a good representation capturing the structure of actions. Therefore we integrate our visual system with a manipulation action grammar based parsing module [30]. Since the output of our visual system is probabilistic, we extend the grammar to a probabilistic one and apply the Viterbi probabilistic parser to select the parse tree with the highest likelihood among the possible candidates.

7.3.2.1 Manipulation Action Grammar

We made two extensions from the original manipulation grammar [30]: (i) Since grasping is conceptually different from other actions, and our system employs a CNN based recognition module to extract the model grasping type, we assign an additional nonterminal symbol G to represent the grasp. (ii) To accommodate the probabilistic output from the processing of unconstrained videos, we extend the manipulation action grammar into a probabilistic one.

The design of this grammar is motivated by three observations: (i) Hands are the main driving force in manipulation actions, so a specialized nonterminal symbol H is used for their representation; (ii) an action (A) or a grasping (G) can be applied to an object (O) directly or to a hand phrase (HP), which in turn contains an object (O), as encoded in Rule (1), which builds up an action phrase (AP); (iii) an action phrase (AP) can be combined either with the hand (H) or a hand phrase, as encoded in rule (2), which recursively builds up the hand phrase. The rules discussed in Table 7.3 form the syntactic rules of the grammar.

To make the grammar probabilistic, we first treat each sub-rule in rules (1) and (2) equally, and assign equal probability to each sub-rule. With regard to the hand H in rule (3), we only consider a robot with two effectors (arms), and assign equal probability to 'LeftHand' and 'RightHand'. For the terminal rules (4-8), we assign the normalized belief distributions ($P_{Object1}, P_{Object2}, P_{GraspType1}, P_{GraspType2}, P_{Action}$) obtained from the visual processes to each candidate object, grasping type and action.

AP	$\rightarrow$	$G_1\ O_1 \mid G_2\ O_2 \mid A\ O_2 \mid A\ HP$	0.25	(1)
HP	$\rightarrow$	$H\ AP \mid HP\ AP$	0.5	(2)
H	$\rightarrow$	$'LeftHand' \mid 'RightHand'$	0.5	(3)
G_1	$\rightarrow$	$'GraspType1'$	$P_{GraspType1}$	(4)
G_2	$\rightarrow$	$'GraspType2'$	$P_{GraspType2}$	(5)
O_1	$\rightarrow$	$'Object1'$	$P_{Object1}$	(6)
O_2	$\rightarrow$	$'Object2'$	$P_{Object2}$	(7)
A	$\rightarrow$	$'Action'$	P_{Action}	(8)

Table 7.3: A Probabilistic Extension of Manipulation Action Context-Free Grammar.

7.3.2.2 Parsing and tree generation

We use a bottom-up variation of the probabilistic context-free grammar parser that uses dynamic programming (best-known as Viterbi parser [164]) to find the most likely parse for an input visual sentence. The Viterbi parser parses the visual sentence by filling in the most likely constituent table, and the parser uses the grammar introduced in Table 7.3. For each testing video, our system outputs the most likely parse tree of the specific manipulation action. By reversely parsing the

tree structure, the robot could derive an action plan for execution. Figure 7.3 shows sample output trees, and Table 7.4 shows the final control commands generated by reverse parsing.

7.4 Experiments

The theoretical framework we have presented suggests two hypotheses that deserve empirical tests: (a) the CNN based object recognition module and the grasping type recognition module can robustly recognize input frame patches from unconstrained videos into correct class labels; (b) the integrated system using the Viterbi parser with the probabilistic extension of the manipulation action grammar can generate a sequence of execution commands robustly.

To test the two hypotheses empirically, we need to define a set of performance variables and how they relate to our predicted results. The first hypothesis relates to visual recognition, and we can empirically test it by measuring the precision and recall metrics by comparing the detected object and grasping type labels with the ground truth ones. The second hypothesis relates to execution command generation, and we can also empirically test it by comparing the generated command predicates with the ground truth ones on testing videos. To validate our system, we conducted experiments on an extended version of a publicly available unconstrained cooking video dataset (YouCook) [53].

7.4.1 Dataset and experimental settings

Cooking is an activity, requiring a variety of manipulation actions, that future service robots most likely need to learn. We conducted our experiments on a publicly available cooking video dataset collected from the WWW and fully labeled, called the Youtube cooking dataset (YouCook) [53]. The data was prepared from 88 open-source Youtube cooking videos with unconstrained third-person view. Frame-by-frame object annotations are provided for 49 out of the 88 videos. These features make it a good empirical testing bed for our hypotheses.

We conducted our experiments using the following protocols: (1) 12 video clips, which contain one typical kitchen action each, are reserved for testing; (2) all other video frames are used for training; (3) we randomly reserve 10% of the training data as validation set for training the CNNs.

For training the grasping type, we extended the dataset by annotating image patches containing hands in the training videos. The image patches were converted to gray-scale and then resized to 32×32 pixels. The training set contains 1525 image patches and was labeled with the six grasping types. We used a GPU based CNN implementation [54] to train the neural network, following the structures described.

For training the object recognition CNN, we first extracted annotated image patches from the labeled training videos, and then resized them to $32 \times 32 \times 3$. We used the same GPU based CNN implementation to train the neural network, following the structures described above.

For localizing hands on the testing data, we first applied the hand detector

from [2] and picked the top two hand patch proposals (left hand and right hand, if present). For objects, we trained general object detectors from labeled training data using techniques from [165]. Furthermore we associated candidate object patches with the left or right hand, respectively depending on which had the smaller Euclidean distance.

7.4.2 Grasping Type and Object Recognition

On the reserved 10% validation data, the grasping type recognition module achieved an average precision of 77% and an average recall of 76%. On the reserved 10% validation data, the object recognition module achieved an average precision of 93%, and an average recall of 93%. Figure 7.2 shows the confusion matrices for grasping type and object recognition, respectively. From the figure we can see the robustness of the recognition.

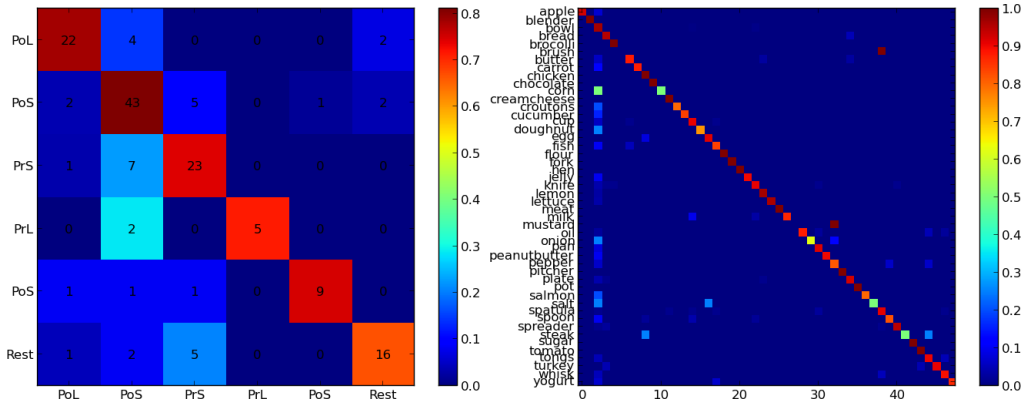


Figure 7.2: Confusion matrices. Left: grasping type; Right: object.

The performance of the object and grasping type recognition modules is also reflected in the commands that our system generated from the testing videos. We

observed an overall recognition accuracy of 79% on objects, of 91% on grasping types and of 83% on predicted actions (see Table 7.4). It is worth mentioning that in the generated commands the performance in the recognition of object drops, because some of the objects in the testing sequences do not have training data, such as "Tofu". The performance in the classification of grasping type goes up, because we sum up the grasping types belief distributions over the frames, which helps to smooth out wrong labels. The performance metrics reported here empirically support our hypothesis (a).

7.4.3 Visual Sentence Parsing and Commands Generation for Robots

Following the probabilistic action grammar from Table 7.3, we built upon the implementation of the Viterbi parser from the Natural Language Processing Kit [166] to generate the single most likely parse tree from the probabilistic visual sentence input. Figure 7.3 shows the sample visual processing outputs and final parse trees obtained using our integrated system. Table 7.4 lists the commands generated by our system on the reserved 12 testing videos, shown together with the ground truth commands (LH:LeftHand; RH: RightHand; PoS: Power-Small; PoL: Power-Large; PoP: Power-Spherical; PrS: Precision-Small; PrL: Precision-Large). The overall percentage of correct commands is 68%. Note, that we considered a command predicate wrong, if any of the object, grasping type or action was recognized incorrectly. The performance metrics reported here, empirically support our hypothesis (b).

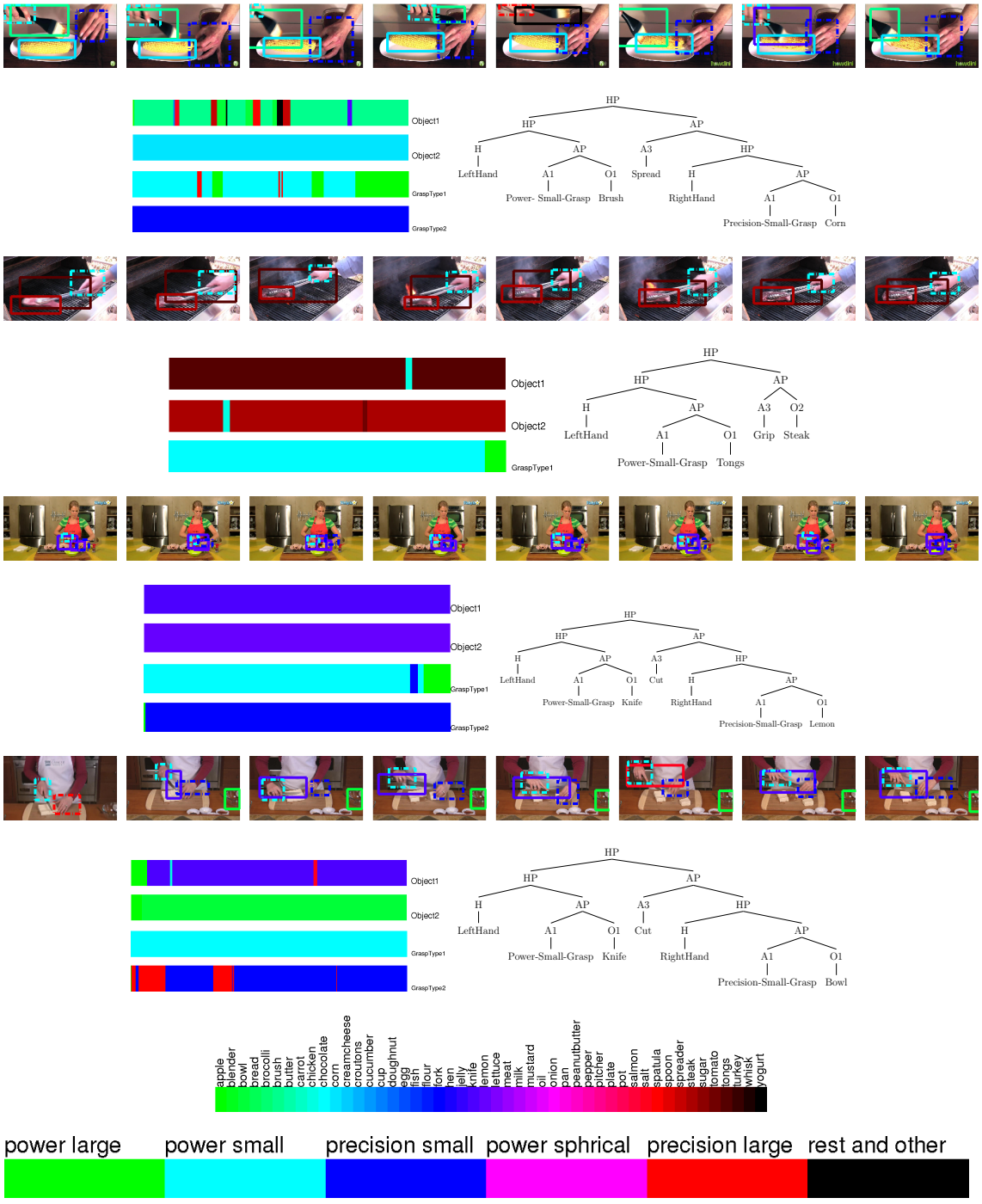


Figure 7.3: Upper row: input unconstrained video frames; Lower left: color coded (see lengend at the bottom) visual recognition output frame by frame along timeline; Lower right: the most likely parse tree generated for each clip.

Snapshot	Ground Truth Commands	Learned Commands
	Grasp_PoS(LH, Knife) Grasp_PrS(RH, Tofu) Action_Cut(Knife, Tofu)	Grasp_PoS(LH, Knife) Grasp_PrS(RH, Bowl) Action_Cut(Knife, Bowl)
	Grasp_PoS(LH, Blender) Grasp_PrL(RH, Bowl) Action_Blend(Blender, Bowl)	Grasp_PoS(LH, Bowl) Grasp_PoL(RH, Bowl) Action_Pour(Bowl, Bowl)
	Grasp_PoS(LH, Tongs) Action_Grip(Tongs, Chicken)	Grasp_PoS(LH, Chicken) Action_Cut(Chicken, Chicken)
	Grasp_PoS(LH, Brush) Grasp_PrS(RH, Corn) Action_Spread(Brush, Corn)	Grasp_PoS(LH, Brush) Grasp_PrS(RH, Corn) Action_Spread(Brush, Corn)
	Grasp_PoS(LH, Tongs) Action_Grip(Tongs, Steak)	Grasp_PoS(LH, Tongs) Action_Grip(Tongs, Steak)
	Grasp_PoS(LH, Spreader) Grasp_PrL(RH, Bread) Action_Spread(Spreader, Bread)	Grasp_PoS(LH, Spreader) Grasp_PrL(RH, Bowl) Action_Spread(Spreader, Bowl)
	Grasp_PoL(LH, Mustard) Grasp_PrS(RH, Bread) Action_Spread(Mustard, Bread)	Grasp_PoL(LH, Mustard) Grasp_PrS(RH, Bread) Action_Spread(Mustard, Bread)
	Grasp_PoS(LH, Spatula) Grasp_PrS(RH, Bowl) Action_Stir(Spatula, Bowl)	Grasp_PoS(LH, Spatula) Grasp_PrS(RH, Bowl) Action_Stir(Spatula, Bowl)
	Grasp_PoL(LH, Pepper) Grasp_PoL(RH, Pepper) Action_Sprinkle(Pepper, Bowl)	Grasp_PoL(LH, Pepper) Grasp_PoL(RH, Pepper) Action_Sprinkle(Pepper, Pepper)
	Grasp_PoS(LH, Knife) Grasp_PrS(RH, Lemon) Action_Cut(Knife, Lemon)	Grasp_PoS(LH, Knife) Grasp_PrS(RH, Lemon) Action_Cut(Knife, Lemon)
	Grasp_PoS(LH, Knife) Grasp_PrS(RH, Broccoli) Action_Cut(Knife, Broccoli)	Grasp_PoS(LH, Knife) Grasp_PoL(RH, Broccoli) Action_Cut(Knife, Broccoli)
	Grasp_PoS(LH, Whisk) Grasp_PrL(RH, Bowl) Action_Stir(Whisk, Bowl)	Grasp_PoS(LH, Whisk) Grasp_PrL(RH, Bowl) Action_Stir(Whisk, Bowl)
Overall Recognition Accuracy	Object: 79% Grasping type: 91% Action: 83%	Overall percentage of correct commands: 68%

Table 7.4: Incorrect entities learned are marked in red.

7.4.4 Discussion

The performance metrics reported in the experiment section empirically support our hypotheses that: (1) our system is able to robustly extract visual sentences with high accuracy; (2) our system can learn atomic action commands with few errors compared to the ground-truth commands. We believe this preliminary integrated system raises hope towards a fully intelligent robot for manipulation tasks that can automatically enrich its own knowledge resource by "watching" recordings from the World Wide Web.

Chapter 8: Concluding Remarks and Future Work

8.1 Concluding Remarks

In Chapter. 2, the experiments produced three results: (i) we achieved in average 59% accuracy using the CNN based method for grasp type recognition from unconstrained image patches; (ii) we achieved in average 65% prediction accuracy in inferring human intention using the grasp type only; (iii) using the grasp type temporal evolution, we achieved 78% recall and 80% precision in fine grain manipulation action segmentation tasks. Overall, the empirical results support our hypotheses (a-c) respectively. Recognizing grasp type and its use in inference for human action intention and fine level segmentation of human manipulation actions, are novel problems in computer vision. We have proposed a CNN based learning framework to address these problems with decent success. We hope our contributions can help advance the field of static scene understanding and human action fine level analysis, and we hope that they can be useful to other researchers in other applications. Additionally, we augmented a currently available hand data set and a cooking data set with grasp type labels, and provided human action intention labels for a subset of them, for future research.

Chapter. 3 has shown a principled approach of integrating large scale lan-

guage corpora for the purpose of action recognition in videos involving hand-tools. We validated our approach in both supervised and unsupervised scenarios and outperformed the current state-of-the-art STIP+BoW features significantly. These results demonstrate the strength of using language which encodes the intrinsic relationships between tools and actions, leveraging it to aid in the action recognition task. In this chapter, we also have introduced a computationally feasible framework that integrates visual perception together with semantic grounding obtained from a large textual corpus for the purpose of generating a descriptive sentence of an image. Experimental results show that our approach produces sentences that are both relevant and readable.

A system for detecting action consequences and classifying videos of manipulation action according to action consequences has been proposed in Chapter. 4. A dataset has been provided, which includes both data that we collected and eligible manipulation action video sequences from other publicly available datasets. Experimental results were performed that validate our method, and at the same time point out several weaknesses for future improvement.

In Chapter. 5, we presented a cognitive system for understanding human manipulation actions. The system integrates vision modules that ground semantically meaningful events in perceptual input with a reasoning module based on a context-free grammar and associated parsing algorithms, which dynamically build the sequence of structural representations. Experimental results showed that the cognitive system can extract the key events from the raw input and can interpret the observations by generating a sequence of tree structures.

In Chapter.6 we presented a formal computational framework for modeling manipulation actions based on a Combinatory Categorial Grammar. An empirical study on a large manipulation action dataset validates that 1) with the introduced formalism, a learning system can be devised to deduce the semantic meaning of manipulation actions in λ-schema; 2) with the learned schema and several common sense Axioms, our system is able to reason beyond just observation and deduce "hidden" action consequences, yielding a decent performance improvement.

In Chapter. 7 we presented an approach to learn manipulation action plans from unconstrained videos for cognitive robots. Two convolutional neural network based recognition modules (for grasping type and objects respectively), as well as a language model for action prediction, compose the lower level of the approach. The probabilistic manipulation action grammar based Viterbi parsing module is at the higher level, and its goal is to generate atomic commands in predicate form. We conducted experiments on a cooking dataset which consists of unconstrained demonstration videos. From the performance on this challenging dataset, we can conclude that our system is able to recognize and generate action commands robustly.

8.2 Future Work

Our experiments in Chapter. 2 indicate that there is still significant space for improving the recognition of grasp type and inference of human intention. We believe that advances in understanding high-level cognitive structure underlying human intention can help improve the performance. With the development of deep

learning systems and more data, we can also expect a robust grasp type recognition system beyond the seven categories used. Moreover, we believe that progress in natural language processing, such as mining the relationship between grasp type and actions, can advance high-level reasoning about human action intention to improve computer vision methods.

Our approach presented in Chapter. 3 goes beyond action recognition and can be extended to other vision problems faced in robotics with a more careful treatment of language [167]. Progress in the areas of object recognition, image segmentation and general scene understanding have been slow as these problems require *semantic grounding*. Language, when exploited properly, provides for this. For e.g. using shallow-parsing or Named-Entity Recognition to improve $\mathcal{P}_L$ predictions and subsequently $\mathcal{F}_d$; or performing a dependency parse to reduce the need to use synonyms to extract the tool and related verb from a sentence more accurately. An important limitation of our current approach is that we need to know in advance the action labels and tools of the video. We are currently working on approaches to discover, using *attributes* of the potential tool and action features obtained from the video, a prediction of the tool and action labels directly from language. The potential world-knowledge embedded in language, along with its complexities, was clearly demonstrated with Watson in Jeopardy! which set a milestone in AI. We believe it will do the same for vision and robotics in the near future.

There are instances where our strategy presented in Chapter. 3 fails to predict the appropriate verbs or nouns (see Fig. 3.18). This is due to the fact that object/scene detections can be wrong and noise from the corpus itself remains a

problem. Compared to human gold standards, therefore, much work still remains in terms of detecting these objects and scenes with high precision. Currently, at most two object classes are used to generate simple sentences which was shown in the results to have penalized the relevance score of our approach. This can be addressed by designing more complex HMMs to handle larger numbers of object and verb classes. Another interesting direction of future work would be to detect *salient* objects, learned from training image+corpus or eye-movement data, and to verify if these objects aid in improving the descriptive sentences we generate. Another potential application of representing images using $\mathcal{T}^*$ is that we can easily sort and retrieve images that are similar in terms of their *semantic* content. This would enable us to retrieve, for example, more relevant images given a verbal search query such as {`ride,sit,fly`}, returning images where these verbs are found in $\mathcal{T}^*$. Some results of retrieved images based on their verbal components are shown in Fig. 8.1: many images with dissimilar visual content are correctly classified based on their semantic meaning.

In Chapter. 4, to avoid the influence from the manipulating hands, especially occlusions caused by hands, a hand detection and segmentation algorithm can be applied. Then we can design a hallucination process to complete the contour of the occluded object under manipulation. Preliminary results are shown in Fig. 8.2. However, resolving the ambiguity between occlusion and deformation from visual analysis is a difficult task that requires further attention.

In Chapter. 5, since the grammar does not assume constraints such as the number of operators, it can be further adapted to process scenarios with multiple

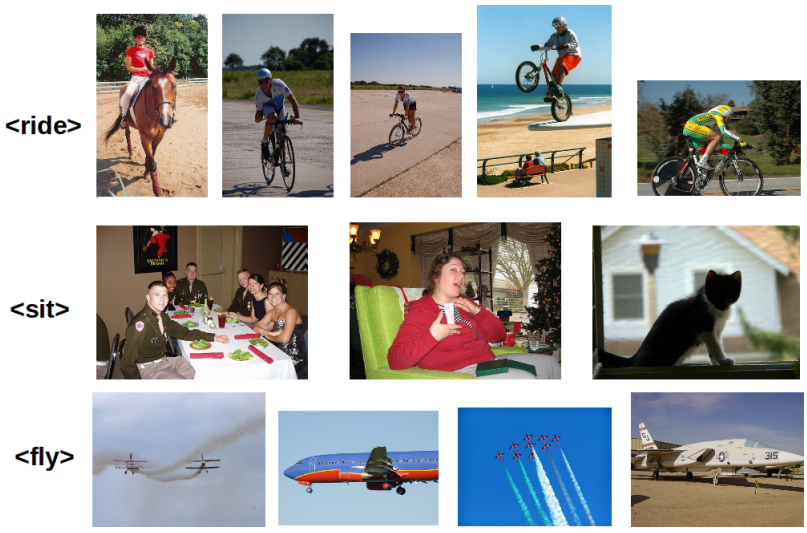


Figure 8.1: Images retrieved from 3 verbal search terms: `ride,sit,fly`.

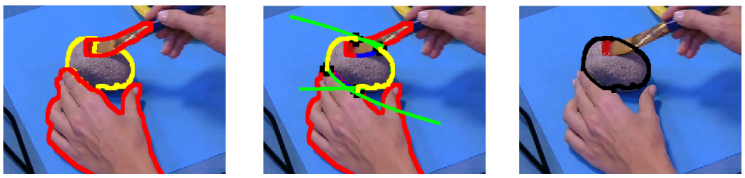
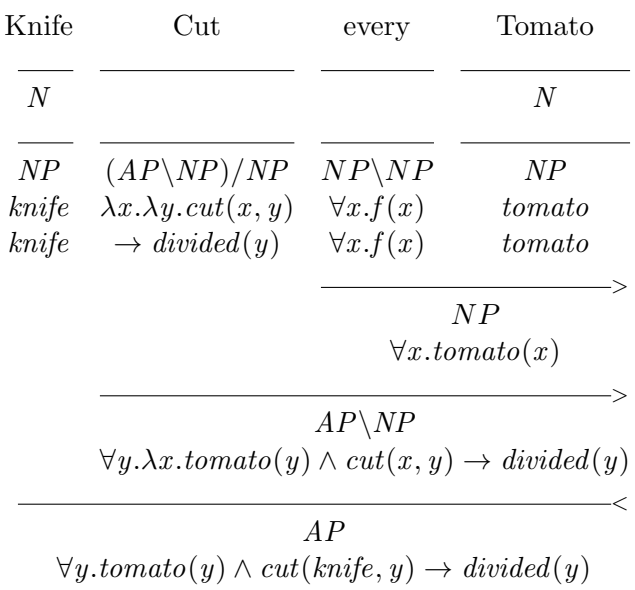


Figure 8.2: A hallucination process of contour completion (paint stone sequence in MAC 1.0). Left: original segments; Middle: contour hallucination with second order polynomials fitting (green lines); Right: final hallucinated contour.

agents doing complicated manipulation actions once the perception tools have been developed. Moreover, we also plan to investigate operations that enable the system to reason during observation. After the system observes a significant number of manipulation actions, it can build a database of all sequences of trees [168]. By querying this database, we expect the system to predict things such as which object will be manipulated next or which action will follow. Also, the action trees could be learned not only from observation but also from language resources, such as dictionaries, recipes, manuals etc. This link to computational linguistics constitutes an interesting avenue for future research.

In Chapter. 6, due to the limitation of current testing scenarios, we conducted experiments only considering a relatively small set of seed lexicon rules and logical expressions. Nevertheless, we want to mention that the presented CCG framework can also be extended to learn the formal logic representation of more complex manipulation action semantics. For example, the temporal order of manipulation actions can be modeled by considering a seed rule such as $AP\backslash AP : \lambda f.\lambda g.before(f(\cdot), g(\cdot))$, where $before(\cdot, \cdot)$ is a temporal predicate. For actions we consider seed main type $(AP\backslash NP)/NP$. For more general manipulation scenarios, based on whether the action is transitive or intransitive, the main types of action can be extended to include $AP\backslash NP$. Moreover, the logical expressions can also be extended to include universal quantification $\forall$ and existential quantification $\exists$. Thus, manipulation action such as "knife cut every tomato" can be parsed into a representation as $\forall x.tomato(x) \wedge cut(knife, x) \rightarrow divided(x)$ (the parse is given in the following chart). Here, the concept "every" has a main type of $NP\backslash NP$ and semantic mean-

ing of $\forall x.f(x)$. The same framework can also extended to have other combinatory rules such as **composition** and **type-raising** [154].



8.3 Final Remarks

The presented framework enables an intelligent agent to predict and reason action goals from observation, and thus has many potential applications such as human intention prediction, robot action policy planning and human robot collaboration. We believe that our formalism of manipulation actions bridges computational linguistics, vision and robotics, and opens further research in Artificial Intelligence and Robotics. As the robotics industry is moving towards robots that function safely, effectively and autonomously to perform tasks in real-world unstructured environments, they will need to be able to understand the meaning of manipulation actions and acquire human-like common-sense reasoning capabilities (please refer to [169, 170] for pilot studies of scene understanding with common-sense reasoning

and knowledge).

Some parts of this thesis have been integrated into a pilot robotic system running on a Baxter research humanoid to visually learn manipulation actions (such as making a drink) from observing human doing it[1]. These direct applications of the techniques presented in this thesis could potentially 1) reduce the costly reprogramming time to teach industrial or domestic robots new tasks, 2) increase the level of flexibility and adaptivity for current robotic systems, and 3) enrich robots' procedural knowledge through a continuous learning process.

[1] A live recording of the system could be found at http://www.umiacs.umd.edu/~yzyang/WaitWhatDemo

Bibliography

[1] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI Magazine*, 27(4):12, 2006.

[2] Arpit Mittal, Andrew Zisserman, and Philip HS Torr. Hand detection using multiple proposals. In *BMVC*, pages 1–11. Citeseer, 2011.

[3] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE, 2000.

[4] Pedro F. Felzenszwalb, Ross B. Girshick, David A. McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, 2010.

[5] Antonio Torralba, Kevin P. Murphy, William T. Freeman, and Mark A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, pages 273–280. IEEE Computer Society, 2003.

[6] Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1830–1837. IEEE, 2012.

[7] Ibrahim Adalbert Kapandji and LH Honoré. *The physiology of the joints*. Churchill Livingstone, Boca Raton, FL, 2007.

[8] E E. Aksoy and F. Wörgötter. Semantic decomposition and recognition of long and complex manipulation action sequences. *Computer Vision and Image Understading (CVIU)*, page Under Review, 2015.

[9] Giacomo Rizzolatti, Leonardo Fogassi, and Vittorio Gallese. Neurophysiological mechanisms underlying the understanding and imitation of action. *Nature Reviews Neuroscience*, 2(9):661–670, 2001.

[10] V Gazzola, G Rizzolatti, B Wicker, and C Keysers. The anthropomorphic brain: the mirror neuron system responds to human and robotic actions. *Neuroimage*, 35(4):1674–1684, 2007.

[11] Hedvig Kjellström, Javier Romero, David Martínez, and Danica Kragić. Simultaneous visual recognition of manipulation actions and manipulated objects. In *Computer Vision–ECCV 2008*, pages 336–349. Springer Berlin Heidelberg, 2008.

[12] Marcus Rohrbach, Anna Rohrbach, Michaela Regneri, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision*, pages 1–28, 2015.

[13] Bingbing Ni, Vignesh R Paramathayalan, and Philippe Moulin. Multiple granularity analysis for fine-grained action detection. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 756–763. IEEE, 2014.

[14] E.E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. Learning the semantics of object–action relations by observation. *The International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[15] E E. Aksoy, M. Tamosiunaite, and F. Wörgötter. Model-free incremental learning of the semantics of manipulation actions. *Robotics and Autonomous Systems*, pages 1–42, 2014.

[16] Karinne Ramirez-Amaro, Michael Beetz, and Gordon Cheng. Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. *Artificial Intelligence*, pages –, 2015.

[17] K. Pastra and Y. Aloimonos. The minimalist grammar of action. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1585):103–117, 2012.

[18] D. Summers-Stay, C.L. Teo, Y. Yang, C. Fermüller, and Y. Aloimonos. Using a minimal action grammar for activity understanding in the real world. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4104–4111, Vilamoura, Portugal, 2013. IEEE.

[19] Kyuhwa Lee, Yanyu Su, Tae-Kyun Kim, and Yiannis Demiris. A syntactic approach to robot imitation learning using probabilistic activity grammars. *Robotics and Autonomous Systems*, 61(12):1323–1334, 2013.

[20] N. Siddharth, A. Barbu, and J. M. Siskind. Seeing unseeability to see the unseeable. *Advances in Cognitive Systems (ACS)*, 2:77–94, December 2012.

[21] Yiannis Aloimonos. *Active perception*. Psychology Press, 2013.

[22] N. Chomsky. *Syntactic Structures*. Mouton de Gruyter, 1957.

[23] Yezhou Yang, Cornelia Fermuller, Yi Li, and Yiannis Aloimonos. Grasp type revisited: A modern perspective on a classical feature for vision. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015.

[24] C.L. Teo, Y. Yang, H. Daume, C. Fermüuller, and Y. Aloimonos. Towards a watson that sees: Language-guided action recognition for robots. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation*, pages 374–381, Saint Paul, MN, 2012. IEEE.

[25] Yezhou Yang, Ching Lik Teo, Hal Daumé III, and Yiannis Aloimonos. Corpus-guided sentence generation of natural images. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 444–454. Association for Computational Linguistics, 2011.

[26] Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Detection of manipulation action consequences (MAC). In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2563–2570, Portland, OR, 2013. IEEE.

[27] Yezhou Yang, Cornelia Fermüller, Yiannis Aloimonos, and Anupam Guha. A cognitive system for understanding human manipulation actions. *Advances in Cognitive Systems*, 3:67–86, 2014.

[28] Yezhou Yang, Cornelia Fermuller, Yiannis Aloimonos, and Eren Erdal Aksoy. Learning the semantics of manipulation action. In *The 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2015.

[29] Yezhou Yang, Yi Li, Cornelia Fermuller, and Yiannis Aloimonos. Robot learning manipulation action plans by "watching" unconstrained videos from the world wide web. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[30] Y. Yang, C. Fermuller, and Y. Aloimonos. A cognitive system for human manipulation action understanding. In *the Second Annual Conference on Advances in Cognitive Systems (ACS)*, 2013.

[31] Anupam Guha, Yezhou Yang, Cornelia Fermüller, and Yiannis Aloimonos. Minimalist plans for interpreting manipulation actions. *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5908–5914, 2013.

[32] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using Kinect. In *Proceedings of the 2011 British Machine Vision Conference*, pages 1–11, Dundee, UK, 2011. BMVA.

[33] Raoul Tubiana, Jean-Michel Thomine, and Evelyn Mackin. *Examination of the Hand and the Wrist*. CRC Press, 1998.

[34] Zhenyao Mo and Ulrich Neumann. Real-time hand pose recognition using low-resolution depth images. In *CVPR (2)*, pages 1499–1505, 2006.

[35] Xenophon Zabulis, Haris Baltzakis, and Antonis Argyros. Vision-based hand gesture recognition for human-computer interaction. *The Universal Access Handbook. LEA*, 2009.

[36] Karun B. Shimoga. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research*, 15(3):230–266, 1996.

[37] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[38] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *International Journal of Robotics Research*, page to appear, 2014.

[39] Dan Xie, Sinisa Todorovic, and Song-Chun Zhu. Inferring "dark matter" and "dark energy" from videos. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2224–2231. IEEE, 2013.

[40] Jungseock Joo, Weixin Li, Francis F Steen, and Song-Chun Zhu. Visual persuasion: Inferring communicative intents of images. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 216–223. IEEE, 2014.

[41] Hamed Pirsiavash, Carl Vondrick, and Antonio Torralba. Inferring the why in images. *arXiv preprint arXiv:1406.5472*, 2014.

[42] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[43] 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.

[44] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[45] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS 2012*, 2013.

[46] Dan Claudiu Ciresan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *CVPR 2012*, 2012.

[47] T. Feix, J. Romero, C. H. Ek, H Schmiedmayer, and D. Kragic. A Metric for Comparing the Anthropomorphic Motion Capability of Artificial Hands. *Robotics, IEEE Transactions on*, 29(1):82–93, February 2013.

[48] Marc Jeannerod. The timing of natural prehension movements. *Journal of motor behavior*, 16(3):235–254, 1984.

[49] Yann LeCun and Yoshua Bengio. The handbook of brain theory and neural networks. chapter Convolutional networks for images, speech, and time series, pages 255–258. MIT Press, Cambridge, MA, USA, 1998.

[50] Felix Warneken and Michael Tomasello. Altruistic helping in human infants and young chimpanzees. *science*, 311(5765):1301–1303, 2006.

[51] Dan Song, Nikolaos Kyriazis, Iason Oikonomidis, Chavdar Papazov, Antonis Argyros, Darius Burschka, and Danica Kragic. Predicting human intention in visual observations of hand/object interactions. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1608–1615. IEEE, 2013.

[52] John R Napier. The prehensile movements of the human hand. *Journal of bone and joint surgery*, 38(4):902–913, 1956.

[53] P. Das, C. Xu, R. F. Doell, and J. J. Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[54] Yangqing Jia. Caffe: An open source convolutional architecture for fast feature embedding. http://caffe.berkeleyvision.org/, 2013.

[55] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57:469–483, May 2009.

[56] W.R. Schwartz, A. Kembhavi, D. Harwood, and L.S. Davis. Human detection using partial least squares analysis. In *International Conference on Computer Vision*, 2009.

[57] Ivan Laptev and Tony Lindeberg. Space-time interest points. In *ICCV*, pages 432–439. IEEE Computer Society, 2003.

[58] Ross Messing, Chris Pal, and Henry Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV '09: Proceedings of the Twelfth IEEE International Conference on Computer Vision*, Washington, DC, USA, 2009. IEEE Computer Society.

[59] Pavan K. Turaga, Rama Chellappa, V. S. Subrahmanian, and Octavian Udrea. Machine recognition of human activities: A survey. *IEEE Trans. Circuits Syst. Video Techn.*, 18(11):1473–1488, 2008.

[60] Daniel Weinland, Remi Ronfard, and Edmond Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2010.

[61] Ana Paula Brando Lopes, Eduardo Alves do Valle, Jussara Marques de Almeida, and Arnaldo Albuquerque de Arajo. Action recognition in videos: from motion capture labs to the web. (arXiv:1006.3506), Jun 2010. Preprint submitted to CVIU.

[62] Benjamin Sapp, Alexander Toshev, and Ben Taskar. Cascaded models for articulated pose estimation. In *ECCV*, 2010.

[63] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, October 2005.

[64] A. Efros, A. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, pages 726–733, 2003.

[65] L. Gorelick, M. Blank, E. Shechtman, R. Basri, and M. Irani. Actions as space-time shapes. *PAMI*, 29(12):2247–2253, 2007.

[66] Alper Yilmaz and Mubarak Shah. Actions sketch: A novel action representation. In *CVPR*, pages 984–989, 2005.

[67] Alessandro Bissacco, Alessandro Chiuso, and Stefano Soatto. Classification and recognition of dynamical models: The role of phase, independent components, kernels and optimal transport. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1958–1972, November 2007.

[68] Rizwan Chaudhry, Avinash Ravichandran, Gregory Hager, and Rene Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *CVPR*, 2009.

[69] Pinar Duygulu, Kobus Barnard, Joo F. G. de Freitas, and David A. Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *ECCV (4)*, volume 2353 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2002.

[70] Abhinav Gupta and Larry S. Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (1)*, volume 5302 of *Lecture Notes in Computer Science*, pages 16–29. Springer, 2008.

[71] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, and David A. Forsyth. Who's in the picture? In *NIPS*, 2004.

[72] L. Jie, B. Caputo, and V. Ferrari. Who's doing what: Joint modeling of names and verbs for simultaneous face and pose annotation. In NIPS, editor, *Advances in Neural Information Processing Systems*, NIPS. NIPS, December 2009.

[73] T. Cour, C. Jordan, E. Miltsakaki, and B. Taskar. Movie/script: Alignment and parsing of video and text transcription. In *ECCV*. 2008.

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

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

[76] D. Graff. English gigaword. In *Linguistic Data Consortium, Philadelphia, PA*, 2003.

[77] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[78] Thomas Brox, Christoph Bregler, and Jitendra Malik. Large displacement optical flow. In *CVPR*, pages 41–48. IEEE, 2009.

[79] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, 2004.

[80] Christian Schuldt, Ivan Laptev, and Barbara Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.

[81] Leonid Sigal, Alexandru O. Balan, and Michael J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):4–27, 2010.

[82] Abhinav Gupta and Larry S. Davis. Objects in action: An approach for combining action understanding and object perception. In *CVPR*. IEEE Computer Society, 2007.

[83] F. De la Torre, J. Hodgins, J. Montano, S. Valcarcel, R. Forcada, and J. Macey. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Technical report, CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University, July 2009.

[84] Mark Hall, Eibe Frank, Geoffrey Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations, Volume 11, Issue 1*, 2009.

[85] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results, 2008.

[86] Cosimo Urgesi, Valentina Moro, Matteo Candidi, and Salvatore M Aglioti. Mapping implied body actions in the human motor system. *J Neurosci*, 26(30):7942–9, 2006.

[87] Zoe Kourtzi. But still, it moves. *Trends in Cognitive Sciences*, 8(2):47 – 49, 2004.

[88] Bangpeng Yao and Li Fei-Fei. Grouplet: a structured image representation for recognizing human and object interactions. In *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, June 2010.

[89] Weilong Yang, Yang Wang, and Greg Mori. Recognizing human actions from still images with latent poses. In *CVPR*, 2010.

[90] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001.

[91] P. Liang, M. I. Jordan, and D. Klein. Learning from measurements in exponential families. In *International Conference on Machine Learning (ICML)*, 2009.

[92] Gideon S. Mann and Andrew Mccallum. Simple, robust, scalable semi-supervised learning via expectation regularization. In *The 24th International Conference on Machine Learning*, 2007.

[93] A. Kojima, M. Izumi, T. Tamura, and K. Fukunaga. Generating natural language description of human behavior from video images. In *Pattern Recognition, 2000. Proceedings. 15th International Conference on*, volume 4, pages 728 –731 vol.4, 2000.

[94] Ali Farhadi, Seyyed Mohammad Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David A. Forsyth. Every picture tells a story: Generating sentences from images. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science*, pages 15–29. Springer, 2010.

[95] B.Z. Yao, Xiong Yang, Liang Lin, Mun Wai Lee, and Song-Chun Zhu. I2t: Image parsing to text description. *Proceedings of the IEEE*, 98(8):1485 –1508, aug. 2010.

[96] David Traum, Michael Fleischman, and Eduard Hovy. Nl generation for virtual humans in a complex social environment. In *In Proceedings of he AAAI Spring Symposium on Natural Language Generation in Spoken and Written Dialogue*, pages 151–158, 2003.

[97] Kathy McKeown. Query-focused summarization using text-to-text generation: When information comes from multilingual sources. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation (UCNLG+Sum 2009)*, page 3, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[98] Dave Golland, Percy Liang, and Dan Klein. A game-theoretic approach to generating spatial descriptions. In *Proceedings of EMNLP*, 2010.

[99] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/, 2008.

[100] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C. Weng. A note on platt's probabilistic outputs for support vector machines. *Mach. Learn.*, 68:267–276, October 2007.

[101] Jinho D. Choi and Martha Palmer. Robust constituent-to-dependency conversion for english. In *Proceedings of the 9th International Workshop on Treebanks and Linguistic Theories*, pages 55–66, Tartu, Estonia, 2010.

[102] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[103] Chin-Yew Lin and Eduard Hovy. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *NAACLHLT*, 2003.

[104] David Zajic Bonnie and Bonnie Dorr. Bbn/umd at duc-2004: Topiary. In *In Proceedings of the 2004 Document Understanding Conference (DUC 2004) at NLT/NAACL 2004*, pages 112–119, 2004.

[105] A.D. Jepson, D.J. Fleet, and T.F. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1296–1311, 2003.

[106] V. Papadourakis and A. Argyros. Multiple objects tracking in the presence of long-term occlusions. *Computer Vision and Image Understanding*, 114(7):835–846, 2010.

[107] Gutemberg Guerra-Filho, Cornelia fermüller, and Yiannis Aloimonos. Discovering a language for human activity. In *Proceedings of the AAAI 2005 Fall Symposium on Anticipatory Cognitive Embodied Systems*, Washington,DC, 2005. AAAI.

[108] T.B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006.

[109] P. Turaga, R. Chellappa, V.S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.

[110] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2):107–123, 2005.

[111] G. Willems, T. Tuytelaars, and L. Van Gool. An efficient dense and scale-invariant spatio-temporal interest point detector. *Proceedings of the 2008 IEEE European Conference on Computer Vision*, pages 650–663, 2008.

[112] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Proceedings of the 2005 IEEE International Conference on Computer Vision*, volume 2, pages 1395–1402, 2005.

[113] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. In *Proceedings of the 2005 IEEE Intenational Conference on Computer Vision and Pattern Recognition*, volume 1, pages 984–989, San Diego, CA, 2005. IEEE.

[114] A. Kale, A. Sundaresan, AN Rajagopalan, N.P. Cuntoor, A.K. Roy-Chowdhury, V. Kruger, and R. Chellappa. Identification of humans using gait. *IEEE Transactions on Image Processing*, 13(9):1163–1173, 2004.

[115] P. Saisan, G. Doretto, Y.N. Wu, and S. Soatto. Dynamic texture recognition. In *Proceedings of the 2001 IEEE Intenational Conference on Computer Vision and Pattern Recognition*, volume 2, pages 58–63, Kauai, HI, 2001. IEEE.

[116] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Proceedings of the 2009 IEEE Intenational Conference on Computer Vision and Pattern Recognition*, pages 1932–1939, Miami,FL, 2009. IEEE.

[117] I.S. Vicente, V. Kyrki, D. Kragic, and M. Larsson. Action recognition and understanding through motor primitives. *Advanced Robotics*, 21(15):1687–1707, 2007.

[118] M. Sridhar, A.G. Cohn, and D.C. Hogg. Learning functional object-categories from a relational spatio-temporal representation. In *Proceedings of the 2008 IEEE European Conference on Artificial Intelligence*, pages 606–610, 2008.

[119] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *Proceedings of the 2010 IEEE Intenational Conference on Computer Vision and Pattern Recognition*, pages 17–24, 2010.

[120] E.A. Locke and G.P. Latham. *A theory of goal setting & task performance.* Prentice-Hall, Inc, 1990.

[121] A. Mishra, C. Fermüller, and Y. Aloimonos. Active segmentation for robots. In *Proceedings of the 20009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3133–3139, St. Louis,MO, 2009. IEEE.

[122] B. Han, Y. Zhu, D. Comaniciu, and L.S. Davis. Visual tracking by continuous density propagation in sequential bayesian filtering framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):919–930, 2009.

[123] J.K. Tsotsos. Analyzing vision at the complexity level. *Behavioral and Brain Sciences*, 13(3):423–469, 1990.

[124] Y. Yang, M. Song, N. Li, J. Bu, and C. Chen. Visual attention analysis by pseudo gravitational field. In *Proceedings of the 2009 ACM International Conference on Multimedia*, pages 553–556. ACM, 2009.

[125] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[126] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of the 2001 IEEE International Conference on Computer Vision*, volume 2, pages 416–423, 2001.

[127] K. Nummiaro, E. Koller-Meier, L. Van Gool, et al. A color-based particle filter. In *First International Workshop on Generative-Model-Based Vision*, volume 2002, page 01, 2002.

[128] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *Proceedings of the 2004 IEEE European Conference on Computer Vision*, pages 25–36, 2004.

[129] C. Liu et al. *Beyond pixels: exploring new representations and applications for motion analysis*. PhD thesis, MIT, 2009.

[130] J. Neumann and etc. Localizing objects and actions in videos with the help of accompanying text. *Final Report, JHU summer workshop*, 2010.

[131] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Proceedings of the 2011 IEEE International conference on Robotics and Automation*, pages 1817–1824. IEEE, 2011.

[132] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *Advances in neural information processing systems*, pages 831–837, 2001.

[133] Yi Li, Cornelia Fermüller, Yiannis Aloimonos, and Hui Ji. Learning shift-invariant sparse representation of actions. In *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2630–2637, San Francisco,CA, 2010. IEEE.

[134] Dariu M Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999.

[135] Piotr Dollár, Vincent Rabaud, Garrison Cottrell, and Serge Belongie. Behavior recognition via sparse spatio-temporal features. In *2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72, San Diego,CA, 2005. IEEE.

[136] Liang Wang and David Suter. Learning and matching of dynamic shape manifolds for human action recognition. *IEEE Transactions on Image Processing*, 16(6):1646–1661, 2007.

[137] Jezekiel Ben-Arie, Zhiqian Wang, Purvin Pandit, and Shyamsundar Rajaram. Human activity recognition using multidimensional indexing. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(8):1091–1104, 2002.

[138] Changbo Hu, Qingfeng Yu, Yi Li, and Songde Ma. Extraction of parametric human model for posture recognition using genetic algorithm. In *Proceedings of the 2000 IEEE International Conference on Automatic Face and Gesture Recognition*, pages 518–523, Grenoble,France, 2000. IEEE.

[139] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[140] Georgios E Fainekos, Hadas Kress-Gazit, and George J Pappas. Hybrid controllers for path planning: A temporal logic approach. In *44th IEEE Conference on Decision and Control*, pages 4885–4890, Philadelphia,PA, 2005. IEEE.

[141] N. Dantam and M. Stilman. The motion grammar: Analysis of a linguistic method for robot control. *Transactions on Robotics*, 29(3):704–718, 2013.

[142] F Wörgötter, Eren Erdal Aksoy, N Kruger, Justus Piater, Ales Ude, and Minija Tamosiunaite. A simple ontology of manipulation actions based on hand-object relations. *IEEE Transactions on Autonomous Mental Development*, pages 117–134, 2012.

[143] Javad Mohamad Aein, Eren Erdal Aksoy, Minija Tamosiunaite, Jeremie Papon, Ales Ude, and Florentin Wörgötter. Toward a library of manipulation actions based on semantic object-action relations. In *Proceedings of the 2013 EEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4555–4562, Tokyo,Japan, 2013. IEEE.

[144] Matthew Brand. Understanding manipulation in video. In *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 94–99, Killington,VT, 1996. IEEE.

[145] Michael S Ryoo and Jake K Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *Proceedings of the 2006 IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1709–1718, New York City, NY, 2006. IEEE.

[146] Yuri A. Ivanov and Aaron F. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):852–872, 2000.

[147] Darnell Moore and Irfan Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *Proceedings of the National Conference on Artificial Intelligence*, pages 770–776, Menlo Park, CA, 2002. AAAI.

[148] Noam Chomsky. *Lectures on government and binding: The Pisa lectures*. Walter de Gruyter, 1993.

[149] Roger C Schank and Larry Tesler. A conceptual dependency parser for natural language. In *Proceedings of the 1969 conference on Computational linguistics*, pages 1–3, Sanga-Saby, Sweden, 1969. Association for Computational Linguistics.

[150] Vikram Manikonda, Perinkulam S Krishnaprasad, and James Hendler. *Languages, behaviors, hybrid architectures, and motion control*. Springer, New York, 1999.

[151] R Jackendorff. X-bar-syntax: A study of phrase structure. *Linguistic Inquiry Monograph*, 2, 1977.

[152] Daniel H Younger. Recognition and parsing of context-free languages in time n^3. *Information and control*, 10(2):189–208, 1967.

[153] M. Nishigaki, C. fermüller, and D. DeMenthon. The image torque operator: A new tool for mid-level vision. In *Proceedings of the 2012 IEEE International Conference on Computer Vision and Pattern Recogntion*, pages 502–509, Providence,RI, 2012. IEEE.

[154] Mark Steedman. Plans, affordances, and combinatory grammar. *Linguistics and Philosophy*, 25(5-6):723–753, 2002.

[155] Mark Steedman. *The syntactic process*, volume 35. MIT Press, 2000.

[156] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*, 2005.

[157] Luke S Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *EMNLP-CoNLL*, pages 678–687, 2007.

[158] Raymond J Mooney. Learning to connect language and perception. In *AAAI*, pages 1598–1601, 2008.

[159] Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine learning (ICML)*, 2011.

[160] Stefanie Tellex, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):151–167, 2014.

[161] Cynthia Matuszek, Liefeng Bo, Luke Zettlemoyer, and Dieter Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[162] Chitta Baral, Juraj Dzifcak, Marcos Alvarez Gonzalez, and Jiayu Zhou. Using inverse λ and generalization to translate english to formal languages. In *Proceedings of the Ninth International Conference on Computational Semantics*, pages 35–44. Association for Computational Linguistics, 2011.

[163] Moritz Tenorth, Johannes Ziegltrum, and Michael Beetz. Automated alignment of specifications of everyday manipulation tasks. In *IROS*. IEEE, 2013.

[164] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, pages 136–143. Association for Computational Linguistics, 1988.

[165] Ming-Ming Cheng, Ziming Zhang, Wen-Yan Lin, and Philip H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *IEEE CVPR*, 2014.

[166] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python.* " O'Reilly Media, Inc.", 2009.

[167] Yezhou Yang, Ching L Teo, Cornelia Fermuller, and Yiannis Aloimonos. Robots with language: Multi-label visual recognition using nlp. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4256–4262. IEEE, 2013.

[168] Yezhou Yang, Anupam Guha, Cornelia Fermüller, and Yiannis Aloimonos. Manipulation action tree bank: A knowledge resource for humanoids. In *IEEE/RAS International Conference on Humanoid Robots*, 2014.

[169] Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos. Visual common-sense for scene understanding using perception, semantic parsing and reasoning. In *The Twelfth International Symposium on Logical Formalization on Commonsense Reasoning*, 2015.

[170] Somak Aditya, Yezhou Yang, Chitta Baral, Cornelia Fermuller, and Yiannis Aloimonos. From images to sentences through scene description graphs using commonsense reasoning and knowledge. *arXiv preprint arXiv:1511.03292*, 2015.