UNDERSTANDING PEOPLE FROM RGBD DATA FOR ASSISTIVE ROBOTS

A Dissertation
Presented to the Faculty of the Graduate School
of Cornell University
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

by
Hema Swetha Koppula
February 2016

UNDERSTANDING PEOPLE FROM RGBD DATA FOR ASSISTIVE ROBOTS

Hema Swetha Koppula, Ph.D.

Cornell University 2016

Understanding people in complex dynamic environments is important for many applications such as robotic assistants, health-care monitoring systems, self driving cars, etc. This is a challenging problem as human actions and intents are not always observable and often contain large amounts of ambiguity. Moreover, human environments are complex with lots of objects and many possible ways of interacting with them. This leads to a huge variation in the way people perform various tasks.

The focus of this dissertation is to develop learning algorithms for understanding people and their environments from RGB-D data. We address the problems of labeling environments, detecting past activities and anticipating what will happen in the future. In order to enable agents operating in human environments to perform holistic reasoning, we need to jointly model the humans, objects and environments and capture the rich context between them.

We propose graphical models that naturally capture the rich spatio-temporal relations between human poses and objects in a 3D scene. We propose an efficient method to sample multiple possible graph structures and reason about the many alternate future possibilities. Our models also provide a functional representation of the environments, allowing agents to reactively plan their own actions to assist in the activities. We applied these algorithms successfully on our robot for performing various assistive tasks ranging from finding objects in large cluttered rooms to working alongside humans in collaborative tasks.

BIOGRAPHICAL SKETCH

Hema Koppula grew up in Visakhapatnam, a southern city in India known for it's beautiful beaches. As a kid, she was always curious and enjoyed doing science experiments and solving puzzles. During high school, her favorite thought experiments were on space time continuum and about how the human brain works. In 2003, Hema was admitted to the Indian Institute of Technology (IIT) Kharagpur, one of the premier engineering institutes in India, for undergraduate studies. It was there she discovered her love for computer science.

During her time at IIT Kharagpur, she explored her many interests including building robots. The first robot she built was an all-terrain robot for an annual national robotics competition. Even though she didn't win the competition, she inspired many by being the first female student to take part in the competition. She later built more robots and won competitions, and took an active role in building the Robotics and A.I. club at IIT Kharagpur.

After graduating, Hema went on to explore the area of machine learning at Yahoo! Labs. She learnt a lot by working on challenging and interesting large scale data science problems. After her short stint at Yahoo! Labs, Hema went to Cornell University to pursue PhD by combining her love for scientific enquiry, interests in A.I. and robotics and practical knowledge from the industry.

To my parents – Lakshmi Kumari and Prabhakara Reddy.

ACKNOWLEDGEMENTS

I am very grateful to my advisor, Ashutosh Saxena, for his continuous support and mentoring. I have learned many things from him, particularly choosing and solving research problems. I am very grateful to Thorsten Joachims, with whom I collaborated and from whom I learnt many valuable insights into machine learning research. I am very grateful to Bart Selman, who was my minor advisor and defense committee member, for always ensuring things were going well and giving invaluable advice along the way. I am also very thankful to Robert Kleinberg and Jitendra Malik, for being on my defense committee and providing insightful and constructive suggestions on my work.

I am very thankful for the many intellectually inspiring interactions as well as the many friendships that have developed during my time at Cornell. In particular, I would like to thank my collaborators, Yun Jiang, Ashesh Jain, Rudhir Gupta and Abhishek Anand, for their hard work and our great times together. It has been my pleasure to get to interact with my fellow students in the Cornell Robot Learning lab, in particular Ian Lenz, Jaeyoung Sung, Ozan Sener, Dipendra Kumar Mishra, and Chenxia Wu. I would also like to thank Anshumali Srivastava, Karthik Raman and Adith Swaminathan, for the many discussions over lunches and for their invaluable feedback on my work. I would also like to thank Becky Stewart for helping with basically everything.

There are a number of close friends at Cornell who are not directly related to my research, but their company made this experience so much more enjoyable. I am particularly grateful to Arunima Singh, Sunayna Jain, Ritu Sahore, Kriti Charan, Shouvik Chatterjee, Sudip Roy and Siddharth Karkare for all the great times and adventurers we had together. Last and foremost, I am thanking my parents and brother, for believing in me and joining me in this endeavor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Understanding people in complex dynamic environments is one of the key challenges in the area of machine perception. It comprises a variety of aspects, ranging from understanding what objects are present in the environments to what activities people are doing and how they interact with objects. Obtaining such an understanding would enable new applications for improving aspects of daily living significantly. Examples include robotic assistants, health-care monitoring systems, self-driving cars, gaming, and mobile devices.

Most of the current scene understanding algorithms are designed for 2D images and videos. These algorithms address the tasks of identifying objects or human poses in an image or a video, for enabling applications such as scene classification, activity recognition, etc. Although these are important visual perception tasks, for more challenging applications such as assistive robotics or self-driving cars, we need to be able to reason about human intentions. This is very critical as in these applications the agents also take actions in human environments. For example, in automated driving we need to not only detect a pedestrian but also predict if they might cross the road to determine safety.

With the advances in 3D sensing technology and their wide availability, we are collecting enormous amounts of visual data about people with the additional modality of 3D depths. The rich information present in the 3D data allows us to significantly improve the basic visual perception tasks. For example, the Microsoft Kinect [178] significantly improved human pose estimation to revolutionize the gaming industry. However, it is still very challenging to extract meaningful information from data which allows an agent to act in human environments. For example, an assistive robot at home needs to be able to understand human environments in a way such that it knows what objects are present and how to interact with these objects as well as with humans.

The focus of this dissertation is to develop learning algorithms for understanding people and their environments from RGB-D data. Our models capture the rich spatio-temporal context of activities as well as human intentions, and provide a functional representation of the environments. We believe that our approach bridges the gap between perception and action, thus enabling many of the applications described above. In the following sections, we introduce and describe the challenges in understanding human environments and activities.

1.1 Modeling Objects and Environments

Inexpensive RGB-D sensors that augment an RGB image with depth data have become widely available and are increasingly becoming the de-facto standard for perception on many robots. Data from these RGB-D sensors provide rich 3D information of the human environments, such as important geometric properties in addition to the local shape and appearance of the environments.

In many domains such as automated driving, service robots, augmented reality, etc., it is essential to learn the spatial structure of the surrounding environment with which people interact. Reasoning about what objects are present in the scene, how they relate to each other, their functional attributes, what to expect in occluded regions, etc., can immensely help these applications. For example, a robot assistant needs to infer what objects are present in the environment and their functional attributes to perform various tasks such as searching

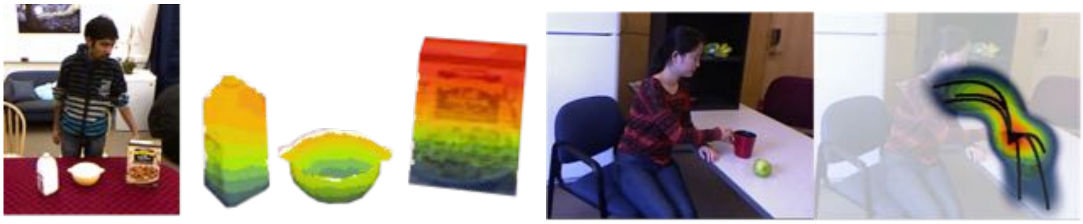


Figure 1.1: **Learned object affordances and trajectories.** The figure shows the learned affordance heatmap (left) for *reachability* (red indicates more *reachable* parts on the object) and the object motion trajectories (right) for *drinking* and *moving* sub-activities.

for objects, arranging objects in a room, etc. We address the following aspects:

- **Modeling spatial relations.** We need to model the spatial structure of the human environments. In addition to predicting object classes, the spatial structure present in the surrounding environments can help us answer other important questions such as where can one find a given object in a scene or how to identify possibly occluded objects. In Chapter 2, we address a part of this problem where we label 3D point clouds with semantic object labels. We proposed a graphical model that naturally captures the geometric relationships of a 3D scene, such as the shape and convexity of objects, the geometric arrangement of objects in an environment, the spatial configuration of parts of an object, object co-occurrences, etc.
- **Modeling object affordances.** It is sometimes more informative to know *how* an object is being used (associated affordances, [43]), rather than knowing *what* the object is (i.e., the object category). For example, if an object is being moved near the mouth of a person, most likely it is a drinking activity as compared to a storing activity where the object would be moved to a self. For such reasoning, we need to go beyond the traditional semantic affordance labels and obtain a physical grounding in terms of

the spatial distribution over the objects and temporal motion trajectories of the objects (see Figure 1.1). We take an generative modeling approach to learn these physically grounded affordances from videos of people using the objects as described in Chapter 4.

1.2 Modeling Humans

Beyond geometry and objects, humans are an important part of the environments. It is important to understand what they are doing, how they interact with the environment and what can they do in the future. Modeling human intent is very challenging as the activities take place over long periods of time and have large variations. For example, a set of simple actions such as reaching, moving and pouring can be combined in different ways to perform a high-level activity such as ‘making cereal’.

Humans constantly interact with their surroundings creating a dynamic environment. Therefore, spatio-temporal reasoning of human actions and intents is a very important task. For many applications it is important to be able to detect what a human in currently doing as well as anticipate what she is going to do next and how. The former ability is useful for applications such as monitoring and surveillance, but we need the latter for applications that require reactive responses, for example, an assistive robot. In order to achieve this goal, there are several challenges:

- **Modeling human activities.** We need to model the spatio-temporal structure of human activities. Human actions vary from simple gestures to complex long duration activities such as taking a medicine or arranging a room, each of which consists of a *long sequence of sub-activities*. Obtaining a descriptive labeling of these activities proves to be a very challenging task given the variability across individuals in performing each sub-activity, and other environment induced conditions such as cluttered background and viewpoint changes. We address this task of labeling complex human activities from 3D videos in Chapter 3. We propose a method to learn human activities by modeling the sub-activities and affordances of the objects, how they change over time, and how they relate to each other.

- **Anticipating future activities.** We need to predict the possible future human activities which allows for planning reactive responses in various applications. For example, if a robot has seen a person move his hand to a coffee mug, it is possible he would move the coffee mug to a few potential places such as his mouth, to a kitchen sink or just move it to a different location on the table. If a robot can anticipate this, then it would rather not start pouring milk into the coffee when the person is moving his hand towards the mug, thus avoiding a spill. In Chapter 4, we address this problem for anticipating future activities and as well as the details of how a human is going to perform them in short-term (e.g., 1-10 seconds) using the learnt physically grounded affordances. We show that anticipation can also improve detection of past activities.
- **Modeling human behavior.** When working with a human to assist, a robot also needs to model how a human would behave in the environment. This includes modeling the habits or preferences of the human while performing various activities as well as how they might adapt to the robot working with them. We propose a collaborative planning al-

gorithm for the robot in Chapter 5, which takes into account the human behavior and learns what actions the robot should perform for effective collaboration.

1.3 First published appearances of the described contributions

Most contributions described in this thesis have first appeared as various publications.

- Chapter 2: Koppula, Anand, Joachims, and Saxena [96, 6].
- Chapter 3: Koppula, Gupta, and Saxena [97].
- Chapter 4: Koppula and Saxena [92, 91, 93, 95].
- Chapter 5: Koppula and Saxena [94].

CHAPTER 2

SCENE UNDERSTANDING

2.1 Introduction

Inexpensive RGB-D sensors that augment an RGB image with depth data have recently become widely available. These cameras are increasingly becoming the de-facto standard for perception for many robots. At the same time, years of research on SLAM (Simultaneous Localization and Mapping) has now made it possible to merge multiple RGB-D images into a single point cloud, easily providing an approximate 3D model of a complete indoor scene (i.e., a room). In this chapter, we explore how this move from part-of-scene 2D images to full-scene 3D point clouds can improve the richness of models for object labeling.

In the past, a significant amount of work has been done in semantic labeling of 2D images [17, 33, 57, 114, 138]. However, a lot of valuable information about the 3D shape and geometric layout of objects is lost when a 2D image is formed from the corresponding 3D world. A classifier that has access to a full 3D model can access important geometric properties in addition to the local shape and appearance of an object. For example, many objects occur in characteristic relative geometric configurations (e.g., a monitor is almost always on a table), and many objects consist of visually distinct parts that occur in a certain relative configuration. More generally, a 3D model makes it possible to reason about a variety of 3D properties such as 3D distances, volume and local convexity.

Some previous works attempt to first infer the 3D structure from 2D images [27, 66, 170, 173] for improving object detection. However, the inferred 3D struc-

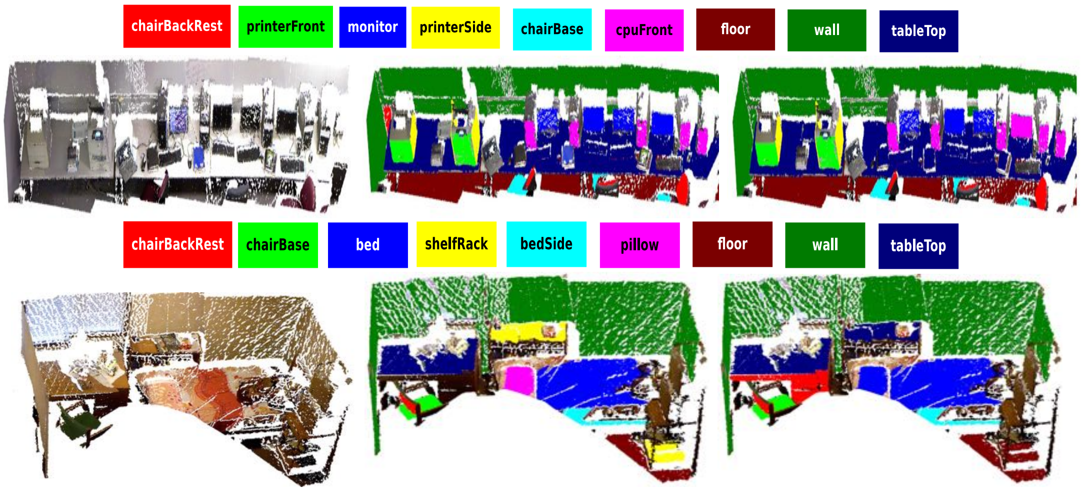


Figure 2.1: Office scene (top) and Home scene (bottom) with the corresponding label coloring above the images. The left-most is the original point cloud, the middle is the ground truth labeling and the right most is the point cloud with predicted labels.

ture is not accurate enough to give significant improvement. Another recent work [207] considers labeling a scene using a single 3D view (i.e., a 2.5D representation). In this work, we first use SLAM (Simultaneous Localization and Mapping) in order to compose multiple views from a Microsoft Kinect RGB-D sensor into one 3D point cloud, providing each RGB pixel with an absolute 3D location in the scene. We then (over-)segment the scene and predict semantic labels for each segment (see Fig. 2.1). We not only predict coarse classes as in [7, 207] (i.e., wall, ground, ceiling, building), but also label individual objects (e.g., printer, keyboard, monitor). Furthermore, we model rich relational information beyond an associative coupling of labels [7].

In this chapter, we propose and evaluate a model and learning algorithm for scene understanding that exploits rich relational information derived from the full-scene 3D point cloud for object labeling and search. In particular, we propose a graphical model that naturally captures the geometric relationships of a

3D scene. Each 3D segment is associated with a node, and pairwise potentials model the relationships between segments (e.g., co-planarity, convexity, visual similarity, object co-occurrences and proximity). The model admits efficient approximate inference [162], and we show that it can be trained using a maximum-margin approach [190, 195, 35] that globally minimizes an upper bound on the training loss. We model both associative and non-associative coupling of labels. With a large number of object classes, the model's parsimony becomes important. Some features are better indicators of label similarity, while other features are better indicators of non-associative relations such as geometric arrangement (e.g., *on-top-of*, *in-front-of*). We therefore model them using appropriate clique potentials rather than using general clique potentials. Our model is thus highly flexible.

Next, we present an algorithm that uses the contextual information present in a 3D scene to predict where an object can be found. For example, its more likely to find a keyboard on top of a table and in front of a monitor, and find a table drawer between the table top and the floor. A robot can use this information in many ways. The robot can move towards the contextually likely location to obtain a better view of the object, resulting in an increase in detection performance. This is especially helpful for small objects, such as a keyboard, that often appear as a segment with only a few points in the original view. It also helps when an object is not visible in the current view or occluded—the robot can move to obtain additional views in contextually likely positions of the object.

We extensively evaluate our model and algorithms over a total of 52 scenes of two types: offices and homes. These scenes were built from about 550 views from the Kinect sensor. We considered the problem of labeling each segment in

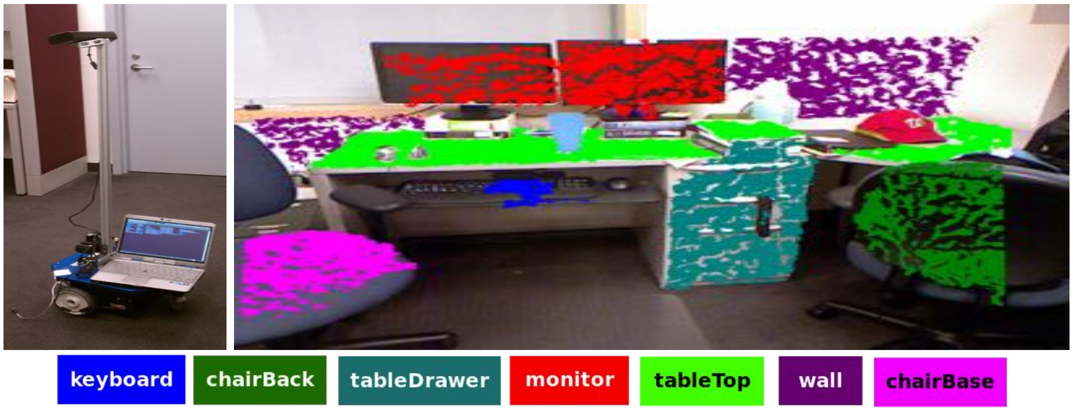


Figure 2.2: **Semantic Scene Labeling**. (Left) Cornell's Blue robot mounted with an RGB-D camera (Microsoft Kinect). (Right) Predicted labeling of a scene.

the scene (from a total of about 50 segments per scene) into 27 classes (17 for offices and 17 for homes, with 7 classes in common). Our experiments show that our method, which captures several local cues and contextual properties, achieves an overall performance of 84.06% on office scenes and 73.38% on home scenes.

We also evaluated our labeling and contextual search algorithms on two mobile robots. In particular, in the task of finding 12 objects in 10 cluttered offices, our robot found the objects with 96% precision and 75% recall. Fig. 2.2 shows Cornell's Blue robot which was used in our experiments and a sample output labeling of an office scene. We have made the videos, data and the full source code (as a ROS and PCL package) available online at: `http://pr.cs.cornell.edu/sceneunderstanding`.

2.2 Related Work

There is a huge body of work in the area of scene understanding and object recognition from 2D images. Previous works have focussed on several different aspects: designing good local features such as HOG (histogram-of-gradients) [21], bag of words [20], and eigenvectors and eignevalues of the scatter matrix [104], active vision for robotics [e.g., 72], and designing good global (context) features such as GIST features [193]. Collet et al. [17]'s MOPED framework performs single-image and multi-image object recognition and pose estimation in complex scenes using an algorithm which iteratively estimates groups of features that are likely to belong to the same object through clustering, and then searches for object hypotheses within each of the groups.

However, the above mentioned approaches do not consider the relative arrangement of the parts of an object or of different objects with respect to each other. It has been shown that this contextual information significantly improves the performance of image-based object detectors. A number of works propose models that explicitly capture the relations between different parts of the object, e.g., part-based models [33], and between different objects in 2D images [57, 113, 138]. However, a lot of valuable information about the shape and geometric layout of objects is lost when a 2D image is formed from the corresponding 3D world. In some recent works, 3D layout or depths have been used for improving object detection [e.g., 8, 55, 56, 66, 111, 112, 114, 170, 171, 173]. Here a rough 3D scene geometry (e.g., main surfaces in the scene) is inferred from a single 2D image or a stereo video stream, respectively. However, the estimated geometry is not accurate enough to give significant improvements. With 3D data, we can more precisely determine the shape, size and geometric orientation of the objects, and several other properties and therefore capture much stronger context.

Visual context can also improve the performance of object detection techniques, by providing cues about an object presence. Perko and Leonardis [151] provided a method to use contextual features and train a classifier which can predict likely locations of objects, also referred to as the ‘focus of attention’, for directing object detection. In methods using active recognition, the performance of object detection for robotics is improved by letting the robots take certain actions, such as moving to an optimal location for obtaining a different view of the object, based on measurement related to entropy [e.g., 23, 105, 106, 124, 129]. The goal here is to obtain high performance in less number of actions. Jia et al. [71] proposes a path planning method which selects a path where the robot obtains new views of an object, and then these views are used for training the classifiers. However, these methods only use 2D images and do not have advantage of using the rich information present in 3D data. We show that our proposed model captures context which not only helps in labeling the scene but also to infer most contextually likely locations of objects using the objects already found in the scene. This enables the robot to move to contextually likely locations and obtain better views for finding objects and improve the performance of scene labeling.

Some earlier works [26, 46, 156, 163] have tried to combine shape and color information from multiple sensors for tasks such as object and obstacle detection. The recent availability of synchronized videos of both color and depth obtained from RGB-D (Kinect-style) depth cameras, shifted the focus to making use of both visual as well as shape features for object detection [e.g., 102, 103], 3D segmentation [e.g., 19], human pose estimation [e.g., 123], and human activity detection [187]. These methods demonstrate that augmenting visual features with 3D information can enhance object detection in cluttered, real-world environments. However, these works do not make use of the contextual relationships between various objects which have been shown to be useful for tasks such as object detection and scene understanding in 2D images. Our goal is to perform semantic labeling of indoor scenes by modeling and learning several contextual relationships.

There is also some recent work in labeling outdoor scenes obtained from LIDAR data into a few geometric classes (e.g., ground, building, trees, vegetation, etc.). Golovinskiy et al. [44] capture context by designing node features. Xiong et al. [208] do so by stacking layers of classifiers; however this models only limited correlation between the labels. An Associative Markov Network is used in [7, 137, 205] to favor similar labels for nodes in the cliques. However, many relative features between objects are not associative in nature. For example, the relationship *on-top-of* does not hold in between two ground segments, i.e., a ground segment cannot be *on-top-of* another ground segment. Therefore, using an associative Markov network is very restrictive for our problem. Contemporary work by [175] did address this issue by using a cutting plane method to train non-associative Markov network. However, as we later show in our experiments, a fully non-associative Markov Model is not ideal when we have a large number of classes and features. More importantly, many of the works discussed above [e.g., 7, 137, 175, 176, 208] were designed for outdoor scenes with LIDAR data (without RGB values). Using RGB information together with depths presents some new challenges, such as designing RGB features for 3D segments. Also, since we consider much larger number of classes compared to previous works (17 vs 3-6 for previous works), the learning task is more challenging due to the large number of parameters. We address this by proposing a parsimonious model.

The most related work to ours is [207], where they label the planar patches in a point cloud of an indoor scene with four geometric labels (walls, floors, ceilings, clutter). They use a CRF to model geometrical relationships such as orthogonal, parallel, adjacent, and coplanar. The learning method for estimating the parameters in [29, 207] was based on maximizing the pseudo-likelihood resulting in a sub-optimal learning algorithm. In comparison, our basic representation is 3D segments (as compared to planar patches) and we consider a much larger number of classes (beyond just the geometric classes). We also capture a much richer set of relationships between pairs of objects, and use a principled max-margin learning method to learn the parameters of our model.

2.3 Approach

We now outline our approach, including the model, its inference methods, and the learning algorithm. Our input is multiple Kinect RGB-D images of a scene (i.e., a room) stitched into a single 3D point cloud using RGBDSLAM.[1] Each such combined point cloud is then over-segmented based on smoothness (i.e., difference in the local surface normals) and continuity of surfaces (i.e., distance between the points). Fig. 2.3 shows the segmentation output for an office scene. These segments are the atomic units in our model. Our goal is to label each one of them.

---

[1] http://openslam.org/rgbdslam.html

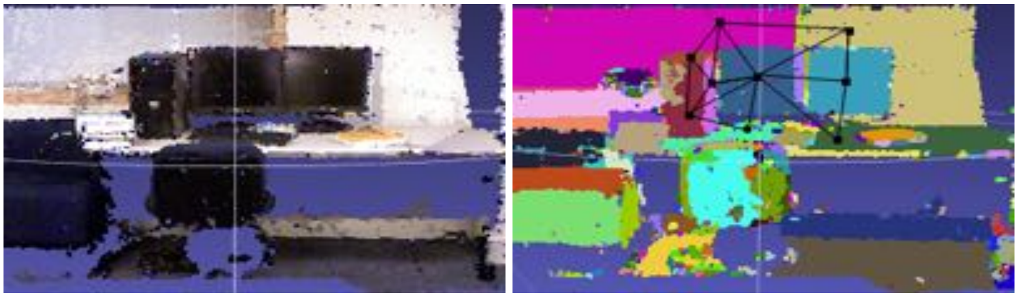


Figure 2.3: **Scene Point Cloud Segmentation.** (Left) Point cloud of an office scene containing 2 monitors and a CPU on a table and a chair beside it. (Right) The segmentation output of the point cloud with each segment represented with a different color. The black dots and lines represent the nodes and edges of the undirected graph defined over the segments (for clarity not all nodes and edges are shown).

Before getting into the technical details of the model, we first outline the properties we aim to capture in our model below:

Visual appearance. The reasonable success of object detection in 2D images shows that visual appearance is a good indicator for labeling scenes. We therefore model the local color, texture, gradients of intensities, etc. for predicting the labels. In addition, we also model the property that if nearby segments are similar in visual appearance, they are more likely to belong to the same object.

Local shape and geometry. Objects have characteristic shapes—for example, a table is horizontal, a monitor is vertical, a keyboard is uneven, and a sofa is usually smoothly curved. Furthermore, parts of an object often form a convex shape. We compute 3D shape features to capture this.

Geometrical context. Many sets of objects occur in characteristic relative geometric configurations. For example, a monitor is always *on-top-of* a table, chairs are usually found *near* tables, a keyboard is *in-front-of* a monitor. This means that our model needs to capture non-associative relationships (e.g., that neighboring segments differ in their labels in specific patterns).

Note the examples given above are just illustrative. For any particular practical application, there will likely be other properties that could also be included. As demonstrated in the following section, our model is flexible enough to include a wide range of features.

2.3.1 Model Formulation

We model the three-dimensional structure of a scene using a model isomorphic to a Markov Random Field with log-linear node and pairwise edge potentials. Given a segmented point cloud $\mathbf{x} = (x_1, ..., x_N)$ consisting of segments x_i, we aim to predict a labeling $\mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_N)$ for the segments. Each segment label $\mathbf{y}_i$ is itself a vector of K binary class labels $\mathbf{y}_i = (y_i^1, ..., y_i^K)$, with each $y_i^k \in \{0, 1\}$ indicating whether a segment i is a member of class k. Note that multiple y_i^k can be 1 for each segment (e.g., a segment can be both a "chair" and a "movable object"). We use such multi-labelings in our attribute experiments where each segment can have multiple attributes, but not in segment labeling experiments where each segment can have only one label.

For a segmented point cloud $\mathbf{x}$, the prediction $\hat{\mathbf{y}}$ is computed as the argmax of a discriminant function $E_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ that is parameterized by a vector of weights $\mathbf{w}$.

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} E_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \tag{2.1}$$

The discriminant function captures the dependencies between segment labels as defined by an undirected graph $(\mathcal{V}, \mathcal{E})$ of vertices $\mathcal{V} = \{1, ..., N\}$ and edges $\mathcal{E} \subseteq$

$\mathcal{V} \times \mathcal{V}$. This undirected graph is derived from the point cloud by adding a vertex for every segment in the point cloud and adding an edge between vertices based on the spatial proximity of the corresponding segments. In detail, we connect two segments (nodes) i and j by an edge if there exists a point in segment i and a point in segment j which are less than *context_range* distance apart. This captures the closest distance between two segments (as compared to centroid distance between the segments)—we study the effect of context range more in Section 2.5. Fig. 2.3 shows the graph structure induced by a few segments of an office scene.

Given $(\mathcal{V}, \mathcal{E})$, we define the following discriminant function based on individual segment features $\phi_n(i)$ and edge features $\phi_t(i, j)$ as further described below.

$$E_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \sum_{i \in \mathcal{V}} \sum_{k=1}^{K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] + \sum_{(i,j) \in \mathcal{E}} \sum_{T_t \in \mathcal{T}} \sum_{(l,k) \in T_t} y_i^l y_j^k \left[\mathbf{w}_{\mathbf{t}}^{lk} \cdot \phi_t(i, j) \right] \tag{2.2}$$

The node feature map $\phi_n(i)$ describes segment i through a vector of features, and there is one weight vector for each of the K classes. Examples of such features are the ones capturing local visual appearance, shape and geometry. The edge feature maps $\phi_t(i, j)$ describe the relationship between segments i and j. Examples of edge features are the ones capturing similarity in visual appearance and geometric context.[2] There may be multiple types t of edge feature maps $\phi_t(i, j)$, and each type has a graph over the K classes with edges T_t. If T_t contains an edge between classes l and k, then this feature map and a weight vector $\mathbf{w}_{\mathbf{t}}^{lk}$ is used to model the dependencies between classes l and k. If the edge is not present in T_t, then $\phi_t(i, j)$ is not used.

---

[2] Even though it is not represented in the notation, note that both the node feature map $\phi_n(i)$ and the edge feature maps $\phi_t(i, j)$ can compute their features based on the full $\mathbf{x}$, not just x_i and x_j.

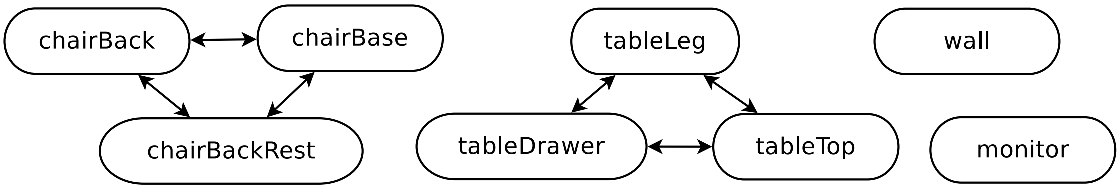


Figure 2.4: Dependency graph for object-associative features. It also contains self-loops (which we do not show here for clarity).

We say that a type t of edge features is modeled by an associative edge potential if the corresponding graph only has self loops, i.e. $T_t = \{(k, k) | \forall k = 1..K\}$. And it is modeled by an non-associative edge potential if the corresponding graph is a fully connected graph, i.e. $T_t = \{(l, k) | \forall l, k = 1..K\}$.

Parsimonious model. Most of the works on 2D images only used associative edge potentials (e.g., [188]), where the idea is that visual similarity is usually an indicator of whether two segments (or pixels) belong to same class. As discussed before, using only associative edge potentials is very restrictive for the task of object labeling and by using non-associative edge potentials we are able to model the dependencies between different objects. Examples of such dependencies are geometric arrangements such as *on-top-of*; we usually find monitor, keyboard etc. *on-top-of* a table, rather than a table *on-top-of* a table. However, modeling every edge feature via a non-associative edge potential will need K^2 parameters per edge feature. Therefore, the number of parameters becomes very large with increase in the number of classes and the number of edge features. Even though, given sufficient data, a non-associative clique potential is general enough to learn associative relationships, this generality comes at an increased cost of training time and memory requirements when the number of classes is large.

We make the observation that not all features need to be modeled using non-associative edge potentials that relate every pair of classes. As described above, certain features such as the visual similarity features indicate when the two segments belong to the same class or parts of the same object, where as other geometric features capture relations between any pair of classes. A key reason for distinguishing between object-associative and non-associative features is parsimony of the model. As not all edge features are non-associative, we avoid learning weight vectors for relationships which do not exist.

Based on this observation we propose our parsimonious model (referred to as *svm_mrf_parsimon*) which partitions edge features into two types—object-associative features (T_{oa}), such as visual similarity, coplanarity and convexity, which usually indicate that two segments belong to the same object, and non-associative features (T_{na}), such as relative geometric arrangement (e.g., *on-top-of*, *in-front-of*), which can capture characteristic configurations under which pairs of different objects occur. We model the object-associative features using object-associative potentials which capture only the dependencies between parts of the same object. The graph for this type of features contains only edges between parts of same object and self loops as shown in Fig. 2.4. Formally, $T_{oa} = \{(l,k)|\exists object,\ l,k \in \text{parts(object)}\}$. The non-associative features are modeled as non-associative edge potentials. Note that $|T_{na}| >> |T_{oa}|$ since, in practice, the number of parts of an objects is much less than K. Due to this, the model we learn with both type of edge features will have much lesser number of parameters compared to a model learnt with all edge features as non-associative features. We show the performance of modeling the edge features using various types of edge potentials in Section 2.5.

2.3.2 Features

The various properties of objects and relations between them are captured in our model with the help of various features that we compute from the segmented point clouds. Our model uses two types of features – the node features and the edge features. Since, the robot knows the height and orientation of its Kinect sensor, we align all the point clouds so that the z-axis is vertical and the ground is at zero height for computing the features. Our features are insensitive to horizontal translation and rotation of the camera, but they place a lot of emphasis on the vertical direction because gravity influences the shape and relative positions of objects to a large extent. Note that unlike images, there is no ambiguity of scale in the input point could and all distance measurements are in meters.

Tables 2.1 and 2.2 summarize the features used in our experiments. $\lambda_{i0}, \lambda_{i1}$ and λ_{i2} are the first three eigenvalues of the scatter matrix computed from the points of segment i. s_i is the set of points in segment i and c_i is the centroid of segment i. r_i is the ray vector to the centroid of segment i from the camera position when it was captured. rh_i is the projection of r_i on horizontal plane. $\hat{n}_i$ is the unit normal of segment i which points towards the camera ($r_i.\hat{n}_i < 0$). The node features $\phi_n(i)$ (Table 2.1) consist of visual appearance features based on histogram of HSV values and the histogram of gradients (HOG) [21], as well as local shape and geometry features that capture properties such as how planar a segment is, its absolute location above ground, and its shape. The local shape features commonly used in the spectral analysis of point clouds (N4-N5) are given by $\lambda_{i2} - \lambda_{i1}$ (linearness), $\lambda_{i2} - \lambda_{i1}$ (planarness) and λ_{i0} (scatterness). N9 captures the tendency of some objects to be near the scene boundaries (walls), e.g., shelf, table, etc.

Table 2.1: Node features for segment *i*.

Description	Count
Visual Appearance	**48**
N1. Histogram of HSV color values	14
N2. Average HSV color values	3
N3. Average of HOG features of the blocks in image spanned by the points of a segment	31
Local Shape and Geometry	**8**
N4. linearness (λ_{i0} - λ_{i1}), planarness (λ_{i1} - λ_{i2})	2
N5. Scatter: λ_{i0}	1
N6. Vertical component of the normal: $\hat{n}_{i_z}$	1
N7. Vertical position of centroid: c_{i_z}	1
N8. Vert. and Hor. extent of bounding box	2
N9. Dist. from the scene boundary (Fig. 2.5)	1

The edge features $\phi_t(i, j)$ (Table 2.2) consist of associative features (E1-E2) based on visual appearance and local shape, as well as non-associative features (E3-E9) that capture the tendencies of two objects to occur in certain configurations. The local shape features include coplanarity and convexity. Coplanarity is defined as:

$$Coplanarity(s_i, s_j) = \begin{cases} -1 & abs(\hat{n}_i \cdot \hat{n}_j) \leq \cos\alpha \\ 1/d_{ij} & \text{otherwise} \end{cases}$$

where $d_{ij} = abs((r_i - r_j) \cdot \hat{n}_i)$ is the distance between centroids in the direction of the normal. Coplanarity only makes sense if the planes are almost parallel. So we use a tolerance angle α (which was set to 30 degrees). The value of coplanarity feature is inversely proportional to the distance between the segments

Table 2.2: Edge features for edge (segment i, segment j).

Description	Count
Visual Appearance (associative)	**3**
E1. Difference of avg HSV color values	3
Local Shape and Geometry (associative)	**2**
E2. Coplanarity and convexity (Fig. 2.5)	2
Geometric context (non-associative)	**6**
E3. Horizontal distance b/w centroids.	1
E4. Vertical Displacement b/w centroids: $(c_{iz} - c_{jz})$	1
E5. Angle between normals (Dot product): $\hat{n}_i \cdot \hat{n}_j$	1
E6. Difference in angle with vertical: $\cos^{-1}(n_{iz})$ - $\cos^{-1}(n_{jz})$	1
E8. Distance between closest points: $\min_{u \in s_i, v \in s_j} d(u,v)$ (Fig. 2.5)	1
E9. Relative position from camera (*in-front-of*/*behind*). (Fig. 2.5)	1

when they are parallel, a large value when the segments are coplanar and -1 when they are not parallel. Convexity determines if the two adjoined segments form a convex surface and is defined as:

$$Convexity(s_i, s_j) = \begin{cases} 1 & (dmin_{ij} < \tau) \wedge [[(\hat{n}_i \cdot \overrightarrow{d_{ij}} \leq 0) \\ & \wedge(\hat{n}_j \cdot \overrightarrow{d_{ji}} \leq 0)] \vee \hat{n}_i \cdot \hat{n}_j \leq \cos\alpha] \\ 0 & \text{otherwise} \end{cases}$$

where $dmin_{ij}$ is the minimum distance between segments s_i and s_j, $\overrightarrow{d_{ij}} = (r_j - r_i)$, is the displacement vector from r_i to r_j and $\overrightarrow{d_{ji}} = -\overrightarrow{d_{ij}}$. τ and α are the tolerance values for minimum distance and the angle respectively. The geometric configuration features include features that capture relationships such as *on-top-of* and *in-front-of*. These are encoded by the vertical distance between centroids (E4), which is positive if segment i is *on-top-of* segment j and negative otherwise, and the relative distance from the camera (E9), which is positive if segment i is *in-front-of* segment j and negative otherwise.

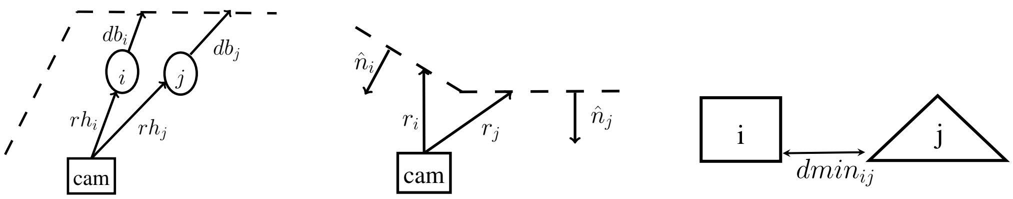


Figure 2.5: Illustration of a few features. (Left) Features N9 and E9. Segment i is *in-front-of* segment j if $\mathbf{rh_i} < \mathbf{rh_j}$. (Middle) Two connected segment i and j form a convex shape if $(\mathbf{r_i} - \mathbf{r_j}).\mathbf{\hat{n}_i} \geq \mathbf{0}$ and $(\mathbf{r_j} - \mathbf{r_i}).\mathbf{\hat{n}_j} \geq \mathbf{0}$. (Right) Illustrating feature E8.

Finally, all the features are binned using a cumulative binning strategy. Each feature instance is represented by n binary values (each corresponding to a bin), where n is the number of bins. n thresholds are computed where i^{th} one is the $\frac{100i}{n}^{th}$ percentile of values of that features in the dataset. The value corresponding to the i^{th} bin is set to 1 if the feature value for that instance lies in the range $[min, th_i)$, where th_i is the i^{th} threshold. This gives us a new set of binary features which are then used for learning the model and during inference. Binning helps in capturing various non-linear functions of features and hence, significantly improves prediction accuracies. In our experiments we use 10 bins for every non-binary feature.

2.3.3 Computing Predictions

Our goal is to label each segment in the segmented point cloud with the most appropriate semantic label. This is achieved by finding the label assignment which maximizes the value of the discriminant function in Eq. (2.2). Given the learned parameters of the model and the features computed from the segmented point cloud, we need to solve the argmax in Eq. (2.1) for the discriminant function in Eq. (2.2). This is NP hard. It can be equivalently formulated as the following mixed-integer program.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \max_{\mathbf{z}} \sum_{i \in \mathcal{V}} \sum_{k=1}^{K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] + \sum_{(i,j) \in \mathcal{E}} \sum_{T_t \in \mathcal{T}} \sum_{(l,k) \in T_t} z_{ij}^{lk} \left[\mathbf{w_t}^{lk} \cdot \phi_t(i,j) \right] \tag{2.3}$$

$$\forall i, j, l, k : \;\; z_{ij}^{lk} \le y_i^l, z_{ij}^{lk} \le y_j^k, \quad y_i^l + y_j^k \le z_{ij}^{lk} + 1 \tag{2.4}$$

$$z_{ij}^{lk}, y_i^l \in \{0, 1\} \tag{2.5}$$

$$\forall i : \sum_{j=1}^{K} y_i^j = 1 \tag{2.6}$$

Note that the products $y_i^l y_j^k$ have been replaced by auxiliary variables z_{ij}^{lk}. We can compute the exact mixed integer solution using a general-purpose MIP solver.[3] We use this method for inference in our offline object labeling experiments (described in Section 2.5.3) and set a time limit of 30 minutes for the MIP solver. This takes 18 minutes on average for a full-scene point cloud and 2 minutes on average for a single-view point cloud. All runtimes are for single CPU implementations using 17 classes.

However, when using our algorithm on labeling new scenes (e.g., during our robotic experiments), objects other than the 27 objects we modeled might be present. Forcing the model to predict one of the 27 objects for all segments would result in wrong predictions for every segment of an unseen class. Therefore, we relax the constraints $\forall i : \sum_{j=1}^{K} y_i^j = 1$ in Eq. (2.6) to $\forall i : \sum_{j=1}^{K} y_i^j \le 1$, which allows a segment to remain unlabeled. This increases precision significantly at the cost of some drop in recall. Also, this relaxed MIP only takes 30 seconds on

---

[3] http://www.gnu.org/software/glpk

an average for a single-view point cloud.

We further relax the problem by removing the constraints in Eq. (2.6) and let the variables z_{ij}^{lk} and y_i^l take values in the interval $[0, 1]$. This results in a linear program that can be shown to always have half-integral solutions (i.e., y_i^l only take values $\{0, 0.5, 1\}$ at the solution) [52]. Furthermore, this relaxation can also be solved as a QPBO (Quadratic Pseudo-Boolean Optimization) problem using a graph-cut method[4] [162], which is orders of magnitude faster than using a general purpose LP solver. The graph-cut method takes less than ***0.05 seconds*** for labeling any full-scene or single-view point cloud. We refer to the solution of this relaxation as $\hat{\mathbf{y}}_{cut}$.

The relaxation solution $\hat{\mathbf{y}}_{cut}$ has an interesting property called *Persistence* [11, 52]. Persistence says that any segment for which the value of y_i^l is integral in $\hat{\mathbf{y}}_{cut}$ (i.e., does not take value 0.5) is labeled just like it would be in the optimal mixed-integer solution. Note that in all relaxations, we say that the node i is predicted as label l iff $y_i^l = 1$.

In Section 2.5, Table 2.4 shows that the solution $\hat{\mathbf{y}}_{cut}$, despite being a relaxation, achieves similar results on all metrics for both full-scene and single-view point clouds, and is orders of magnitude faster to compute.

2.3.4 Learning Algorithm

We take a large-margin approach to learning the parameter vector $\mathbf{w}$ of Eq. (2.2) from labeled training examples $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)$ [191, 190, 195]. Compared to Conditional Random Field training [101] using maximum likelihood, this has

---

[4] http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software/QPBO-v1.3.src.tar.gz

the advantage that the partition function need not be computed, and that the training problem can be formulated as a convex program for which efficient algorithms exist.

Our method optimizes a regularized upper bound on the training error

$$R(h) = \frac{1}{n} \sum_{j=1}^{n} \Delta(\mathbf{y}_j, \hat{\mathbf{y}}_j), \tag{2.7}$$

where $\hat{\mathbf{y}}_j$ is the optimal solution of Eq. (2.1), $\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i=1}^{N} \sum_{k=1}^{K} |y_i^k - \hat{y_i^k}|$, and h is the function mapping the input $\mathbf{x}$ to an output $\mathbf{y}$. In our training problem, the function h parameterized by w is $h_w(\mathbf{x}) = \hat{\mathbf{y}}$. To simplify notation, note that Eq. (2.3) can be equivalently written as $\mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$ by appropriately stacking the w_n^k and $\mathbf{w}_{\mathbf{t}}^{lk}$ into $\mathbf{w}$ and the $y_i^k \phi_n(k)$ and $z_{ij}^{lk} \phi_t(l, k)$ into $\Psi(\mathbf{x}, \mathbf{y})$, where each z_{ij}^{lk} is consistent with Eq. (2.5) given $\mathbf{y}$. Training can then be formulated as the following convex quadratic program [81]:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi && (2.8) \\ s.t. \quad & \forall \bar{\mathbf{y}}_1, ..., \bar{\mathbf{y}}_n \in \{0, 0.5, 1\}^{N \cdot K} : \\ & \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} [\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \bar{\mathbf{y}}_i)] \geq \Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi \end{aligned}$$

While the number of constraints in this quadratic program is exponential in n, N, and K, it can nevertheless be solved efficiently using the cutting-plane algorithm for training structural SVMs [81]. The algorithm maintains a working set of constraints, and it can be shown to provide an ϵ-accurate solution after adding at most $O(R^2 C/\epsilon)$ constraints (ignoring log terms). The algorithm merely needs access to an efficient method for computing

$$\bar{\mathbf{y}}_i = \underset{\mathbf{y} \in \{0, 0.5, 1\}^{N \cdot K}}{\operatorname{argmax}} \left[\mathbf{w}^T \Psi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) \right]. \tag{2.9}$$

Due to the structure of $\Delta(.,.)$, this problem is identical to the relaxed prediction problem in Eqs. (2.3)-(2.5) and can be solved efficiently using graph cuts.

Since our training problem is an overgenerating formulation as defined in [35], the value of ξ at the solution is an upper bound on the training error in Eq. (2.7). Furthermore, Finley and Joachims [35] observed empirically that the relaxed prediction $\hat{\mathbf{y}}_{cut}$ after training $\mathbf{w}$ via Eq. (2.8) is typically largely integral, meaning that most labels y_i^k of the relaxed solution are the same as the optimal mixed-integer solution due to persistence. We made the same observation in our experiments as well: specifically, the average percentage of variables per example that are labeled with integral values is 98.54%.

2.4 Contextual Search

In robotic tasks such as of finding objects, a robot might not find the object it was looking for in its current view. The object might be far away, or occluded, or even out of view. In this section, we propose a method which determines the optimal location for the robot to move for finding the object it was looking for. Our method uses the context from the objects already identified in the robot's current view.

Formally, the goal is to find the 3D location where the desired object is most likely to be found, given a (partially) labeled point cloud. This can be done by sampling 3D locations in the current view and using the learned discriminant function to compare the chances of finding the desired object at those locations. We generate these samples by discretizing the 3D bounding box of the labeled point cloud and considering the centre of each grid as a sample. We considered 1000 equally spaced samples in our experiments.

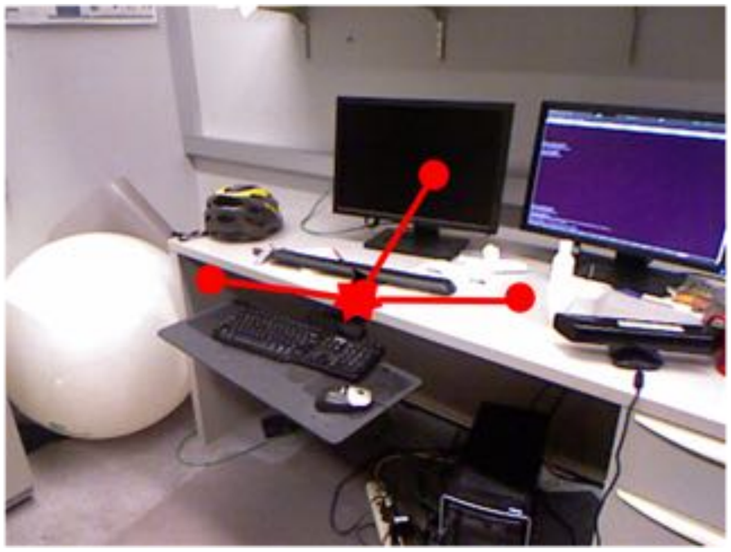


Figure 2.6: Figure illustrating our method for contextual search. The location of the hallucinated segment is shown by a red star and its neighbors are denoted using red edges. In this particular scene the hallucinated keyboard segment is connected to monitor, table-top and table-leg segments. The keyboard and tray segments were left unlabeled by the algorithm, so they are not connected.

2.4.1 Augmented Graph

To evaluate the score of a sample (l_x, l_y, l_z) we first generate a graph on which the discriminant function can be applied. For this, we first take the graph of labeled segments (labeled using our algorithm) and augment it with a node h corresponding to a hallucinated segment x_h at location (l_x, l_y, l_z). We then add edges between this node, h, and the nearby nodes (labeled segments) that are within *context_range* distance from the sampled location. We denote these neighboring segments of the hallucinated segment by $Nbr(x_h)$. Suppose we are looking for an (missing) object class k, we label the newly added node as class k, i.e., the k^{th} element of y_h is 1 and rest are 0. Fig. 2.6 illustrates one such augmentation step. We can now apply our discriminant function to this augmented graph.

Note that each of our sampled locations will give us a unique corresponding augmented graph.

2.4.2 Discriminant Function and Inference

We use the discriminant function from Eq. (2.2) to compute the score of each augmented graph. The optimal sample (location) for finding an instance of class k, $OL(k)$, is the one with highest value of the discriminant function applied to its corresponding augmented graph. Note that only the location-dependent terms in the discriminant function for the newly added node and edges can effect its value, since the rest of the terms are same for every sample. Thus, we can compute $OL(k)$ very efficiently.

We denote the sum of all terms of the discriminant function which do not depend on the location (l_x, l_y, l_z) by a constant ρ. We denote to the label of a node j in the original labeled graph (before augmentation) as $label(j)$. Let $\phi'_n(h)$ denote the features of the hallucinated node h, which only depend on its location (l_x, l_y, l_z). Similarly, let $\phi'_t(h, j)$ denote the features of the edge (h, j) which only depend on location of the hallucinated node. Let w'_n and w'_t denote the projection of node and edge weight vectors respectively that contain only the components corresponding to the location-dependent features. Now we can formally define the optimal location as:

$$OL(k) = \operatorname*{argmax}_{x_h \in samples} E_{\mathbf{w}}(\mathbf{x} \cup x_h, \mathbf{y} \cup y_h) \tag{2.10}$$

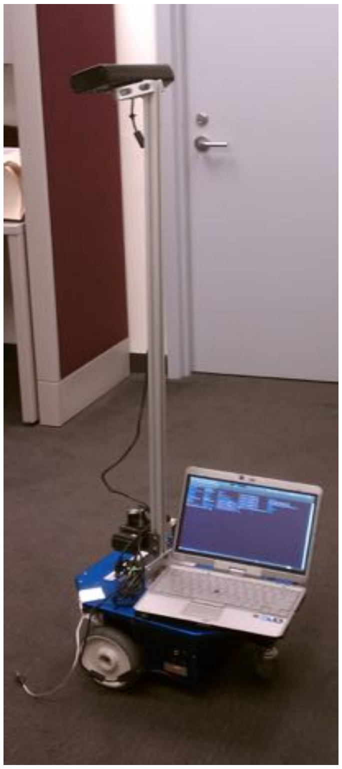


Figure 2.7: Cornell's Blue Robot. It consists of a mobile base mounted with a Kinect sensor.

where,

$$E_{\mathbf{w}}(\mathbf{x} \cup x_h, \mathbf{y} \cup y_h) = \\ w_n'^{k} \cdot \phi_n'(h) + \sum_{(j) \in Nbr(x_h)} \sum_{T_t \in \mathcal{T}} \left[w_t'^{k\ label(j)} \cdot \phi_t'(h, j) \ + w_t'^{label(j)\ k} \cdot \phi_t'(j, h) \right] + \rho \quad (2.11)$$

Once the optimal location $OL(k)$ is found by solving Eq. (2.10), the robot moves close to this location to find objects as described in our contextually-guided object detection experiments presented in Section 2.5.5.

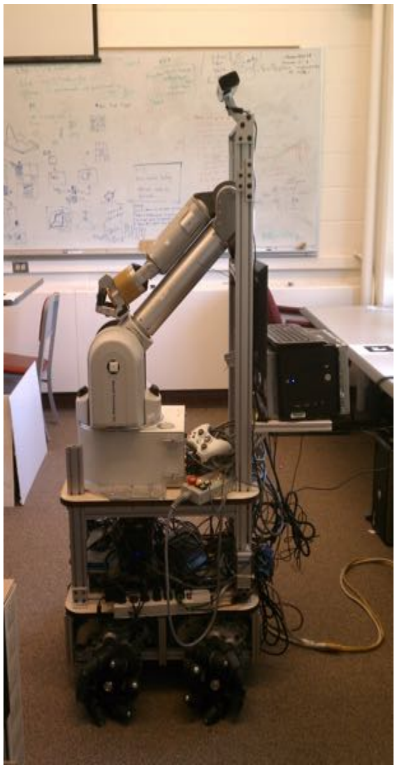


Figure 2.8: Cornell's POLAR Robot. It consists of a omni-directional mobile base with a robotic arm and Kinect sensor mounted on the top.

2.5 Experiments

In this section, we first describe our dataset and present results of the scene labeling task on this dataset. We present a detailed analysis of various factors such as the effect of the 3D features, the effect of adding context, the effect of the presence of unlabeled segments, etc.

We then present the results on robotic experiments on attribute learning and contextually-guided object detection. The object detection experiments were performed using two robots: Cornell's Blue and POLAR robot, shown in Fig. 2.7 and Fig. 2.8 respectively. These robots consist of a mobile base (Erratic and

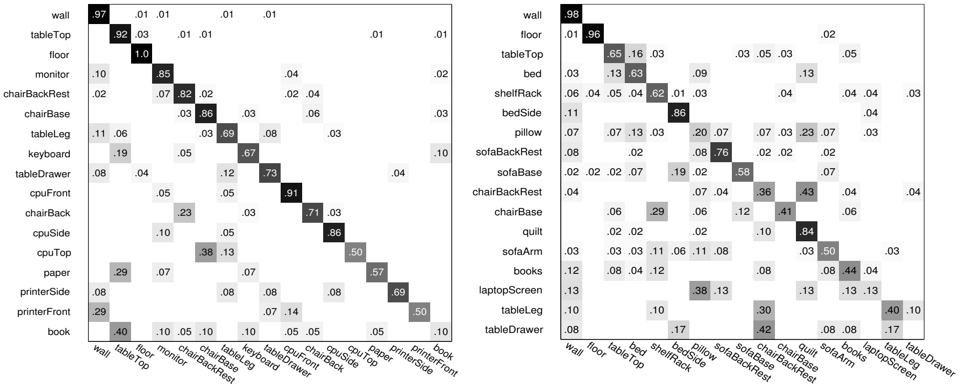
Segway Omni 50 respectively) and were mounted with Kinect sensors as shown in respective figures. (For more details on the hardware specification of the POLAR robot, please see [80].) We used ROS libraries[5] to program these robots.

2.5.1 Data

We consider labeling object segments in full 3D scene (as compared to 2.5D data from a single view). For this purpose, we collected data of 24 office and 28 home scenes from a total of about 550 views. Each scene was reconstructed from about 8-9 RGB-D views from a Microsoft Kinect sensor and contains about one million colored points. We first over-segment the 3D scene (as described earlier) to obtain the atomic units of our representation. For training, we manually labeled the segments, and we selected the labels which were present in a minimum of 5 scenes in the dataset. Specifically, the office labels are: {*wall, floor, tableTop, tableDrawer, tableLeg, chairBackRest, chairBase, chairBack, monitor, printerFront, printerSide keyboard, cpuTop, cpuFront, cpuSide, book, paper*}, and the home labels are: {*wall, floor, tableTop, tableDrawer, tableLeg, chairBackRest, chairBase, sofaBase, sofaArm, sofaBackRest, bed, bedSide, quilt, pillow, shelfRack, laptop, book*}. This gave us a total of 1108 labeled segments in the office scenes and 1387 segments in the home scenes. Often one object may be divided into multiple segments because of over-segmentation. We have made this data available at: `http://pr.cs.cornell.edu/sceneunderstanding/data/data.php`.

---

[5] `http://www.ros.org`



	wall	tableTop	floor	monitor	chairBackRest	chairBase	tableLeg	keyboard	tableDrawer	cpuFront	chairBack	cpuSide	cpuTop	paper	printerSide	printerFront	book
wall	.97		.01	.01			.01		.01								
tableTop		.92	.03		.01	.01								.01			.01
floor			1.0														
monitor	.10			.85						.04							.02
chairBackRest	.02			.07	.82	.02				.02	.04						
chairBase					.03	.86		.03			.06						.03
tableLeg	.11	.06				.03	.69		.08			.03					
keyboard		.19			.05			.67									.10
tableDrawer	.08		.04				.12		.73						.04		
cpuFront				.05			.05			.91							
chairBack					.23			.03			.71	.03					
cpuSide				.10			.05					.86					
cpuTop						.38	.13						.50				
paper		.29		.07				.07						.57			
printerSide	.08						.08		.08			.08			.69		
printerFront	.29								.07	.14						.50	
book		.40		.10	.05	.10		.10		.05	.05			.05			.10

	wall	floor	tableTop	bed	shelfRack	bedSide	pillow	sofaBackRest	sofaBase	chairBackRest	chairBase	quilt	sofaArm	books	laptopScreen	tableLeg	tableDrawer
wall	.98																
floor	.01	.96											.02				
tableTop			.65	.16	.03				.03	.05	.03			.05			
bed	.03		.13	.63			.09					.13					
shelfRack	.06	.04	.05	.04	.62	.01	.03				.04			.04	.04		.03
bedSide	.11					.86									.04		
pillow	.07		.07	.13	.03		.20	.07		.07	.03	.23	.07		.03		
sofaBackRest	.08			.02			.08	.76		.02	.02		.02				
sofaBase	.02	.02	.02	.07		.19	.02		.58				.07				
chairBackRest	.04						.07	.04		.36		.43		.04			.04
chairBase			.06		.29		.06		.12		.41			.06			
quilt			.02	.02			.02			.10		.84					
sofaArm	.03		.03	.03	.11	.06	.11	.08				.03	.50			.03	
books	.12		.08	.04	.12					.08			.08	.44	.04		
laptopScreen	.13						.38	.13				.13		.13	.13		
tableLeg	.10				.10					.30						.40	.10
tableDrawer	.08					.17				.42			.08	.08		.17	

Figure 2.9: Confusion Matrix on office dataset (left) and home dataset (right) with *svm_mrf_parsimon* trained on Shape and Image features.

2.5.2 Segmentation

For segmenting the point cloud, we use a region growing algorithm similar to the Euclidean clustering in the Point-Cloud Library (PCL).[6] It randomly picks a seed point and grows it into a segment. New points are added to an existing segment if their distance to the closest point in the segment is less than a threshold and the local normals calculated at these points are at an angle less than a threshold. We also made the distance threshold proportional to the distance from camera because points far from the camera have more noisy depth estimates. In detail, we used a distance threshold of $0.1d$, where d is the distance of the candidate point from camera, and an angle threshold of $30°$. We observed that these thresholds slightly over-segmented almost all scenes into object-parts of desired granularity. Such a simple approach would not perfectly segment all objects, but we find that our learning method is quite robust to such noisy segmentation.

---

[6] `http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php`

Table 2.3: **Learning experiment statistics.** The table shows average micro precision/recall, and average macro precision and recall for home and office scenes.

		Office Scenes			Home Scenes		
		micro	macro		micro	macro	
features	algorithm	P/R	Prec.	Recall	P/R	Prec.	Recall
None	*max_class*	26.33	26.33	5.88	29.38	29.38	5.88
Image Only	*svm_node_only*	46.67	35.73	31.67	38.00	15.03	14.50
Shape Only	*svm_node_only*	75.36	64.56	60.88	56.25	35.90	36.52
Image+Shape	*svm_node_only*	77.97	69.44	66.23	56.50	37.18	34.73
Image+Shape+Context	*single_frames*	84.32	77.84	68.12	69.13	47.84	43.62
Image+Shape+Context	*svm_mrf_assoc*	75.94	63.89	61.79	62.50	44.65	38.34
Image+Shape+Context	*svm_mrf_nonassoc*	81.45	76.79	70.07	72.38	**57.82**	53.62
Image+Shape+Context	*svm_mrf_parsimon*	**84.06**	**80.52**	**72.64**	**73.38**	56.81	**54.80**

2.5.3 Object Labeling Results

In this subsection, we report the results of offline labeling experiments. Table 2.3 shows the results, performed using 4-fold cross-validation and averaging performance across the folds for the models trained separately on home and office datasets. We use both the macro and micro averaging to aggregate precision and recall over various classes. Let $pred_i(k)$ denote that the i^{th} segment was predicted to be of class k, and $gt_i(k)$ denote that the ground truth label of the i^{th} segment was class k. We now formally define the following metrics used in literature.

$$\begin{aligned}
precision(k) &= \frac{|\{i : (pred_i(k) \wedge gt_i(k))\}|}{|\{i : pred_i(k)\}|} \\
recall(k) &= \frac{|\{i : (pred_i(k) \wedge gt_i(k))\}|}{|\{i : gt_i(k)\}|} \\
macro\_precision &= \frac{\sum_{k=1}^{K} precision(k)}{K} \\
macro\_recall &= \frac{\sum_{k=1}^{K} recall(k)}{K} \\
micro\_precision &= \frac{\sum_{k=1}^{K} |\{i : (pred_i(k) \wedge gt_i(k))\}|}{\sum_{k=1}^{K} |\{i : pred_i(k)\}|} \\
micro\_recall &= \frac{\sum_{k=1}^{K} |\{i : (pred_i(k) \wedge gt_i(k))\}|}{\sum_{k=1}^{K} |\{i : gt_i(k)\}|}
\end{aligned}$$

where, for any set S, $|S|$ denotes its size.

In these experiments, prediction is done using an MIP solver with the constraint that a segment can have exactly one label ($\forall i : \sum_{j=1}^{K} y_i^j = 1$). So, micro precision and recall are same as the percentage of correctly classified segments. The optimal C value is determined separately for each of the algorithms by cross-validation.

Fig. 2.1 shows the original point cloud, ground-truth and predicted labels for one office (top) and one home scene (bottom). Fig. 2.9 show the confusion matrices for office and home scenes on the left and right respectively. On majority of the classes our model predicts the correct label as can be seen from the strong diagonal in the confusion matrices. Some of the mistakes are reasonable, such as a pillow getting confused with the quilt in homes. They often have similar location and texture. In offices, books placed on table-tops sometimes get confused with the table-top.

One of our goals is to study the effect of various factors, and therefore we compared our algorithm with various settings. We discuss them in the following.

Do Image and Point Cloud Features Capture Complimentary Information? The RGB-D data contains both image and depth information, and enables us to compute a wide variety of features. In this experiment, we compare the two kinds of features: Image (RGB) and Shape (Point Cloud) features. To show the effect of the features independent of the effect of context, we only use the node potentials from our model, referred to as *svm_node_only* in Table 2.3. The *svm_node_only* model is equivalent to the multi-class SVM formulation [81]. Table 2.3 shows that Shape features are more effective compared to the Image, and the combination works better on both precision and recall. This indicates that the two types of features offer complementary information and their combination is better for our classification task.

How Important is Context? Using our *svm_mrf_parsimon* model as described in Section 2.3.1, we show significant improvements in the performance over using *svm_node_only* model on both datasets. In office scenes, the micro precision increased by 6.09% over the best *svm_node_only* model that does not use any context. In home scenes the increase is much higher, 16.88%.

The type of contextual relations we capture depend on the type of edge potentials we model. To study this, we compared our method with models using only associative or only non-associative edge potentials referred to as *svm_mrf_assoc* and *svm_mrf_nonassoc*. We observed that modeling all edge features using associative potentials is poor compared to our full model. In fact, using only associative potentials showed a drop in performance compared to *svm_node_only* model on the office dataset. This indicates it is important to capture the relations between regions having different labels. Our *svm_mrf_nonassoc* model does so, by modeling all edge features using non-associative potentials, which can favor or disfavor labels of different classes for nearby segments. It gives higher precision and recall compared to *svm_node_only* and *svm_mrf_assoc*. This shows that modeling using non-associative potentials is a better choice for our labeling problem.

However, not all the edge features are non-associative in nature, modeling them using only non-associative potentials could be an overkill (each non-associative feature adds K^2 more parameters to be learnt). Therefore using our *svm_mrf_parsimon* model to model these relations achieves higher performance in both datasets.

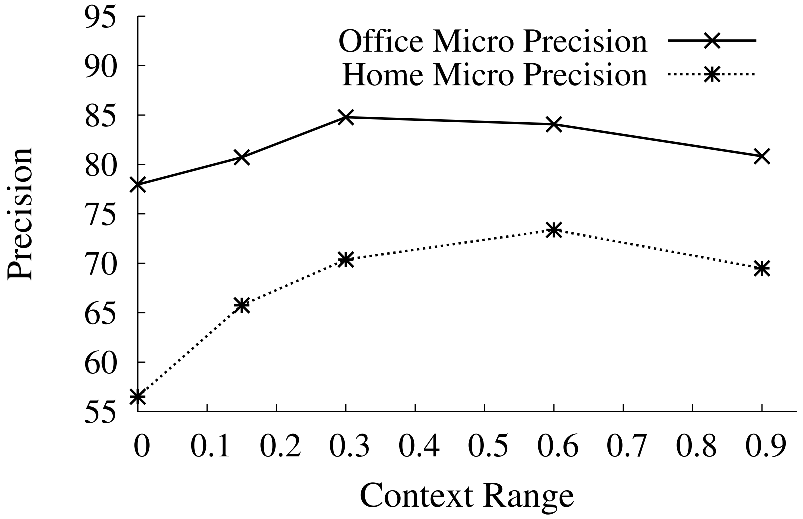


Figure 2.10: Effect of context range on precision (=recall here).

How Large should the Context Range be? Context relationships of different objects can be meaningful for different spatial distances. This range may vary depending on the environment as well. For example, in an office, keyboard and monitor go together, but they may have little relation with a sofa that is slightly farther away. In a house, sofa and table may go together even if they are farther away.

In order to study this, we compared our *svm_mrf_parsimon* with varying *context_range* for determining the neighborhood (see Figure 2.10 for average micro precision vs range plot). Note that the *context_range* is determined from the boundary of one segment to the boundary of the other, and hence it is somewhat independent of the size of the object. We note that increasing the *context_range* increases the performance to some level, and then it drops slightly. We attribute this to the fact that with increasing the *context_range*, irrelevant objects may get an edge in the graph, and with limited training data, spurious relationships may be learned. We observe that the optimal *context_range* for office scenes is around 0.3 meters and 0.6 meters for home scenes.

How does a Full 3D Model Compare to a 2.5D Model? In Table 2.3, we compare the performance of our full model with a model that was trained and tested on single-view point clouds of the same scene. During the comparison, the training folds were consistent with other experiments, however the segmentation of this point cloud was different (because the input point cloud is from a single view). This makes the micro precision values meaningless because the distribution of labels is not same for the two cases. In particular, many large object in a scene (e.g., wall, ground) get split up into multiple segments in single-view point clouds. We observed that the macro precision and recall are higher when multiple views are combined to form the scene. We attribute the improvement in macro precision and recall to the fact that larger scenes have more context, and models are more complete because of multiple views.

What is the effect of the inference method? The results for *svm_mrf* algorithms in Table 2.3 were generated using the MIP solver. The QPBO algorithm however, gives a higher precision and lower recall on both datasets. For example, on office data, the graphcut inference for our *svm_mrf_parsimon* gave a micro precision of 90.25 and micro recall of 61.74. Here, the micro precision and recall are not same as some of the segments might not get any label.

What is the effect of having different granularity when defining the object classes? In our experiments, we have considered class labels at object-part level, e.g., classes *chairBase*, *chairBack* and *chairBackRest* which are parts of a chair. We think that such finer knowledge of objects is important for many robotic applications. For example, if a robot is asked to arrange chairs in a room, knowing the chair parts can help determine the chair's orientation. Also, labeling parts of objects gives our learning algorithm an opportunity to exploit relationships between different parts of an object. In order to analyze the performance gain obtained by considering object-part level labeling, we compare our method with one trained on object level classes. With 10 object level classes in office scenes : {*wall, table, chair, floor, cpu, book, paper, keyboard, printer and monitor*}, we observe a drop in performance, obtaining a micro precision/recall of 83.62%, macro precision of 76.89% and recall of 69.81%.

2.5.4 Attribute Labeling Results

In some robotic tasks, such as robotic grasping, it is not necessary to know the exact object category, but just knowing a few attributes of an object may be useful. For example, if a robot has to clean a floor, it would help if it knows which

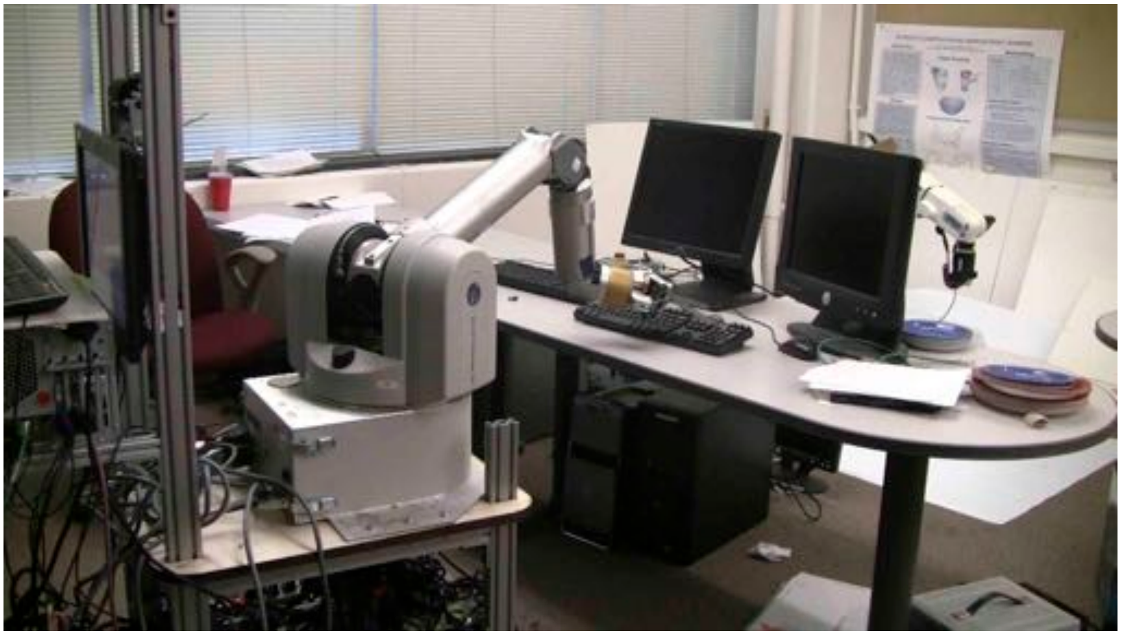


Figure 2.11: Cornell's POLAR (PersOnaL Assistant Robot) using our classifier for detecting a keyboard in a cluttered room.

objects it can move and which it cannot. If it has to place an object, it should place them on horizontal surfaces, preferably where humans do not sit. With this motivation we have designed 8 attributes, each for the home and office scenes, giving a total of 10 unique attributes. They are: {*wall, floor, flat-horizontal-surfaces, furniture, fabric, heavy, seating-areas, small-objects, table-top-objects, electronics*}. Note that each segment in the point cloud can have multiple attributes and therefore we can learn these attributes using our model which naturally allows multiple labels per segment. We compute the precision and recall over the attributes by counting how many attributes were correctly inferred. In home scenes we obtained a precision of 83.12% and 70.03% recall, and in the office scenes we obtain 87.92% precision and 71.93% recall.

2.5.5 Robotic Experiments

The ability to label scenes is very useful for robotics applications, such as of finding/retrieving an object on request. As described in Section 2.3.3, in a detection scenario there can be some segments not belonging to the object classes we consider. Table 2.4 shows the results of running our inference algorithms for detection scenario on the offline office dataset when considering all segments, including those belonging to classes other than the 17 mentioned earlier. The solution of the relaxed MIP (described in Section 2.3.3) gives us high precision (89.87% for micro, and 82.21% for macro), but low recall (55.36% for micro, and 35.25% for macro). The $\hat{\mathbf{y}}_{cut}$ solution, computed using the graph-cut method, also achieves comparable accuracy (see line 3 of Table 2.4) and is very fast to compute (takes less than 0.05 second per scene). Therefore, if the robot finds an object it is likely correct, but the robot may not find all the objects easily. This is where our contextual search algorithm (described in Section 2.4) becomes useful.

Table 2.4: Precision and Recall for detection experiments in office scenes (offline, single-view).

Algorithm	Micro-precision	Micro-recall	Macro-precision	Macro-recall
max_class	22.64	22.64	22.64	5.88
svm_mrf_parsimon	**89.87**	55.36	82.21	35.25
svm_mrf_parsimon w/ QPBO, $\hat{y}_{cut}$	87.41	**56.82**	**82.95**	**38.14**

In order to evaluate our contextual search algorithm, we test our approach on our Blue robot for the task of finding 12 object classes located in 10 office

Table 2.5: Class-wise precision recall for Robotic Experiments using contextual search.

class	# instances	precision	recall
Wall	10	100	100
Table Top	10	100	100
Table Leg	10	71	50
Table Drawer	7	100	71
Chair Backrest	10	100	100
Chair Base	10	100	100
Chair Back	8	100	88
Monitor	9	100	100
Keyboard	9	100	78
CPU	8	50	11
Printer	1	100	100
Paper	10	100	22
Overall Micro	102	96	75
Overall Macro		93	77

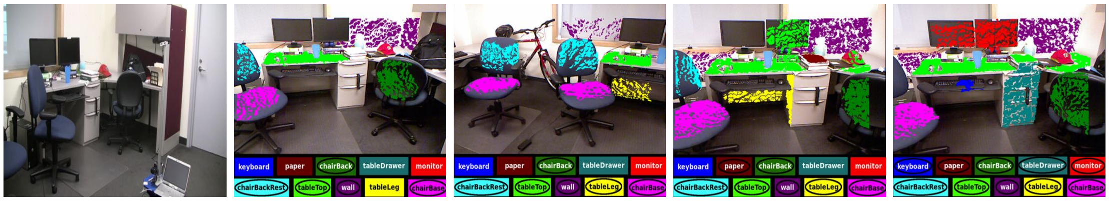


Figure 2.12: (Left): The robot in an office scene. (Columns 2-5): Sequence of colored images corresponding to the labeled point clouds generated by the robot during the object detection experiment.

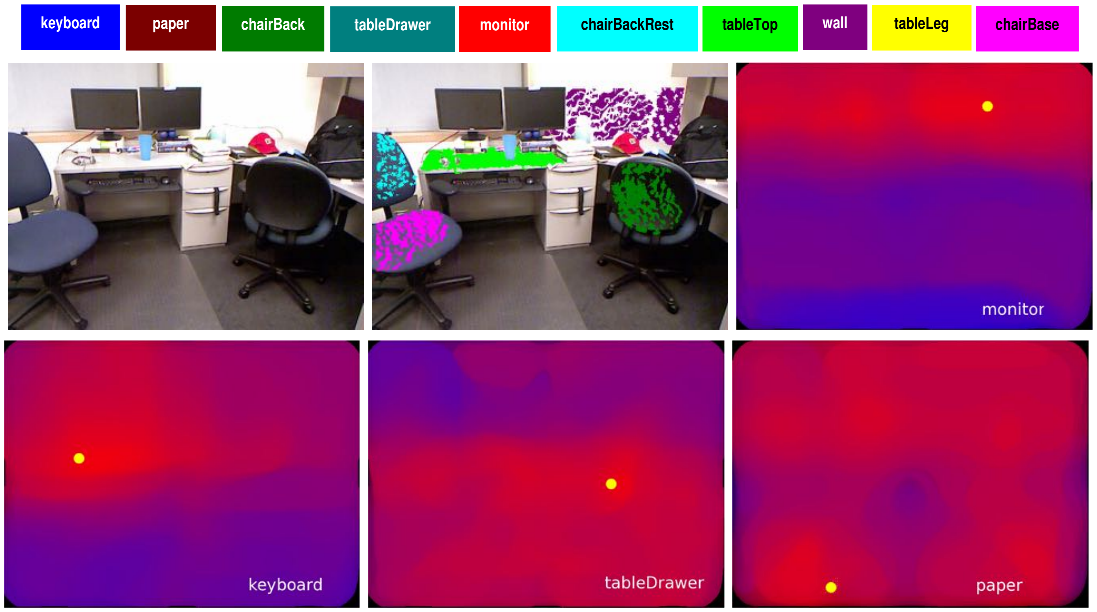


Figure 2.13: (Top left) Original image. (Top mid) Inferred labels color-coded using the legend shown above. (Top right, bottom left, bottom mid and bottom right): Contextually likely positions for finding monitor, keyboard, tableDrawer and paper respectively. In these heatmaps, red indicates that the target object is more likely to be found there. The circular yellow dot indicates the most likely location.

scenes. The robot starts from a pre-determined location in an office and searches for a given list of objects in the room. The goal of the robot is to find at least one instance for each of the object classes it is asked to search for. Since, the RGB-D sensor has a narrow field-of-view (57 degrees horizontally), the robot first scans the room by turning a fixed angle each time. It labels the point cloud it obtains in each view and saves the labeled point clouds.

Next, for all the object classes it did not find, it computes the contextually likely locations for the objects using the algorithm described in Section 2.4. Using the the inferred locations, the robot moves in that direction in order to get better view of the objects. Fig. 2.12 shows the experiment run in one office scene. The first column shows the Blue robot and the scene in which it is asked to find

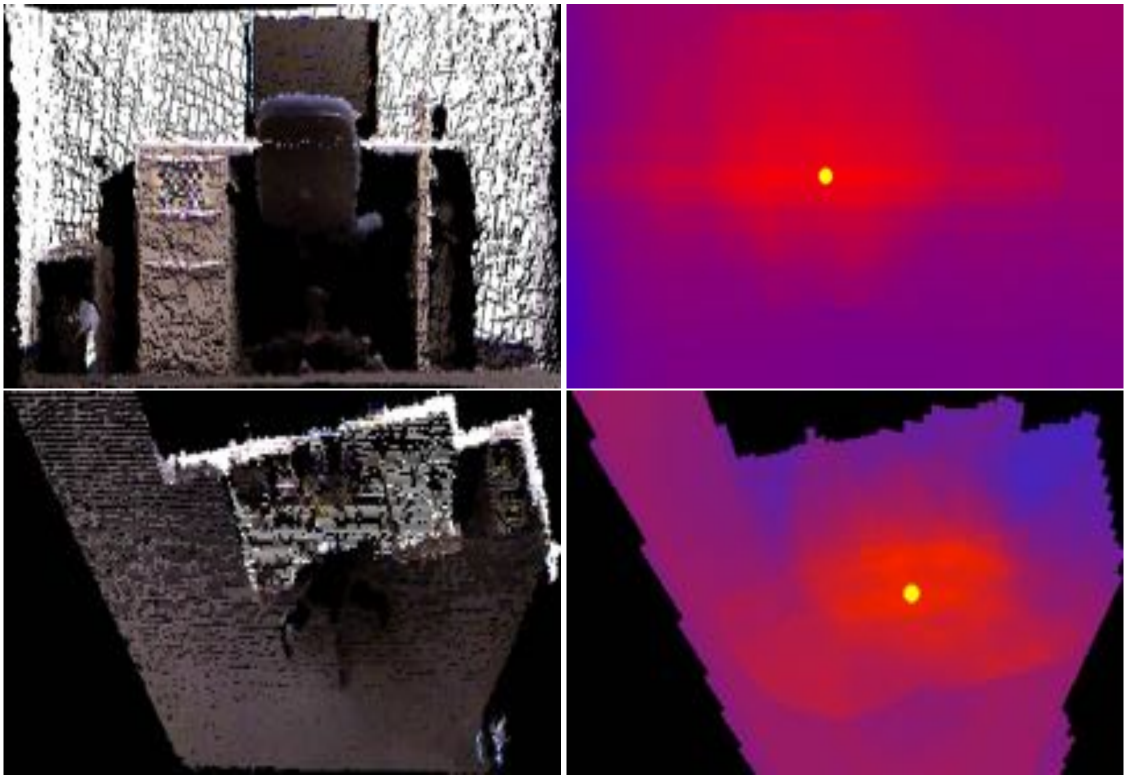


Figure 2.14: (Left column): Front and top views(orthographic projections) of an office scene in which a keyboard is occluded. (Right column): Corresponding heatmaps where red indicates that the keyboard is more likely to be found there. The yellow dot indicates the most likely location.

the objects and the rest of the columns show the sequence of colored images corresponding to the labeled point clouds. The first two point clouds were obtained when the robot was scanning the room from a distance, and the last two are obtained after inferring the contextually likely locations of the objects not found and moving closer to these locations. Table 2.5 shows the precision and recall of finding the 12 object classes in our robotic experiments.

To evaluate our contextual search algorithm, we present both qualitative and quantitative results. Fig. 2.13 qualitatively shows the predictions for finding monitor, keyboard, tableDrawer and paper in a frame in which they were not found. These heatmaps are generated in 3D, but for visualization purposes they are aligned to the original RGB image in Fig. 2.13. As can be seen form Fig. 2.13, a monitor is predicted to be most likely found above the table and a tableDrawer is likely to be found under the table. However, it does not do a great job for paper, and this may be because we had very few examples for this class in our training set.

Fig. 2.14 shows that our algorithm can also be used on a robot to find objects even when they are occluded. Clearly, it predicts that a keyboard is likely to be found in front of the monitor and at about the same height as the tableTop. To quantitatively evaluate the predictions, we collected 10 frames where a keyboard was not detected, but other objects such as table and monitor were detected. We then applied our contextual search algorithm to find the optimal 3D locations for finding a keyboard. For each of these scenes, we computed the minimum distance of the actual keyboard-points to the inferred optimal location. As a baseline, if the predictor always predicted midpoints of the scene as the probable location of the keyboard, the max, mean and median values over 10 scenes were 113.5cm, 32.6cm and 27.2cm respectively. Using our method, we get 36.3cm, 17.5cm and 15.9cm respectively. We found that this helps the robot in finding the objects with only a few moves.

We have the code available as a ROS and PCL package. Code, datasets as well as videos showing our robots finding objects using our algorithm are available at: `http://pr.cs.cornell.edu/sceneunderstanding/`.

2.6 Conclusion

In this chapter, we have proposed the model and learning algorithm for semantic labeling that exploits rich relational information in full-scene 3D point clouds. Our method captures various features and contextual relations, including the local visual appearance and shape cues, object co-occurence relationships and geometric relationships. We showed how visual and shape features can be modeled parsimoniously when the number of classes is large. We also presented an algorithm to infer contextually likely locations for the desired objects given the current labelings in the scene. We tested our method on a large offline dataset, as well as on the task of mobile robots finding objects in cluttered scenes.

CHAPTER 3

ACTIVITY UNDERSTANDING

3.1 Introduction

It is indispensable for a personal robot to perceive the environment in order to perform assistive tasks. Recent works in this area have addressed tasks such as estimating geometry [58], tracking objects [16], recognizing objects [18], placing objects [76] and labeling geometric classes [96, 6]. Beyond geometry and objects, humans are an important part of the indoor environments. Building upon the recent advances in human pose detection from an RGB-D sensor [178], in this chapter we present learning algorithms to detect the human activities and object affordances. This information can then be used by assistive robots as shown in Fig. 3.1.

Most prior works in human activity detection have focussed on activity detection from still images or from 2D videos. Estimating the human pose is the primary focus of these works, and they consider actions taking place over

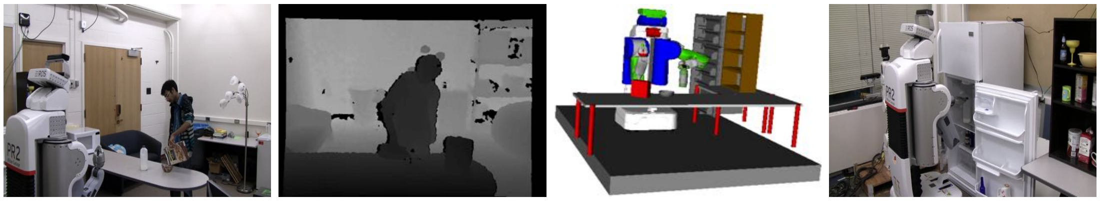


Figure 3.1: An assistive robot observes human activities (making cereal). Using RGB-D images as input, our algorithm detects the activity being performed as well as the object affordances. This enables the robot to figure out how to interact with objects and plan actions, and to respond appropriately to the activities being performed (cleaning up the table).

shorter time scales (see Section 3.2). Having access to a 3D camera, which provides RGB-D videos, enables us to robustly estimate human poses and use this information for learning complex human activities.

Our focus in this work is to obtain a descriptive labeling of the complex human activities that take place over long time scales and consist of a *long sequence of sub-activities*, such as making cereal and arranging objects in a room (see Fig. 3.9). For example, the making cereal activity consists of around 12 sub-activities on average, which includes reaching the pitcher, moving the pitcher to the bowl, and then pouring the milk into the bowl. This proves to be a very challenging task given the variability across individuals in performing each sub-activity, and other environment induced conditions such as cluttered background and viewpoint changes. (See Fig. 3.2 for some examples.)

In most previous works, object detection and activity recognition have been addressed as separate tasks. Only recently, some works have shown that modeling mutual context is beneficial [49, 211]. The key idea in our work is to note that, in activity detection, it is sometimes more informative to know *how* an object is being used [associated affordances, 43] rather than knowing *what* the object is (i.e. the object category). For example, both chair and sofa might be categorized as 'sittable', and a cup might be categorized as both 'drinkable' and 'pourable.' Note that the affordances of an object change over time depending on its use, e.g., a pitcher may first be *reachable*, then *movable* and finally *pourable*. In addition to helping activity recognition, recognizing object affordances is important by itself because of their use in robotic applications [e.g., 98, 75, 73].

We propose a method to learn human activities by modeling the sub-activities and affordances of the objects, how they change over time, and how

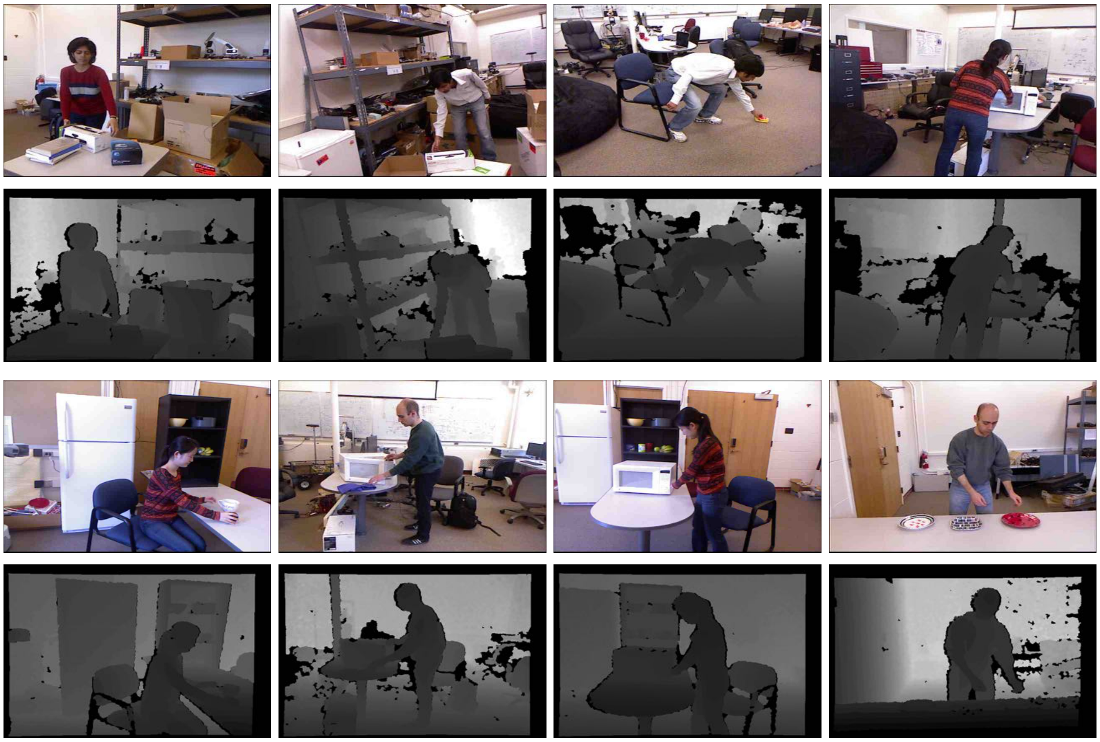


Figure 3.2: **Significant Variations, Clutter and Occlusions**: Example shots of *reaching* sub-activity from our dataset. First and third rows show the RGB images, and the second and bottom rows show the corresponding depth images from the RGB-D camera. Note that there are significant variations in the way the subjects perform the sub-activity. In addition, there is significant background clutter and subjects are partially occluded (e.g., column 1) or not facing the camera (e.g., row 1 column 4) in many instances.

they relate to each other. More formally, we define a Markov random field (MRF) over two kinds of nodes: object and sub-activity nodes. The edges in the graph model the pairwise relations among interacting nodes, namely the object–object interactions, object–sub-activity interactions, and the temporal interactions. This model is built with each spatio-temporal segment being a node. The parameters of this model are learnt using a structural support vector machine (SSVM) formulation [35]. Given a new sequence of frames, we label the high-level activity, all the sub-activities and the object affordances using our learned model.

The activities take place over a long time scale, and different people execute sub-activities differently and for different periods of time. Furthermore, people also often merge two consecutive sub-activities together. Thus, segmentations in time are noisy and, in fact, there may not be one 'correct' segmentation, especially at the boundaries.

One approach could be to consider *all* possible segmentations, and marginalize the segmentation; however, this is computationally infeasible. In this work, we perform sampling of several segmentations, and consider labelings over these temporal segments as latent variables in our learning algorithm.

We first demonstrate significant improvement over previous work on Cornell Activity Dataset (CAD-60). We then contribute a new dataset comprising 120 videos collected from four subjects (CAD-120). These datasets along with our code are available at `http://pr.cs.cornell.edu/humanactivities/`. In extensive experiments, we show that our approach outperforms the baselines in both the tasks of activity as well as affordance detection. We achieved an accuracy of 91.8% for affordance, 86.0% for sub-activity labeling and 84.7% for high-level activities respectively when given the ground truth segmentation, and an accuracy of 79.4%, 63.4% and 75.0% on these respective tasks using our multiple segmentation algorithm.

In summary, our contributions are five-fold:

- We provide a fully annotated RGB-D human activity dataset containing 120 long term activities such as *making cereal*, *microwaving food*, etc. Each video is annotated with the human skeleton tracks, object tracks, object affordance labels, sub-activity labels, and high-level activities.

- We propose a method for joint sub-activity and affordance labeling of RGB-D videos by modeling temporal and spatial interactions between humans and objects.
- We address the problem of temporal segmentation by learning the optimal labeling from multiple temporal segmentation hypotheses.
- We provide extensive analysis of our algorithms on two datasets and also demonstrate how our algorithm can be used by assistive robots.
- We release full source code along with ROS and PCL integration.

The rest of the chapter is organized as follows. We start with a review of the related work in Section 3.2. We describe the overview of our methodology in Section 3.3 and describe the model in Section 3.4. We then describe the object tracking and segmentation methods in Section 3.5 and 3.6 respectively and describe the features used in our model in Section 3.7. We present our learning, inference and temporal segmentation algorithms in Section 3.8. We present the experimental results along with robotic demonstrations in Section 3.9.

3.2 Related Work

There is a lot of recent work in improving robotic perception in order to enable the robots to perform many useful tasks. These works includes 3D modeling of indoor environments [58], semantic labeling of environments by modeling objects and their relations to other objects in the scene [96, 103, 161, 6], developing frameworks for object recognition and pose estimation for manipulation

[18], object tracking for 3D object modeling [99], etc. Robots are now becoming more integrated in human environments and are being used in assistive tasks such as automatically interpreting and executing cooking recipes [10], robotic laundry folding [131] and arranging a disorganized house [76, 73]. Such applications makes it critical for the robots to understand both object affordances as well as human activities in order to work alongside with humans. We describe the recent advances in the various aspects of this problem here.

Object affordances. An important aspect of the human environment a robot needs to understand are the object affordances. Most of the work within the robotics community related to affordances has focused on predicting opportunities for interaction with an object either by using visual clues [184, 59, 5] or through observation of the effects of exploratory behaviors [136, 158, 133]. For instance, [184] proposed a probabilistic graphical model that leverages visual object categorization for learning affordances and [59] proposed the use of physical and visual attributes as a mid-level representation for affordance prediction. [5] proposed a method to find affordances which depends solely on the objects of interest and their position and orientation in the scene. These methods, do not consider the object affordances in human context, i.e. how the objects are usable by humans. We show that human-actor based affordances are essential for robots working in human spaces in order for them to interact with objects in a human-desirable way.

There is some recent work in interpreting human actions and interaction with objects [120, 172, 3, 90, 114] in context of learning to perform actions from demonstrations. [120] use context from objects in terms of possible grasp affordances to focus the attention of their action recognition system and reduce ambiguities. In contrast to these methods, we propose a model to learn human activities spanning over long durations and action-dependent affordances which make robots more capable in performing assistive tasks as we later describe in Section 3.9.5. [172] used supervised learning to detect grasp affordances for grasping novel objects. [114] used a cascaded classification model to model the interaction between objects, geometry and depths. However, their work is limited to 2D images. In recent work, [75] used a data-driven technique for learning spatial affordance maps for objects. This work is different from ours in that we consider semantic affordances with spatio-temporal grounding useful for activity detection. Pandey and Alami [2010, 2012] proposed mightability maps and taskability graphs that capture affordances such as reachability and visibility, while considering efforts required to be performed by the agents. While this work manually defines object affordances in terms of kinematic and dynamic constraints, our approach learns them from observed data.

Human activity detection from 2D videos. There has been a lot of work on human activity detection from images [210, 212] and from videos [108, 118, 64, 177, 128, 152, 159, 167, 189]. Here, we discuss works that are closely related to ours, and refer the reader to [2] for a survey of the field. Most works [e.g. 64, 177, 128] consider detecting actions at a 'sub-activity' level (e.g. *walk*, *bend*, and *draw*) instead of considering high-level activities. Their methods range from discriminative learning techniques for joint segmentation and recognition [177, 64] to combining multiple models [128]. Some works, such as [189], consider high-level activities. [189] proposed a latent model for high-level activity classification and have the advantage of requiring only high-level activity labels for learning. None of these methods explicitly consider the role of objects or object affordances that not only help in identifying sub-activities and high-level activities, but are also important for several robotic applications [e.g. 98].

Some recent works [49, 211, 4, 74, 152] show that modeling the interaction between human poses and objects in 2D videos results in a better performance on the tasks of object detection and activity recognition. However, these works cannot capture the rich 3D relations between the activities and objects, and are also fundamentally limited by the quality of the human pose inferred from the 2D data. More importantly, for activity recognition, the object *affordance* matters more than its category.

[87] used a factorial CRF to simultaneously segment and classify human hand actions, as well as classify the object affordances involved in the activity from 2D videos. However, this work is limited to classifying only hand actions and does not model interactions between the objects. We consider complex full-body activities and show that modeling object–object interactions is important as objects have affordances even if they are not directly interacted with human hands.

Human activity detection from RGB-D videos. Recently, with the availability of inexpensive RGB-D sensors, some works [115, 141, 186, 214, 185] consider detecting human activities from RGB-D videos. [115] proposed an expandable graphical model, to model the temporal dynamics of actions and use a bag of 3D points to model postures. They use their method to classify 20 different actions which are used in context of interacting with a game console, such as draw tick, draw circle, hand clap, etc. [214] designed 4D local spatio-temporal features and use an LDA classifier to identify six human actions such as lifting, removing, waving, etc., from a sequence of RGB-D images. However, both these works only address detecting actions which span short time periods. [141]

also designed feature representations such as spatio-temporal interest points and motion history images which incorporate depth information in order to achieve better recognition performance. [147] used data from laser rangefinder to model observed movement patterns and interactions between persons. They segment tracks into activities based on difference in displacement distributions in each segment, and use a Markov model for capturing the transition probabilities. None of these works model interactions with objects which provide useful information for recognizing complex activities.

In recent previous work from our lab, [186, 185] proposed a hierarchical maximum entropy Markov model to detect activities from RGB-D videos and treat the sub-activities as hidden nodes in their model. However, they use only human pose information for detecting activities and also constrain the number of sub-activities in each activity. In contrast, we model context from object interactions along with human pose, and also present a better learning algorithm. (See Section 3.9 for further comparisons.) [41] also use depth data to perform sub-activity (referred to as action) classification and functional categorization of objects. Their method first detects the sub-activity being performed using the estimated human pose from depth data, and then performs object localization and clustering of the objects into functional categories based on the detected sub-activity. In contrast, our proposed method performs joint sub-activity and affordance labeling and uses these labels to perform high-level activity detection.

All of the above works lack a unified framework that combines all of the information available in human interaction activities and therefore we propose a model that captures both the spatial and temporal relations between object affordances and human poses to perform joint object affordance and activity detection.

3.3 Overview

Over the course of a video, a human may interact with several objects and perform several sub-activities over time. In this section we describe at a high level how we process the RGB-D videos and model the various properties for affordance and activity labeling.

Given the raw data containing the color and depth values for every pixel in the video, we first track the human skeleton using Openni's skeleton tracker[1] for obtaining the locations of the various joints of the human skeleton. However these values are not very accurate, as the Openni's skeleton tracker is only designed to track human skeletons in clutter-free environments and without any occlusion of the body parts. In real-world human activity videos, some body parts are often occluded and the interaction with the objects hinders accurate skeleton tracking. We show that even with such noisy data, our method gets high accuracies by modeling the mutual context between the affordances and sub-activities.

We then segment the object being used in the activity and track them through out the 3D video, as explained in detail in Section 3.5. We model the activities and affordances by defining a MRF over the spatio-temporal sequence we get from an RGB-D video, as illustrated in Fig. 3.3. MRFs are a workhorse of machine learning, and have been successfully applied to many applications [e.g.

---

[1] http://openni.org

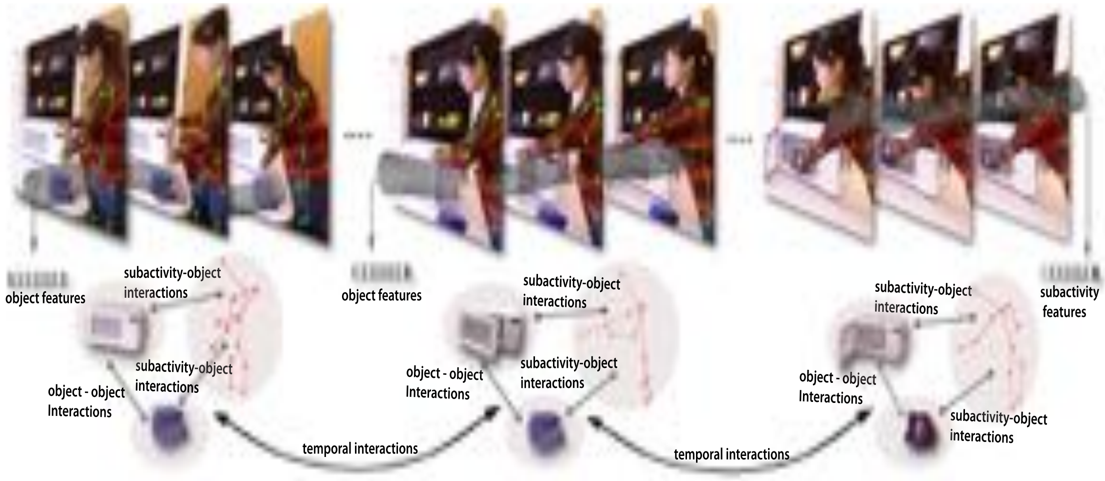

object features
subactivity-object interactions
subactivity-object interactions
object - object Interactions
temporal interactions
object features
subactivity-object interactions
object - object Interactions
subactivity-object interactions
temporal interactions
subactivity-object interactions
subactivity features
object - object Interactions
subactivity-object interactions

Figure 3.3: Pictorial representation of the different types of nodes and relationships modeled in part of the *cleaning objects* activity comprising three sub-activities: *reaching*, *opening* and *scrubbing*. (See Section 3.3.)

173]. Please see [89] for a review. If we build our graph with nodes for objects and sub-activities for each time instant (at 30 fps), then we will end up with quite a large graph. Furthermore, such a graph would not be able to model meaningful transitions between the sub-activities because they take place over a long-time (e.g. a few seconds). Therefore, in our approach we first segment the video into small temporal segments, and our goal is to label each segment with appropriate labels. We try to over-segment, so that we end up with more segments and avoid merging two sub-activities into one segment. Each of these segments occupies a small length of time and therefore, considering nodes per segment gives us a meaningful and concise representation for the graph $\mathcal{G}$. With such a representation, we can model meaningful transitions of a sub-activity following another, e.g. *pouring* followed by *moving*. Our temporal segmentation algorithms are described in Section 3.6. The outputs from the skeleton and object tracking along with the segmentation information and RGB-D videos are then used to generate the features described in Section 3.7.

Given the tracks and segmentation, the graph structure ($\mathcal{G}$) is constructed with a node for each object and a node for the sub-activity of a temporal segment, as shown in Fig. 3.3. The nodes are connected to each other within a temporal segment and each node is connected to its temporal neighbors by edges as further described in Section 3.4. The learning and inference algorithms for our model are presented in Section 3.8. We capture the following properties in our model:

- **Affordance–sub-activity relations.** At any given time, the affordance of the object depends on the sub-activity it is involved in. For example, a cup has the affordance of *'pour-to'* in a *pouring* sub-action and has the affordance of *'drinkable'* in a *drinking* sub-action. We compute relative geometric features between the object and the human's skeletal joints to capture this. These features are incorporated in the energy function as described in Eq. (3.6).
- **Affordance–affordance relations.** Objects have affordances even if they are not interacted directly with by the human, and their affordances depend on the affordances of other objects around them. For example, in the case of *pouring* from a pitcher to a cup, the cup is not interacted with by the human directly but has the affordance *'pour-to'*. We therefore use relative geometric features such as 'on top of', 'nearby', 'in front of', etc., to model the affordance–affordance relations. These features are incorporated in the energy function as described in Eq. (3.5).
- **Sub-activity change over time.** Each activity consists of a sequence of sub-activities that change over the course of performing the activity. We

model this by incorporating temporal edges in $\mathcal{G}$. Features capturing the change in human pose across the temporal segments are used to model the sub-activity change over time and the corresponding energy term is given in Eq. (3.8).

- **Affordance change over time.** The object affordances depend on the sub-activity being performed and hence change along with the sub-activity over time. We model the temporal change in affordances of each object using features such as change in appearance and location of the object over time. These features are incorporated in the energy function as described in Eq. (3.7).

3.4 Model

Our goal is to perform joint activity and object affordance labeling of RGB-D videos. We model the spatio-temporal structure of an activity using a model isomorphic to a MRF with log-linear node and pairwise edge potentials (see Fig. 3.3 and 3.4 for an illustration). The MRF is represented as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Given a temporally segmented 3D video, with temporal segments $s \in \{1, ..., N\}$, our goal is to predict a labeling $\mathbf{y} = (\mathbf{y}_1, ..., \mathbf{y}_N)$ for the video, where $\mathbf{y}_s$ is the set of sub-activity and object affordance labels for the temporal segment s. Our input is a set of features $\mathbf{x}$ extracted from the segmented 3D video as described in Section 3.7. The prediction $\hat{\mathbf{y}}$ is computed as the argmax of a energy function $E_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ that is parameterized by weights $\mathbf{w}$.

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, E_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) \tag{3.1}$$

$$E_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = E_o + E_a + E_{oo} + E_{oa} + E^t_{oo} + E^t_{aa} \tag{3.2}$$

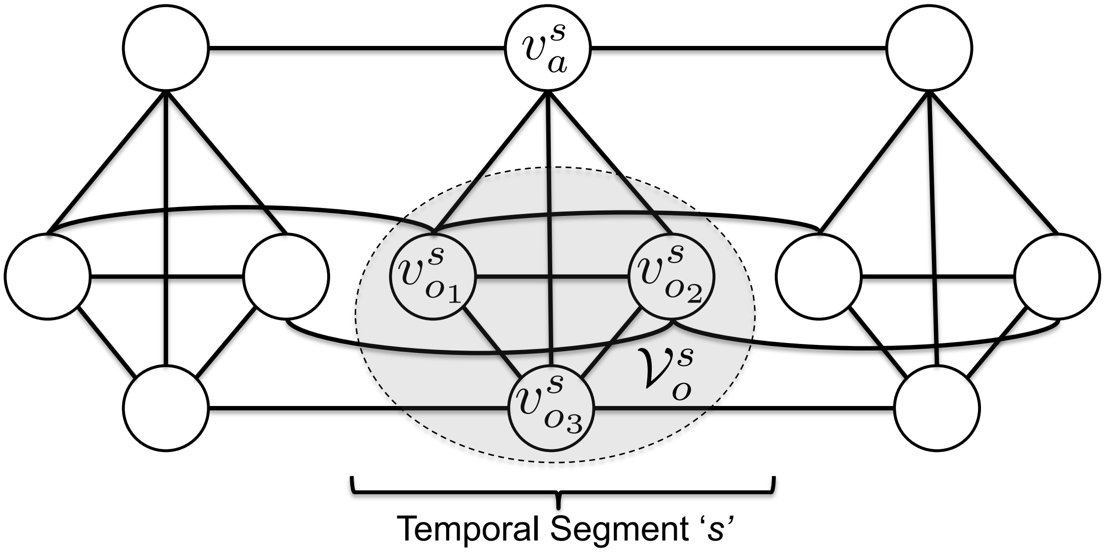


Figure 3.4: An illustrative example of our Markov random field (MRF) for three temporal segments, with one sub-activity node, v_a^s, and three object nodes, $\{v_{o_1}^s, v_{o_2}^s, v_{o_3}^s\}$, per temporal segment.

The energy function consists of six types of potentials that define the energy of a particular assignment of sub-activity and object affordance labels to the sequence of segments in the given video. The various potentials capture the dependencies between the sub-activity and object affordance labels as defined by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

We now describe the structure of this graph along with the corresponding potentials. There are two types of nodes in $\mathcal{G}$: object nodes denoted by $\mathcal{V}_o$ and sub-activity nodes denoted by $\mathcal{V}_a$. Let K_a denote the set of sub-activity labels, and K_o denote the set of object affordance labels. Let y_i^k be a binary variable representing the node i having label k, where $k \in K_o$ for object nodes and $k \in K_a$ for sub-activity nodes. All k binary variables together represent the label of a node. Let $\mathcal{V}_o^s$ denote set of object nodes of segment s, and v_a^s denote the sub-activity node of segment s. Figure 3.4 shows the graph structure for three temporal segments for an activity with three objects.

The energy term associated with labeling the object nodes is denoted by E_o and is defined as the sum of object node potentials $\psi_o(i)$ as:

$$E_o = \sum_{i \in \mathcal{V}_o} \psi_o(i) = \sum_{i \in \mathcal{V}_o} \sum_{k \in K_o} y_i^k \left[\mathbf{w}_\mathbf{o}^k \cdot \phi_o(i) \right], \tag{3.3}$$

where $\phi_o(i)$ denotes the feature map describing the object affordance of the object node i in its corresponding temporal segment through a vector of features, and there is one weight vector for each affordance class in K_o. Similarly, we have an energy term, E_a, for labeling the sub-activity nodes which is defined as the sum of sub-activity node potentials as

$$E_a = \sum_{i \in \mathcal{V}_a} \psi_a(i) = \sum_{i \in \mathcal{V}_a} \sum_{k \in K_a} y_i^k \left[\mathbf{w}_\mathbf{a}^k \cdot \phi_a(i) \right], \tag{3.4}$$

where $\phi_a(i)$ denotes the feature map describing the sub-activity node i through a vector of features, and there is one weight vector for each sub-activity class in K_a.

For all segments s, there is an edge connecting all the nodes in $\mathcal{V}_o^s$ to each other, denoted by $\mathcal{E}_{oo}$, and to the sub-activity node v_a^s, denoted by $\mathcal{E}_{oa}$. These edges signify the relationships within the objects, and between the objects and the human pose within a segment and are referred to as *'object–object interactions'* and *'sub-activity–object interactions'* in the Fig. 3.3, respectively.

The sub-activity node of segment s is connected to the sub-activity nodes in segments $(s-1)$ and $(s+1)$. These temporal edges are denoted by $\mathcal{E}_{aa}^t$. Similarly every object node of segment s is connected to the corresponding object nodes in segments $(s-1)$ and $(s+1)$, denoted by $\mathcal{E}^t_{oo}$. These edges model the *temporal interactions* between the human poses and the objects, respectively, and are represented by dotted edges in the Fig. 3.3.

We have one energy term for each of the four interaction types and they are defined as:

$$E_{oo} = \sum_{(i,j)\in\mathcal{E}_{oo}} \sum_{(l,k)\in K_o\times K_o} y_i^l y_j^k \left[\mathbf{w}_{\mathbf{oo}}^{lk} \cdot \phi_{oo}(i,j)\right], \tag{3.5}$$

$$E_{oa} = \sum_{(i,j)\in\mathcal{E}_{oa}} \sum_{(l,k)\in K_o\times K_a} y_i^l y_j^k \left[\mathbf{w}_{\mathbf{oa}}^{lk} \cdot \phi_{oa}(i,j)\right], \tag{3.6}$$

$$E^t_{oo} = \sum_{(i,j)\in\mathcal{E}^t_{oo}} \sum_{(l,k)\in K_o\times K_o} y_i^l y_j^k \left[\mathbf{w^t}_{\mathbf{oo}}^{lk} \cdot \phi^t_{oo}(i,j)\right], \tag{3.7}$$

$$E^t_{aa} = \sum_{(i,j)\in\mathcal{E}^t_{aa}} \sum_{(l,k)\in K_a\times K_a} y_i^l y_j^k \left[\mathbf{w^t}_{\mathbf{aa}}^{lk} \cdot \phi^t_{aa}(i,j)\right]. \tag{3.8}$$

The feature maps $\phi_{oo}(i,j)$ and $\phi_{oa}(i,j)$ describe the interactions between pair of objects and interactions between an object and the human skeleton within a temporal segment, respectively, and the feature maps $\phi^t_{oo}(i,j)$ and $\phi^t_{aa}(i,j)$ describe the temporal interactions between objects and sub-activities, respectively. Also, note that there is one weight vector for every pair of labels in each energy term.

Given $\mathcal{G}$, we can rewrite the energy function based on individual node potentials and edge potentials compactly as follows:

$$
\begin{aligned}
E_{\mathbf{w}}(\mathbf{x},\mathbf{y}) = & \sum_{i\in\mathcal{V}_a}\sum_{k\in K_a} y_i^k \left[\mathbf{w}_{\mathbf{a}}^k \cdot \phi_a(i)\right] + \sum_{i\in\mathcal{V}_o}\sum_{k\in K_o} y_i^k \left[\mathbf{w}_{\mathbf{o}}^k \cdot \phi_o(i)\right] \\
& + \sum_{t\in\mathcal{T}}\sum_{(i,j)\in\mathcal{E}_t}\sum_{(l,k)\in T_t} y_i^l y_j^k \left[\mathbf{w}_{\mathbf{t}}^{lk} \cdot \phi_t(i,j)\right]
\end{aligned}
\tag{3.9}
$$

where $\mathcal{T}$ is the set of the four edge types described above. Writing the energy function in this form allows us to apply efficient inference and learning algorithms as described later in Section 3.8.

3.5 Object Detection and Tracking

For producing our graph $\mathcal{G}$, we need as input the segments corresponding to the objects (but not their labels) and their tracks over time. In order to do so, we run pre-trained object detectors on a set of frames sampled from the video and then use particle filter tracker to obtain tracks of the detected objects. We then find consistent tracks that connect the various detected objects in order to find reliable object tracks. We present the details below.

Object Detection: We first train a set of 2D object detectors for the common objects present in our dataset (e.g. mugs, bowls). We use features that capture the inherent local and global properties of the object encompassing the appearance and the shape/geometry. Local features includes color histogram and the histogram of oriented gradients (HoG, Dalal and Triggs, 2005) which provide the intrinsic properties of the target object while viewpoint features histogram (VFH, Rusu et al., 2010) provides the global orientation of the normals from the object's surface. For training we used the RGB-D object dataset by [102] and built a one-vs-all SVM classifier using the local features for each object class in order to obtain the probability estimates of each object class. We also build a k-nearest neighbor (kNN) classifier over VFH features. The kNN classifier provides the detection score as inverse of the distance between training and the test instance. We obtain a final classification score by adding the scores from these two classifiers.

At test time, for a given point cloud, we first reduce the set of 3D bounding boxes by only considering those that belong to a volume around the hands of the skeleton. This reduces the number of false detections as well as the detection time. We then run our SVM-based object detectors on the RGB image. This gives us the exact x and y coordinates of the possible detections. The predictions with score above a certain threshold are further examined by calculating the kNN score based on VFH features. To find the exact depth of the object, we do pyramidal window search inside the current 3D bounding box and get the highest scoring box. In order to remove the empty space and any outlier points within a bounding box, we shrink it towards the highest-density region that captures 90% of the object points. These bounding box detections are ordered according to their final classification score.

Object Tracking: We used the particle filter tracker implementation[2] provided under the PCL library for tracking our target object. The tracker uses the color values and the normals to find the next probable state of the object.

Combining Object Detections with Tracking: We take the top detections, track them, and assign a score to each of them in order to compute the potential nodes

---

[2] http://www.willowgarage.com/blog/2012/01/17/tracking-3d-objects-point-cloud-library

in the graph $\mathcal{G}$.

We start with building a graph with the initial bounding boxes as the nodes. In our current implementation, this method needs an initial guess on the 2D bounding boxes of the objects to keep the algorithm tractable. We can obtain this by considering only the tabletop objects by using a tabletop object segmenter [e.g. 164]. We initialize the graph with these guesses. We then perform tracking through the video and grow the graph by adding a node for every object detection and connect two nodes with an edge if a track exists between their corresponding bounding boxes. Our object detection algorithm is run after every fixed number of frames, and the frames on which it is run are referred to as the detection frames. Each edge is assigned a weight corresponding to its track score as defined in Eq. (3.10). After the whole video is processed, the best track for every initial node in the graph is found by taking the highest weighted path starting at that node.

The object detections at the current frame are categorized into one of the following categories: {*merged detection*, *isolated detection*, *ignored detection*} based on their vicinity and similarity to the currently tracked objects as shown in Figure 3.5. If a detection occurs close to a currently tracked object and has high similarity with it, the detection can be merged with the current object track. Such detections are called *merged detections*. The detections which have high detection score but do not occur close to the current tracks are labeled as *isolated detections* and are tracked in both directions. This helps in correcting the tracks which have gone wrong due to partial occlusions or missing due to full occlusions of the objects. The rest of the detections are labeled as *ignored detections* and are not tracked.

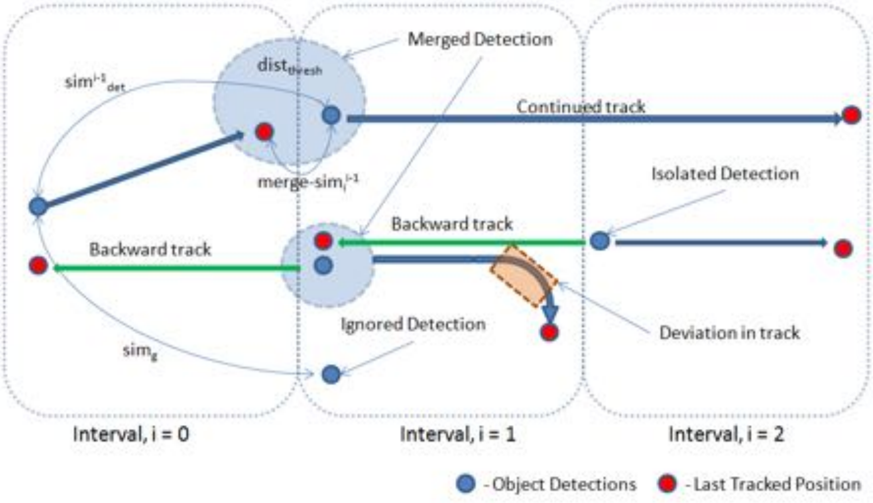


Figure 3.5: Pictorial representation of our algorithm for combining object detections with tracking.

More formally, let d_j^i represent the bounding box of the j^{th} detection in the i^{th} detection frame and let D_j^i represent its tracking score. Let $\hat{d}_j^i$ represent the tracked bounding box at the current frame with the track starting from d_j^i. We define a similarity score, $S(a, b)$, between two image bounding boxes, a and b, as the correlation score of the color histograms of the two bounding boxes. The track score of an edge e connecting the detections d_k^{i-1} and d_j^i is given by

$$ts(e) = S(d_k^{i-1}, d_j^i) + S(\hat{d}_k^{i-1}, d_j^i) + \lambda D_j^i \tag{3.10}$$

Finally, the best track of a given object bounding box b is the path having the highest cumulative track score from all paths originating at node corresponding to b in the graph, represented by the set P_b

$$\hat{t}(b) = \operatorname*{argmax}_{p \in P_b} \sum_{e \in p} ts(e). \tag{3.11}$$

3.6 Temporal Segmentation

We perform temporal segmentation of the frames in an activity in order to obtain groups of frames representing atomic movements of the human skeleton and objects in the activity. This will group similar frames into one segment, thus reducing the total number of nodes to be considered by the learning algorithm significantly.

There are several problems with naively performing one temporal segmentation—if we make a mistake here, then our followup activity detection would perform poorly. In certain cases, when the features are additive, methods based on dynamic programming [64, 177, 65] could be used to search for an optimal segmentation.

However, in our case, we have the following three challenges. First, the feature maps we consider are non-additive in nature, and the feature computation cost is exponential in the number of frames if we want to consider all the possible segmentations. Therefore, we cannot apply dynamic programming techniques to find the optimal segmentation. Second, the complex human-object interactions are poorly approximated with a linear dynamical system, therefore techniques such as [38] cannot be directly applied. Third, the boundary between two sub-activities is often not very clear, as people often start performing the next sub-activity before finishing the current sub-activity. The amount of overlap might also depend on which sub-activities are being performed. Therefore, there may not be one optimal segmentation. In our work, we consider several temporal segmentations and propose a method to combine them in Section 3.8.3.

We consider three basic methods for temporal segmentation of the video frames and generate a number of temporal segmentations by varying the parameters of these methods. The first method is uniform segmentation, in which we consider a set of continuous frames of fixed size as the temporal segment. There are two parameters for this method: the segment size and the offset (the size of the first segment). The other two segmentation methods use the graph-based segmentation proposed by [34] adapted to temporally segment the videos. The second method uses the sum of the Euclidean distances between the skeleton joints as the edge weights, whereas the third method uses the rate of change of the Euclidean distance as the edge weights. These methods consider smooth movements of the skeleton joints to belong to one segment and identify sudden changes in skeletal motion as the sub-activity boundaries.

In detail, we have one node per frame representing the skeleton in the graph based methods. Each node is connected to its temporal neighbor, therefore giving a chain graph. The algorithm begins with having each node as a separate segmentation, and iteratively merges the components if the edge weight is less than a certain threshold (computed based on the current segment size and a constant parameter). We obtain different segmentations by varying the parameter.
[3]

Table 3.1: Summary of the Node Features used in the Energy Function.

Description	Count
Object Features	**18**
N1. Centroid location	3
N2. 2D bounding box	4
N3. Transformation matrix of SIFT matches between adjacent frames	6
N4. Distance moved by the centroid	1
N5. Displacement of centroid	1
Sub-activity Features	**103**
N6. Location of each joint (8 joints)	24
N7. Distance moved by each joint (8 joints)	8
N8. Displacement of each joint (8 joints)	8
N9. Body pose features	47
N10. Hand position features	16

3.7 Features

For a given object node i, the node feature map $\phi_o(i)$ is a vector of features representing the object's location in the scene and how it changes within the temporal segment. These features include the (x, y, z) coordinates of the object's centroid and the coordinates of the object's bounding box at the middle frame of the

---

[3] Details: In order to handle occlusions, we only use the upper body skeleton joints for computing the edge weights that are estimated more reliably by the skeleton tracker. When changing the parameters for the three segmentation methods for obtaining multiple segmentations, we select the parameters such that we always err on the side of over-segmentation instead of under-segmentation. This is because our learning model can handle over-segmentation by assigning the same label to the consecutive segments for the same sub-activity, but under-segmentation is bad as the model can only assign one label to that segment.

Table 3.2: Summary of the Edge Features used in the Energy Function.

Description	Count
Object-object Features (computed at start frame, middle frame, end frame, max and min)	**20**
E1. Difference in centroid locations $(\Delta x, \Delta y, \Delta z)$ E2. Distance between centroids	3 1
Object–sub-activity Features (computed at start frame, middle frame, end frame, max and min)	**40**
E3. Distance between each joint location and object centroid	8
Object Temporal Features	**4**
E4. Total and normalized vertical displacement E5. Total and normalized distance between centroids	2 2
Sub-activity Temporal Features	**16**
E6. Total and normalized distance between each corresponding joint locations (8 joints)	16

temporal segment. We also run a SIFT feature based object tracker [150] to find the corresponding points between the adjacent frames and then compute the transformation matrix based on the matched image points. We add the transformation matrix corresponding to the object in the middle frame with respect to its previous frame to the features in order to capture the object's motion information. In addition to the above features, we also compute the total displacement and the total distance moved by the object's centroid in the set of frames belonging to the temporal segment. See 3.1 for summary of the node features. We then perform cumulative binning of the feature values into 10 bins. In our experiments, we have $\phi_o(i) \in \mathbb{R}^{180}$.

Similarly, for a given sub-activity node i, the node feature map $\phi_a(i)$ gives a vector of features computed using the human skeleton information obtained from running Openni's skeleton tracker[4] on the RGB-D video. We compute the features described above for each of the upper-skeleton joint (neck, torso, left shoulder, left elbow, left palm, right shoulder, right elbow and right palm) locations relative to the subject's head location. In addition to these, we also consider the body pose and hand position features as described by [185], thus giving us $\phi_a(i) \in \mathbb{R}^{1030}$.

The edge feature maps $\phi_t(i, j)$ describe the relationship between node i and j. They are used for modeling four types of interactions: object–object within a temporal segment, object–sub-activity within a temporal segment, object–object between two temporal segments, and sub-activity–sub-activity between two temporal segments. See 3.2 for summary of the edge features. For capturing the *object-object* relations within a temporal segment, we compute relative geometric features such as the difference in (x, y, z) coordinates of the object centroids and the distance between them. These features are computed at the first, middle and last frames of the temporal segment along with minimum and maximim of their values across all frames in the temporal segment to capture the relative motion information. This gives us $\phi_1(i, j) \in \mathbb{R}^{200}$. Similarly for *object–sub-activity* relation features $\phi_2(i, j) \in \mathbb{R}^{400}$, we use the same features as for the *object–object* relation features, but we compute them between the upper-skeleton joint locations and each object's centroid. The temporal relational features capture the change across temporal segments and we use the vertical change in position and the distance between the corresponding object and the joint locations. This

---

[4] http://openni.org

gives us $\phi_3(i, j) \in \mathbb{R}^{40}$ and $\phi_4(i, j) \in \mathbb{R}^{160}$ respectively.

3.8 Inference and Learning Algorithm

3.8.1 Inference.

Given the model parameters w, the inference problem is to find the best labeling $\hat{\mathbf{y}}$ for a new video $\mathbf{x}$, i.e. solving the argmax in Eq. (3.1) for the discriminant function in Eq. (3.9). This is a NP hard problem. However, its equivalent formulation as the following mixed-integer program has a linear relaxation which can be solved efficiently as a quadratic pseudo-Boolean optimization problem using a graph-cut method [162].

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \max_{\mathbf{z}} \sum_{i \in \mathcal{V}_a} \sum_{k \in K_a} y_i^k \left[\mathbf{w_a^k} \cdot \phi_a(i) \right] + \sum_{i \in \mathcal{V}_o} \sum_{k \in K_o} y_i^k \left[\mathbf{w_o^k} \cdot \phi_o(i) \right]$$
$$+ \sum_{t \in \mathcal{T}} \sum_{(i,j) \in \mathcal{E}_t} \sum_{(l,k) \in T_t} z_{ij}^{lk} \left[\mathbf{w_t^{lk}} \cdot \phi_t(i, j) \right] \tag{3.12}$$

$$\forall i, j, l, k\colon\ \ z_{ij}^{lk} \le y_i^l,\ \ z_{ij}^{lk} \le y_j^k,\ \ y_i^l + y_j^k \le z_{ij}^{lk} + 1,\ \ z_{ij}^{lk}, y_i^l \in \{0, 1\} \tag{3.13}$$

Note that the products $y_i^l y_j^k$ have been replaced by auxiliary variables z_{ij}^{lk}. Relaxing the variables z_{ij}^{lk} and y_i^l to the interval $[0, 1]$ results in a linear program that can be shown to always have half-integral solutions (i.e. y_i^l only take values $\{0, 0.5, 1\}$ at the solution) [52]. Since every node in our experiments has exactly one class label, we also consider the linear relaxation from above with the additional constraints $\forall i \in \mathcal{V}_a : \sum_{l \in K_a} y_i^l = 1$ and $\forall i \in \mathcal{V}_o : \sum_{l \in K_o} y_i^l = 1$. This problem can no longer be solved via graph cuts. We compute the exact mixed integer solution including these additional constraint using a general-purpose

MIP solver[5] during inference.

In our experiments, we obtain a processing rate of *74.9 frames/second* for inference and *16.0 frames/second* end-to-end (including feature computation cost) on a 2.93 GHz Intel processor with 16 GB of RAM on Linux. In detail, the MIP solver takes 6.94 seconds for a typical video with 520 frames and the corresponding graph has 12 sub-activity nodes and 36 object nodes, i.e. 15908 variables. This is the time corresponding to solving the argmax in Eq. (3.12-3.13) and does not involve the feature computation time. The time taken for end-to-end classification including feature generation is 32.5 seconds.

3.8.2 Learning

We take a large-margin approach to learning the parameter vector $\mathbf{w}$ of Eq. (3.9) from labeled training examples $(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_M, \mathbf{y}_M)$ [190, 195]. Our method optimizes a regularized upper bound on the training error

$$R(h) = \frac{1}{M} \sum_{m=1}^{M} \Delta(\mathbf{y}_m, \hat{\mathbf{y}}_m),$$

where $\hat{\mathbf{y}}_m$ is the optimal solution of Eq. (3.1) and $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ is the loss function defined as

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i \in \mathcal{V}_o} \sum_{k \in K_o} |y_i^k - \hat{y}_i^k| + \sum_{i \in \mathcal{V}_a} \sum_{k \in K_a} |y_i^k - \hat{y}_i^k|.$$

To simplify notation, note that Eq. (3.12) can be equivalently written as $\mathbf{w}^T \Psi(\mathbf{x}, \mathbf{y})$ by appropriately stacking the $\mathbf{w}_\mathbf{a}^k$, $\mathbf{w}_\mathbf{o}^k$ and $\mathbf{w}_\mathbf{t}^{lk}$ into $\mathbf{w}$ and the $y_i^k \phi_a(i)$, $y_i^k \phi_o(i)$ and $z_{ij}^{lk} \phi_t(i, j)$ into $\Psi(\mathbf{x}, \mathbf{y})$, where each z_{ij}^{lk} is consistent with Eq. (3.13)

---

[5] http://www.tfinley.net/software/pyglpk/readme.html

given $\mathbf{y}$. Training can then be formulated as the following convex quadratic program [81]:

$$\begin{aligned}
\min_{w,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \qquad (3.14)\\
s.t. \quad & \forall \bar{\mathbf{y}}_1, ..., \bar{\mathbf{y}}_M \in \{0, 0.5, 1\}^{N \cdot K}:\\
& \frac{1}{M}\mathbf{w}^T \sum_{m=1}^{M} [\Psi(\mathbf{x}_m, \mathbf{y}_m) - \Psi(\mathbf{x}_m, \bar{\mathbf{y}}_m)] \geq \Delta(\mathbf{y}_m, \bar{\mathbf{y}}_m) - \xi
\end{aligned}$$

While the number of constraints in this QP is exponential in M, N and K, it can nevertheless be solved efficiently using the cutting-plane algorithm [81]. The algorithm needs access to an efficient method for computing

$$\bar{\mathbf{y}}_m = \underset{\mathbf{y} \in \{0,0.5,1\}^{N \cdot K}}{\operatorname{argmax}} \left[\mathbf{w}^T \Psi(\mathbf{x}_m, \mathbf{y}) + \Delta(\mathbf{y}_m, \mathbf{y})\right]. \qquad (3.15)$$

Due to the structure of $\Delta(.,.)$, this problem is identical to the relaxed prediction problem in Eqs. (3.12)-(3.13) and can be solved efficiently using graph cuts.

3.8.3 Multiple Segmentations

Segmenting an RGB-D video in time can be noisy, and multiple segmentations may be valid. Therefore, we perform multiple segmentations by using different methods and criterion of segmentation (see Section 3.6 for details). Thus, we get a set of multiple segmentations, $\mathcal{H}$. Let h_n be the n^{th} segmentation. A discriminant function $E_{\mathbf{w}^{h_n}}(\mathbf{x}^{h_n}, \mathbf{y}^{h_n})$ can now be defined for each h_n as in Eq. (3.9). We now define a score function $g_\theta(\mathbf{y}^{h_n}, \mathbf{y})$ which gives a score for assigning the labels of the segments from $\mathbf{y}^{h_n}$ to $\mathbf{y}$,

$$g_{\theta_n}(\mathbf{y}^{h_n}, \mathbf{y}) = \sum_{k \in K} \sum_{i \in \mathcal{V}} \theta_n^k \mathbf{y}_i^{h_n k} \mathbf{y}_i^k \tag{3.16}$$

where $K = K_o \cup K_a$. Here, θ_n^k can be interpreted as the confidence of labeling the segments of label k correctly in the n^{th} segmentation hypothesis. We want to find the labeling that maximizes the assignment score across all the segmentations. Therefore we can write inference in terms of a joint objective function as follows

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}} \max_{\mathbf{y}^{h_n} \forall h_n \in \mathcal{H}} \sum_{h_n \in \mathcal{H}} [E_{\mathbf{w}^{h_n}}(\mathbf{x}^{h_n}, \mathbf{y}^{h_n}) + g_{\theta_n}(\mathbf{y}^{h_n}, \mathbf{y})] \tag{3.17}$$

This formulation is equivalent to considering the labelings y^{h_n} over the segmentations as unobserved variables. It is possible to use the latent structural SVM [213] to solve this, but it becomes intractable if the size of the segmentation hypothesis space is large. Therefore we propose an approximate two-step learning procedure to address this. For a given set of segmentations $\mathcal{H}$, we first learn the parameters $\mathbf{w}^{h_n}$ independently as described in Section 3.4. We then train the parameters θ on a separate held-out training dataset. This can now be formulated as a QP

$$\begin{aligned} \min_{\theta} \quad & \frac{1}{2}\theta^T \theta - \sum_{h_n \in \mathbf{H}} g_{\theta_n}(\mathbf{y}^{h_n}, \mathbf{y}) \\ s.t. \quad & \forall k \in K : \sum_{n=1}^{|\mathbf{H}|} \theta_n^k = 1 \end{aligned} \tag{3.18}$$

Using the fact that the objective function defined in Eq. (3.17) is convex, we design an iterative two-step procedure where we solve for $\mathbf{y}^{h_n}, \forall h_n \in \mathbf{H}$ in parallel and then solve for $\mathbf{y}$. This method is guaranteed to converge, and when the number of variables scales linearly with the number of segmentation hypothesis considered, the original problem in Eq. (3.17) will become considerably slow, but our method will still scale. More formally, we iterate between the following two problems:

$$\hat{\mathbf{y}}^{h_n} = \operatorname*{argmax}_{\mathbf{y}^{h_n}} E_{\mathbf{w}^{h_n}}(\mathbf{x}^{h_n}, \mathbf{y}^{h_n}) + g_{\theta_n}(\mathbf{y}^{h_n}, \hat{\mathbf{y}}) \tag{3.19}$$

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} g_{\theta_n}(\hat{\mathbf{y}}^{h_n}, \mathbf{y}) \tag{3.20}$$

3.8.4 High-level Activity Classification.

For classifying the high-level activity, we compute the histograms of sub-activity and affordance labels and use them as features. However, some high-level activities, such as *stacking objects* and *unstacking objects*, have the same sub-activity and affordance sequences. Occlusion of objects plays a major role in being able to differentiate such activities. Therefore, we compute additional occlusion features by dividing the video into n uniform length segments and finding the fraction of objects that are occluded fully or partially in the temporal segments. We then train a multi-class SVM classifier on training data using these features.

3.9 Experiments

3.9.1 Data

We test our model on two 3D activity datasets: Cornell Activity Dataset - 60 [CAD-60, 185] and one that we collected. The CAD-60 dataset has 60 RGB-D videos of four different subjects performing 12 high-level activity classes. However, some of these activity classes contain only one sub-activity (e.g. *working on a computer*, *cooking (stirring)*, etc.) and do not contain object interactions (e.g. *talking on couch*, *relaxing on couch*).

We collected the CAD-120 dataset (available at: `http://pr.cs.cornell.edu/humanactivities`, along with the code), which contains 120 activity sequences of ten different high-level activities performed by four different subjects, where each high-level activity was performed three times. We thus have a total of **61,585** RGB-D video frames in our dataset. The high-level activities are {*making cereal*, *taking medicine*, *stacking objects*, *unstacking objects*, *microwaving food*, *picking objects*, *cleaning objects*, *taking food*, *arranging objects*, *having a meal*}.

The subjects were only given a high-level description of the task,[6] and were asked to perform the activities multiple times with *different* objects. For example, the stacking and unstacking activities were performed with pizza boxes, plates and bowls. They performed the activities through a long sequence of sub-activities, which varied from subject to subject significantly in terms of length of the sub-activities, order of the sub-activities as well as in the way they executed

---

[6] For example, the instructions for *making cereal* were: 1) Place bowl on table, 2) Pour cereal, 3) Pour milk. For *microwaving food*, they were: 1) Open microwave door, 2) Place food inside, 3) Close microwave door.

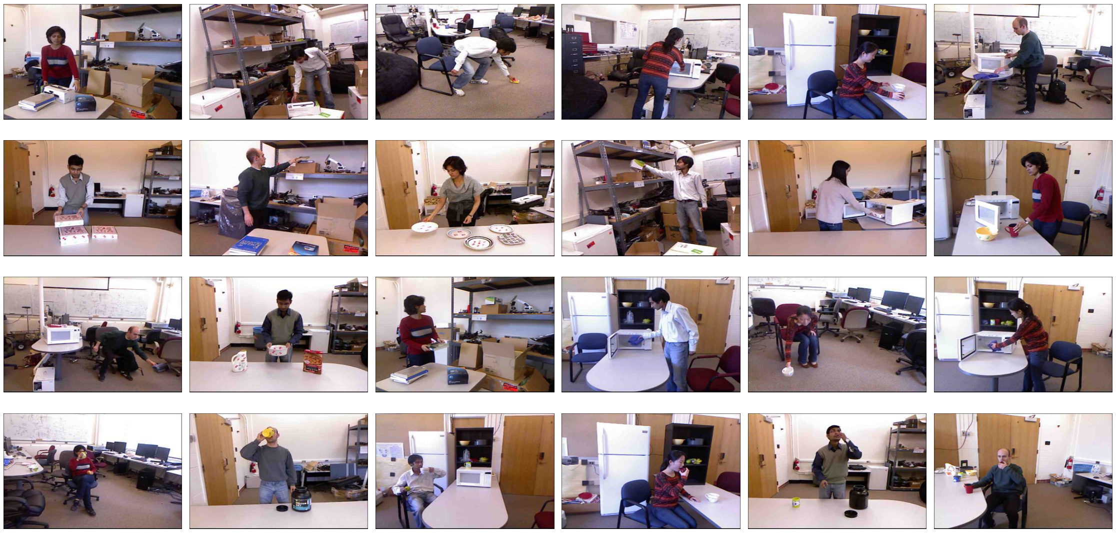


Figure 3.6: Example shots of *reaching* (first row), *placing* (second row), *moving* (third row), *drinking* (fourth row) and *eating* (fourth row) sub-activities from our dataset. There are significant variations in the way the subjects perform the sub-activity.

the task. Table 3.3 specifies the set of sub-activities involved in each high-level activity. The camera was mounted so that the subject was in view (although the subject may not be facing the camera), but often there were significant occlusions of the body parts. See Fig. 3.2 and Fig. 3.6 for some examples.

We labeled our CAD-120 dataset with the sub-activity and the object affordance labels. Specifically, our sub-activity labels are {*reaching*, *moving*, *pouring*, *eating*, *drinking*, *opening*, *placing*, *closing*, *scrubbing*, *null*} and our affordance labels are {*reachable*, *movable*, *pourable*, *pourto*, *containable*, *drinkable*, *openable*, *placeable*, *closable*, *scrubbable*, *scrubber*, *stationary*}.

Table 3.3: **Description of high-level activities in terms of sub-activities.** Note that some activities consist of same sub-activities but are executed in different order. The high-level activities (rows) are learnt using the algorithm in Section 3.8.4 and the sub-activities (columns) are learnt using the algorithm in Section 3.8.2.

	reaching	moving	placing	opening	closing	eating	drinking	pouring	scrubbing	null
Making Cereal	✓	✓	✓					✓		✓
Taking Medicine	✓	✓	✓	✓		✓	✓			✓
Stacking Objects	✓	✓	✓							✓
Unstacking Objects	✓	✓	✓							✓
Microwaving Food	✓	✓	✓	✓	✓					✓
Picking Objects	✓	✓								✓
Cleaning Objects	✓	✓		✓	✓				✓	✓
Taking Food	✓		✓	✓	✓					✓
Arranging Objects	✓	✓	✓							✓
Having a Meal	✓	✓				✓	✓			✓

3.9.2 Object Tracking Results

In order to evaluate our object detection and tracking method, we have generated the ground-truth bounding boxes of the objects involved in the activities. We do this by manually labeling the object bounding boxes in the images corresponding to every 50^{th} frame. We compute the bounding boxes in the rest of the frames by tracking using SIFT feature matching [150], while enforcing depth consistency across the time frames for obtaining reliable object tracks.

Fig. 3.7 shows the visual output of our tracking algorithm. The center of the bounding box for each frame of the output is marked with a blue dot and that of the ground-truth is marked with a red dot. We compute the overlap of the bounding boxes obtained from our tracking method with the generated ground-truth bounding boxes. Table 3.4 shows the percentage overlap with the ground-truth when considering tracking from the given bounding box in the

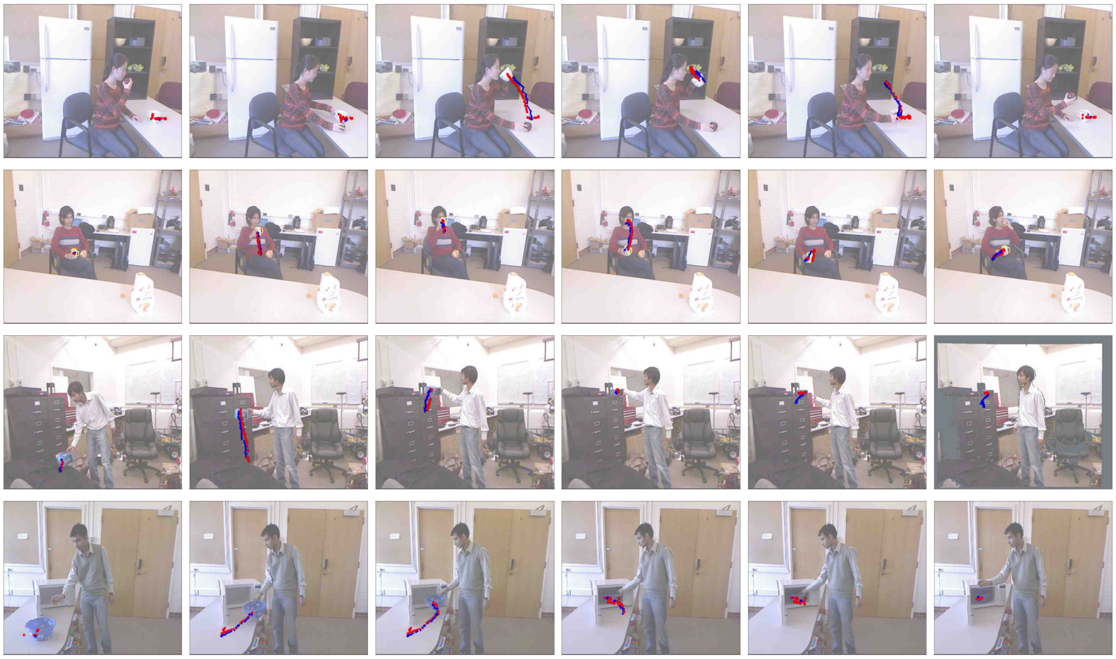


Figure 3.7: Tracking Results: Blue dots represent the trajectory of the center of tracked bounding box and red dots represent the trajectory of the center of ground-truth bounding box. (Best viewed in color.)

first frame both with and without object detections. As can be seen from Table 3.4, our tracking algorithm produces greater than 10% overlap with the ground truth bounding boxes for 77.8% of the frames. Since, we only require that an approximate bounding box of the objects are given, 10% overlap is sufficient. We study the effect of the errors in tracking on the performance of our algorithm in Section 3.9.4.

Table 3.4: **Object Tracking Results**, showing the % of frames which have ≥40%, ≥20% and ≥10% overlap with the ground-truth object bounding boxes.

	≥40%	≥20%	≥10%
tracking w/o detection	49.2	65.7	75
tracking + detection	53.5	69.4	77.8

Table 3.5: **Results on Cornell Activity Dataset [185]**, tested on *"New Person"* data for 12 activity classes.

	bathroom		bedroom		kitchen		living room		office		Average	
	prec. (%)	rec. (%)	prec. (%)	rec. (%)	prec. (%)	rec. (%)	prec. (%)	rec. (%)	prec. (%)	rec. (%)	prec. (%)	rec. (%)
[185]	72.7	**65.0**	**76.1**	59.2	64.4	47.9	52.6	45.7	73.8	59.8	67.9	55.5
Our method	**88.9**	61.1	73.0	**66.7**	**96.4**	**85.4**	**69.2**	**68.7**	**76.7**	**75.0**	**80.8**	**71.4**

3.9.3 Labeling results on the Cornell Activity Dataset 60 (CAD-60)

Table 3.5 shows the precision and recall of the high-level activities on the CAD-60 dataset [185]. Following Sung et al.'s [2012] experiments, we considered the same five groups of activities based on their location, and learnt a separate model for each location. To make it a fair comparison, we do not assume perfect segmentation of sub-activities and do not use any object information. Therefore, we train our model with only sub-activity nodes and consider segments of uniform size (20 frames per segments). We consider only a subset of our features described in Section 3.4 that are possible to compute from the tracked human skeleton and RGB-D data provided in this dataset. Table 3.5 shows that our model significantly outperforms Sung et al.'s MEMM model even when using only the sub-activity nodes and a simple segmentation algorithm.

3.9.4 Labeling results on the Cornell Activity Dataset 120 (CAD-120)

Table 3.6 shows the performance of various models on object affordance, sub-activity and high-level activity labeling. These results are obtained using 4-fold

Table 3.6: **Results on our CAD-120 dataset**, showing average micro precision/recall, and average macro precision and recall for affordance, sub-activities and high-level activities. Standard error is also reported.

Full model, assuming ground-truth temporal segmentation is given.

	Object Affordance			Sub-activity			High-level Activity		
	micro	macro		micro	macro		micro	macro	
method	*P/R* (%)	Prec. (%)	Recall (%)	*P/R* (%)	Prec. (%)	Recall (%)	P/R (%)	Prec. (%)	Recall (%)
max class	65.7 ± 1.0	65.7 ± 1.0	8.3 ± 0.0	29.2 ± 0.2	29.2 ± 0.2	10.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0	10.0 ± 0.0
image only	74.2 ± 0.7	15.9 ± 2.7	16.0 ± 2.5	56.2 ± 0.4	39.6 ± 0.5	41.0 ± 0.6	34.7 ± 2.9	24.2 ± 1.5	35.8 ± 2.2
SVM multiclass	75.6 ± 1.8	40.6 ± 2.4	37.9 ± 2.0	58.0 ± 1.2	47.0 ± 0.6	41.6 ± 2.6	30.6 ± 3.5	27.4 ± 3.6	31.2 ± 3.7
MEMM [185]	-	-	-	-	-	-	26.4 ± 2.0	23.7 ± 1.0	23.7 ± 1.0
object only	86.9 ± 1.0	72.7 ± 3.8	63.1 ± 4.3	-	-	-	59.7 ± 1.8	56.3 ± 2.2	58.3 ± 1.9
sub-activity only	-	-	-	71.9 ± 0.8	60.9 ± 2.2	51.9 ± 0.9	27.4 ± 5.2	31.8 ± 6.3	27.7 ± 5.3
no temporal interactions	87.0 ± 0.8	79.8 ± 3.6	66.1 ± 1.5	76.0 ± 0.6	74.5 ± 3.5	66.7 ± 1.4	81.4 ± 1.3	83.2 ± 1.2	80.8 ± 1.4
no object interactions	88.4 ± 0.9	75.5 ± 3.7	63.3 ± 3.4	85.3 ± 1.0	79.6 ± 2.4	74.6 ± 2.8	80.6 ± 2.6	81.9 ± 2.2	80.0 ± 2.6
full model	91.8 ± 0.4	90.4 ± 2.5	74.2 ± 3.1	86.0 ± 0.9	84.2 ± 1.3	76.9 ± 2.6	84.7 ± 2.4	85.3 ± 2.0	84.2 ± 2.5
full model with tracking	88.2 ± 0.6	74.5 ± 4.3	64.9 ± 3.5	82.5 ± 1.4	72.9 ± 1.2	70.5 ± 3.0	79.0 ± 4.7	78.6 ± 4.1	78.3 ± 4.9

Full model, *without* assuming any ground-truth temporal segmentation is given.

full, 1 segment (best)	83.1 ± 1.1	70.1 ± 2.3	63.9 ± 4.4	66.6 ± 0.7	62.0 ± 2.2	60.8 ± 4.5	77.5 ± 4.1	80.1 ± 3.9	76.7 ± 4.2
full, 1 segment (avg.)	81.3 ± 0.4	67.8 ± 1.1	60.0 ± 0.8	64.3 ± 0.7	63.8 ± 1.1	59.1 ± 0.5	79.0 ± 0.9	81.1 ± 0.8	78.3 ± 0.9
full, multi-seg learning	83.9 ± 1.5	75.9 ± 4.6	64.2 ± 4.0	68.2 ± 0.3	71.1 ± 1.9	62.2 ± 4.1	80.6 ± 1.1	81.8 ± 2.2	80.0 ± 1.2
full, multi-seg + tracking	79.4 ± 0.8	62.5 ± 5.4	50.2 ± 4.9	63.4 ± 1.6	65.3 ± 2.3	54.0 ± 4.6	75.0 ± 4.5	75.8 ± 4.4	74.2 ± 4.6

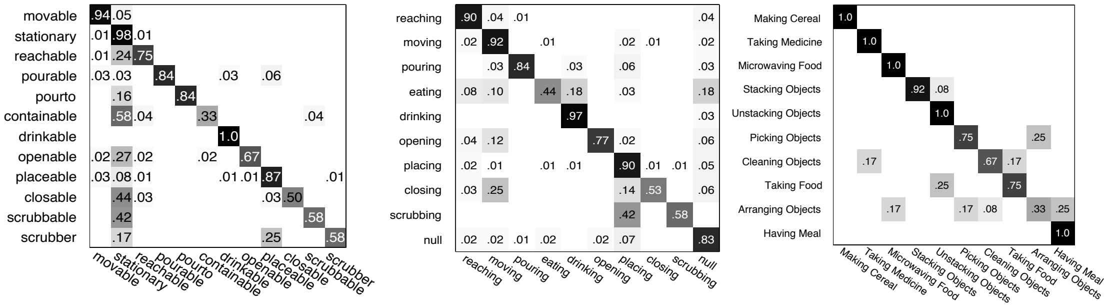
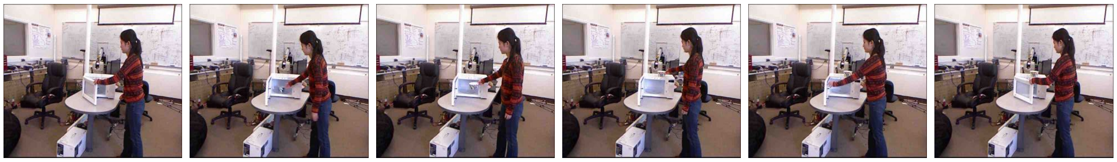


Figure 3.8: Confusion matrix for affordance labeling (left), sub-activity labeling (middle) and high-level activity labeling (right) of the test RGB-D videos.

cross-validation and averaging performance across the folds. Each fold constitutes the activities performed by one subject, therefore the model is trained on activities of three subjects and *tested on a new subject*. We report both the micro and macro averaged precision and recall over various classes along with standard error. Since our algorithm can only predict one label for each segment, micro precision and recall are same as the percentage of correctly classified segments. Macro precision and recall are the averages of precision and recall respectively for all classes.

Assuming ground-truth temporal segmentation is given, the results for our *full model* are shown in Table 3.6 on line 10, its variations on lines 5–9 and the baselines on lines 1–4. The results in lines 11–14 correspond to the case when temporal segmentation is not assumed. In comparison to a basic SVM multiclass model [81] (referred to as *SVM multiclass* when using all features and *image only* when using only image features), which is equivalent to only considering the nodes in our MRF without any edges, our model performs significantly better. We also compare with the high-level activity classification results obtained from the method presented in [185]. We ran their code on our dataset and obtain accuracy of 26.4%, whereas our method gives an accuracy of 84.7% when ground truth segmentation is available and 80.6% otherwise. Figure 3.9 shows a sequence of images from *taking food* activity along with the inferred labels. Figure 3.8 shows the confusion matrix for labeling affordances, sub-activities and high-level activities with our proposed method. We can see that there is a strong diagonal with a few errors such as *scrubbing* misclassified as *placing*, and *picking objects* misclassified as *arranging objects*.

We analyze our model to gain insight into which interactions provide useful



Subject *opening* *openable* object1

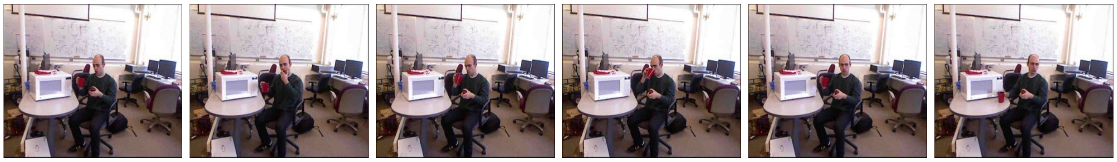
Subject *reaching* *reachable* object2

Subject *moving* *movable* object2

Subject *placing* *placable* object2

Subject *reaching* *reachable* object1

Subject *closing* *closable* object1



Subject *moving* *movable* object1

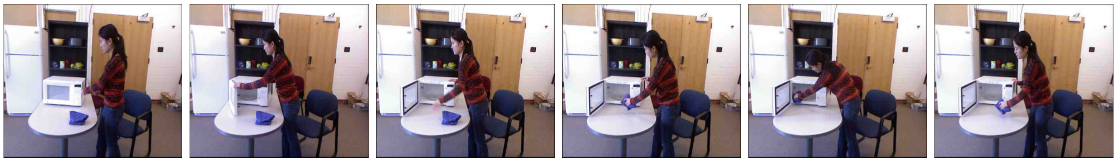
Subject *eating*

Subject *moving* *movable* object1

Subject *drinking* from *drinkable* object1

Subject *moving* *movable* object1

Subject *placing* *placeable* object1



Subject *reaching* *reachable* object1

Subject *opening* *openable* object1

Subject *reaching*

Subject *moving* *movable* object2

Subject *scrubbing* *scrubbable* object1 with *scrubber* object2

Subject *moving* *movable* object2

Figure 3.9: Descriptive output of our algorithm: Sequence of images from the *taking food* (Top Row), *having meal* (Middle Row) and *cleaning objects* (Bottom Row) activities labeled with sub-activity and object affordance labels. A single frame is sampled from the temporal segment to represent it.

information by comparing our full model to variants of our model.

How important is object context for activity detection? We show the importance of object context for sub-activity labeling by learning a variant of our model without the object nodes (referred to as *sub-activity only*). With object context, the micro precision increased by 14.1% and both macro precision and recall increased by around 23.3% over *sub-activity only*. Considering object information (affordance labels and occlusions) also improved the high-level activity accuracy by three-fold.

How important is activity context for affordance detection? We also show the importance of context from sub-activity for affordance detection by learning our model without the sub-activity nodes (referred to as *object only*). With sub-activity context, the micro precision increased by 4.9% and the macro precision and recall increased by 17.7% and 11.1% respectively for affordance labeling over *object only*. The relative gain is less compared with that obtained in sub-activity detection as the *object only* model still has object–object context which helps in affordance detection.

How important is object–object context for affordance detection? In order to study the effect of the object–object interactions for affordance detection, we learnt our model without the object-object edge potentials (referred to as *no object interactions*). We see a considerable improvement in affordance detection when the object interactions are modeled, the macro recall increased by 14.9% and the macro precision by about 10.9%. This shows that sometimes just the context from the human activity alone is not sufficient to determine the affordance of an object.

How important is temporal context? We also learn our model without the temporal edges (referred to as *no temporal interactions*). Modeling temporal interactions increased the micro precision by 4.8% and 10.0% for affordances and sub-activities respectively and increased the micro precision for high-level activity by 3.3%.

How important is reliable human pose detection? In order to understand the effect of the errors in human pose tracking, we consider the affordances that require direct contact by human hands, such as movable, openable, closable, drinkable, etc. The distance of the predicted hand locations to the object should be zero at the time of contact. We found that for the correct predictions, these distances had a mean of 3.8 cm and a variance of 48.1 cm. However, for the incorrect predictions, these distances had a mean that was 43.3% higher and a variance that was 53.8% higher. This indicates that the prediction accuracies can potentially be improved with more robust human pose tracking.

How important is reliable object tracking? We show the effect of having reliable object tracking by comparing to the results obtained from using our object tracking algorithm mentioned in Section 3.5. We see that using the object tracks generated by our algorithm gives slightly lower micro precision/recall values compared with using ground-truth object tracks, around 3.5% drop in affordance and sub-activity detection, and 5.7% drop in high-level activity detection. The drop in macro precision and recall are higher, which shows that the performance of few classes are affected more than the others. In future work, one can increase accuracy by improving object tracking.

Results with multiple segmentations. Given the RGB-D video and initial bounding boxes for objects in the first frame, we obtain the final labeling using our method described in Section 3.8.3. To generate the segmentation hypothesis set $\mathcal{H}$ we consider three different segmentation algorithms, and generate multiple segmentations by changing their parameters as described in Section 3.6. The lines 11–13 of Table 3.6 show the results of the best performing segmentation, average performance all the segmentations considered, and our proposed method for combining the segmentations respectively. We see that

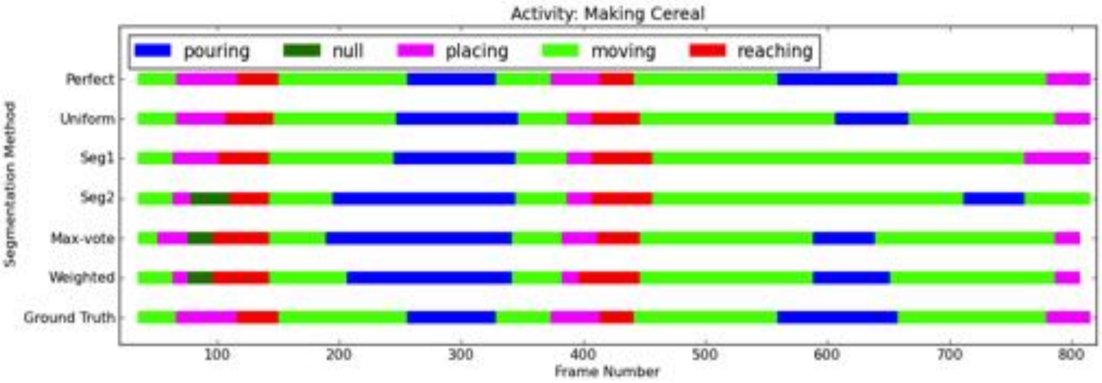


Figure 3.10: Comparison of the sub-activity labeling of various segmentations. This activity involves the sub-activities: *reaching*, *moving*, *pouring* and *placing* as colored in red, green, blue and magenta respectively. The x-axis denotes the time axis numbered with frame numbers. It can be seen that the various individual segmentation labelings are not perfect and make different mistakes, but our method for merging these segmentations selects the correct label for many frames.

our method improves the performance over considering a single best performing segmentation: macro precision increased by 5.8% and 9.1% for affordance and sub-activity labeling respectively. Fig. 3.10 shows the comparison of the sub-activity labeling of various segmentations, our end-to-end labeling and the ground-truth labeling for one *making cereal* high-level activity video. It can be seen that the various individual segmentation labelings are not perfect and make different mistakes, but our method for merging these segmentations selects the correct label for many frames. Line 14 of Table 3.6 show the results of our proposed method for combining the segmentations along with using our object tracking algorithm. The numbers show a drop compared with the case of using ground-truth tracks, therefore providing a scope for improvement by using more reliable tracking algorithms.

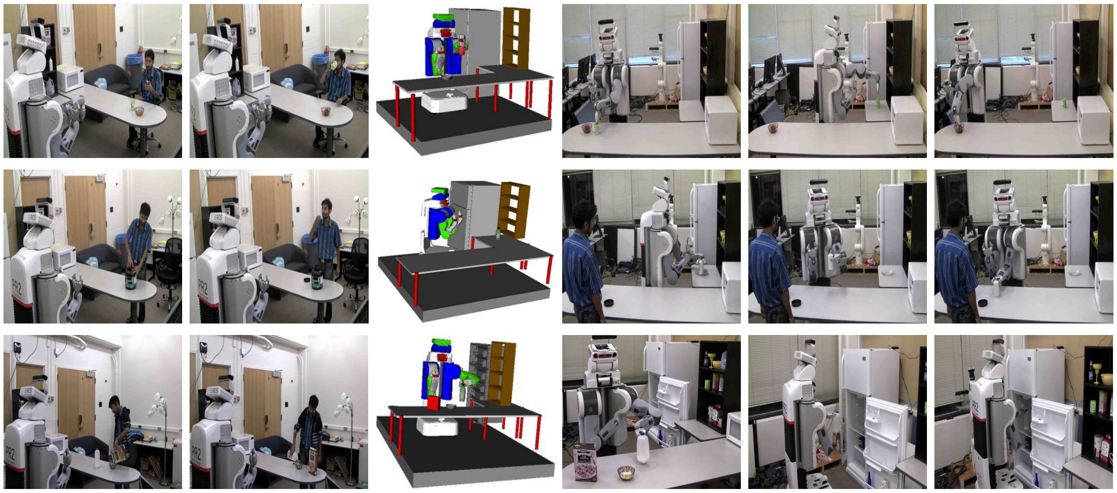


Figure 3.11: Robot performing the task of assisting humans: (top row) robot clearing the table after detecting *having a meal* activity, (middle row) robot fetching a bottle of water after detecting *taking a medicine* activity and (third row) robot putting milk in the fridge after detecting *making cereal* activity. First two columns show the robot observing the activity, third row shows the robot planning the response in simulation and the last three columns show the robot performing the response action.

3.9.5 Robotic Applications

We demonstrate the use of our learning algorithm in two robotics applications. First, we show that the knowledge of the activities currently being performed enables a robot to assist the human by performing an appropriate response action. Second, we show that the knowledge of the affordances of the objects enables a robot to use them appropriately when manipulating them.

We use Cornell's Kodiak, a PR2 robot, in our experiments. Kodiak is mounted with a Microsoft Kinect, which is used as the main input sensor to obtain the RGB-D video stream. We used the OpenRAVE libraries [25] for programming the robot to perform the pre-programmed assistive tasks.

Assisting Humans. There are several modes of operation for a robot performing assistive tasks. For example, the robot can perform some tasks completely autonomously, independent of the humans. For some other tasks, the robot needs to act more reactively. That is, depending on the task and current human activity taking place, perform a complementary sub-task. For example, bring a glass of water when a person is attempting to take medicine (and there is no glass within a person's reach). Such a behavior is possible only when the activities are successfully detected. In this experiment, we demonstrate that our algorithm for detecting the human activities enables a robot to take such (pre-programmed) reactive actions.[7]

We consider the following three scenarios:

- *Having Meal*: The subject eats food from a bowl and drinks water from a cup in this activity. On detecting the *having meal* activity, the robot assists by clearing the table (i.e. move the cup and the bowl to another place) after the subject finishes eating.
- *Taking Medicine*: The subjects opens the medicine container, takes the medicine, and waits as there is no water nearby. The robot assists the subject by bringing a glass of water on detecting the *taking medicine* activity.
- *Making Cereal*: The subject prepares cereal by pouring cereal and milk in to a bowl. On detecting the activity, the robot responds by taking the milk and putting it into the refrigerator.

Our robot was placed in a kitchen environment so that it can observe the activity

---

[7] Our goal is activity detection, therefore we pre-program the response actions using existing open-source tools in ROS. In future, one would need to make significant advances in several fields to make this useful in practice, e.g., object detection [96, 6], grasping [172, 79], human-robot interaction, and so on.

being performed by the subject. We found that our robot successfully detected the activities and performed the above described reactive actions. Fig. 3.11 shows the sequence of images of the robot detecting the activity being performed, planning the response in simulation and then performing the appropriate response for all three activities described above.

Using Affordances. An important component of our work is to learn affordances. In particular, by observing how the humans interact with the objects, a robot can figure out the affordances of the objects. Therefore, it can use these inferred affordances to interact with objects in a meaningful way. For example, given an instruction of 'clear the table', the robot should be able to perform the response in a desirable way: move the bowl with cereal *without* tilting it, and not move the microwave. In this experiment, we demonstrate that our algorithm for labeling the affordances explicitly helps in manipulation.

In our setting, we directly infer the object affordances (movable, pourable, drinkable, etc.). Therefore, we only need to encode the low-level control actions of each affordance, e.g. to move only *movable* objects, and to execute constrained movement, i.e. no rotation in the xy plane, for objects with affordances such as *pour-to*, *pourable* or *drinkable*, etc. The robot is allowed to observe various activities performed with the objects and it uses our learning algorithms to infer the affordances associated with them. When an instruction is given to the robot, such as 'clear the table' or 'move object x', it uses the inferred affordances to perform the response.

We demonstrate this in two scenarios for the task of 'clearing the table': detecting movable objects and detecting constrained movement. We consider a total of seven activities with nine unique objects. Some objects were used in multiple activities, with a total of 19 object instances. Two of these activities were other high-level activities that were *not seen* during training, but comprise sequences of the learned affordances and sub-activities. The results are summarized in Table 3.7.

Table 3.7: **Robot Object Manipulation Results**, showing the accuracy achieved by the Kodiak PR2 performing the two manipulation tasks, with and without multiple observations.

task	# instance	accuracy (%)	accuracy (%) (multi. obsrv.)
object movement	19	100	100
constrained movement	15	80	100

In the scenario of detecting movable objects, the robot was programmed to move only objects with inferred *movable* affordance, to a specified location. There were a total of 15 instances of movable objects. The robot was able to correctly identify all *movable* objects using our model and could perform the moving task with 100% accuracy.

In the scenario of constrained movement, i.e. the robot should not tilt the objects which contain food items or liquids when moving them. In order to achieve this, we have programmed the robot to perform constrained movement without tilting the objects if it has inferred at least one of the following affordances: {*drinkable, pourable, pour-to*}. The robot was able to correctly identify constraint movement for 80% of the movable instances. Also, if we let the robot observe the activities for a longer time, i.e. let the subject perform multiple activities with the objects and aggregate the affordances associated with the objects before performing the task, the robot is able to perform the task with 100%

accuracy.

These experiments show that robot can use the affordances for manipulating the objects in a more meaningful way. Some affordances such as *moving* are easy to detect, where as some complicated affordances such as *pouring* might need more observations to be detected correctly. Also, by considering other high-level activities in addition to those used for learning, we have also demonstrated the generalizability of our algorithm for affordance detection.

We have made the videos of our results, along with the CAD-120 dataset and code, available at `http://pr.cs.cornell.edu/humanactivities`

3.10 Conclusion

In this Chapter, we have considered the task of jointly labeling human sub-activities and object affordances in order to obtain a descriptive labeling of the activities being performed in the RGB-D videos. The activities we consider happen over a long time period, and comprise several sub-activities performed in a sequence. We formulated this problem as a MRF, and learned the parameters of the model using a structural SVM formulation. Our model also incorporates the temporal segmentation problem by computing multiple segmentations and considering labeling over these segmentations as latent variables.

In extensive experiments over a challenging dataset, we show that our method achieves an accuracy of 79.4% for affordance, 63.4% for sub-activity and 75.0% for high-level activity labeling on the activities performed by a different subject than those in the training set. We also showed that it is important to model the different properties (object affordances, object–object interaction, temporal interactions, etc.) in order to achieve good performance. We also demonstrate the use of our activity and affordance labeling by a PR2 robot in the task of assisting humans with their daily activities. We have shown that being able to infer affordance labels enables the robot to perform the tasks in a more meaningful way.

In this growing area of RGB-D activity recognition, we have presented algorithms for activity and affordance detection and also demonstrated their use in assistive robots, where our robot responds with pre-programmed actions. In this chapter, we have focused on the algorithms for temporal segmentation and labeling while using simple bounding-box detection and tracking algorithms. However, improvements to object perception and task-planning, while taking into consideration the human-robot interaction aspects, are needed for making assistive robots working efficiently alongside humans. We address some of these aspects in the following Chapters.

CHAPTER 4

ANTICIPATING HUMAN ACTIVITIES USING OBJECT AFFORDANCES

4.1 Introduction

For many applications it is important to be able to detect what a human is currently doing as well as *anticipate* what she is going to do next and how. The former ability is useful for applications such as monitoring and surveillance, but we need the latter for applications that require reactive responses, for example, an assistive robot (see Figure 4.1). In this chapter, our goal is to use anticipation for predicting future activities as well as improving detection (of past activities).

There has been a significant amount of work in detecting human activities from 2D RGB videos [189, 159, 152], from inertial/location sensors [132], and from RGB-D videos as discussed in Chapter 3. However, these methods can be used only to predict the labeling of an observed activity and cannot be used to anticipate what can happen next and how.

Our goal is to predict the future activities as well as the details of how a human is going to perform them in short-term (e.g., 1-10 seconds). For example, if a robot has seen a person move his hand to a coffee mug, it is possible he would move the coffee mug to a few potential places such as his mouth, to a kitchen sink or just move it to a different location on the table. If a robot can anticipate this, then it would rather not start pouring milk into the coffee when the person is moving his hand towards the mug, thus avoiding a spill. Such scenarios happen in several other settings, for example, manufacturing scenarios in future co-robotic settings (e.g., [48, 145]).

Activities often have a hierarchical structure where an activity is composed of a sequence of sub-activities and involve interactions with certain objects. For example, a cup is used in the *drinking* activity which is composed of a sequence of *reach*, *move* and *drink* sub-activities. Therefore, we can anticipate the future by observing the sub-activities performed in the past and reasoning about the future based on the structure of activities and the functionality of objects being used (also referred to as object affordances [43]). For example, in Figure 4.1, on seeing a person carrying a bowl and walking towards the refrigerator, one of the most likely future actions are to reach the refrigerator, open it and place the bowl inside.

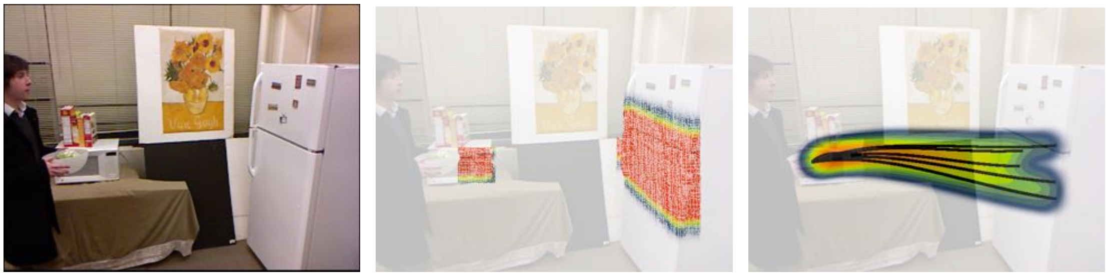
For anticipating the future, we need to predict how the future sub-activities will be performed in terms of motion trajectories of the objects and humans. In order to do this, we propose the use of object affordances and model them in terms of the relative position of the object with respect to the human and the environment[1] and their temporal motions trajectories during the activity, as described in Section 4.4. Modeling trajectories not only helps in discriminating the activities,[2] but is also useful for the robot to reactively plan motions in the workspace.

We use the conditional random field based approach described in Chapter 3 (see Figure 4.2) to model the spatio-temporal structure of activities (see Section 4.5.1). For anticipation, we present an anticipatory temporal conditional random field (ATCRF), where we model the past with the CRF but augmented with the trajectories and with nodes/edges representing the object affordances, sub-

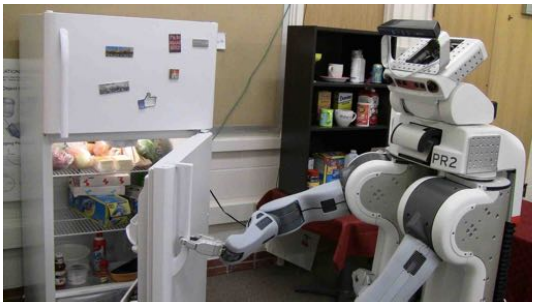
---

[1] For example, a *drinkable* object is found near the mouth of the person performing the *drinking* activity and a *placeable* object is near a stable surface in the environment where it is being placed.

[2] For example, in stirring activity, the target position of the stirrer is immaterial but the circular trajectory motion is.



(a) Robot's RGB-D view. (b) Affordance heatmaps. (c) Trajectories heatmap.



(d) Robot opening the door.

Figure 4.1: **Reactive robot response through anticipation:** Robot observes a person holding an object and walking towards a fridge (a). It uses our ATCRF to anticipate the affordances (b), and trajectories (c). It then performs an anticipatory action of opening the door (d).

activities, and trajectories in the future. Since there are many possible futures, each ATCRF represents only one of them. In order to find the most likely ones, we consider each ATCRF as a particle and propagate them over time, using the set of particles to represent the distribution over the future possible activities. One challenge is to use the discriminative power of the CRFs (where the observations are continuous and labels are discrete) for also producing the generative anticipation—labels over sub-activities, affordances, and spatial trajectories.

We evaluate our anticipation approach extensively on the CAD-120 human activity dataset, which contains 120 RGB-D videos of daily human activities, such as *having meal*, *microwaving food*, *taking medicine*, etc. We first show that anticipation improves the detection of *past* activities: 85.0% with vs 82.3% without. Our algorithm obtains an activity anticipation accuracy (defined as whether one of top three predictions actually happened) of (84.1%,74.4%,62.2%) for predicting (1,3,10) seconds into the future. Videos showing our experiments and *code* are available at: `http://pr.cs.cornell.edu/anticipation`

The main contributions of this chapter are:

- We consider activity *anticipation*, as opposed to activity detection.
- We consider rich contextual relations based on object affordances in RGB-D videos.
- We propose ATCRFs, where each particle represents a CRF.
- We consider joint temporal segmentation and labeling using our approach.
- We test our method on a dataset containing long-term activities, and also test in robotic experiments.

The rest of the chapter is organized as follows. We start with a review of the related work in Section 4.2 and provide an overview of our methodology in Section 4.3. We describe the affordances in Section 4.4 and model in Section 4.5. Section 4.6 describes the features and Section 4.7 describes the learning and inference algorithms. We present the experimental results in Section 4.8.

4.2 Related Work

Activity Detection. In recent years, much effort has been made to detect human activities from still images as well as videos (e.g., [127, 206]). Many meth-

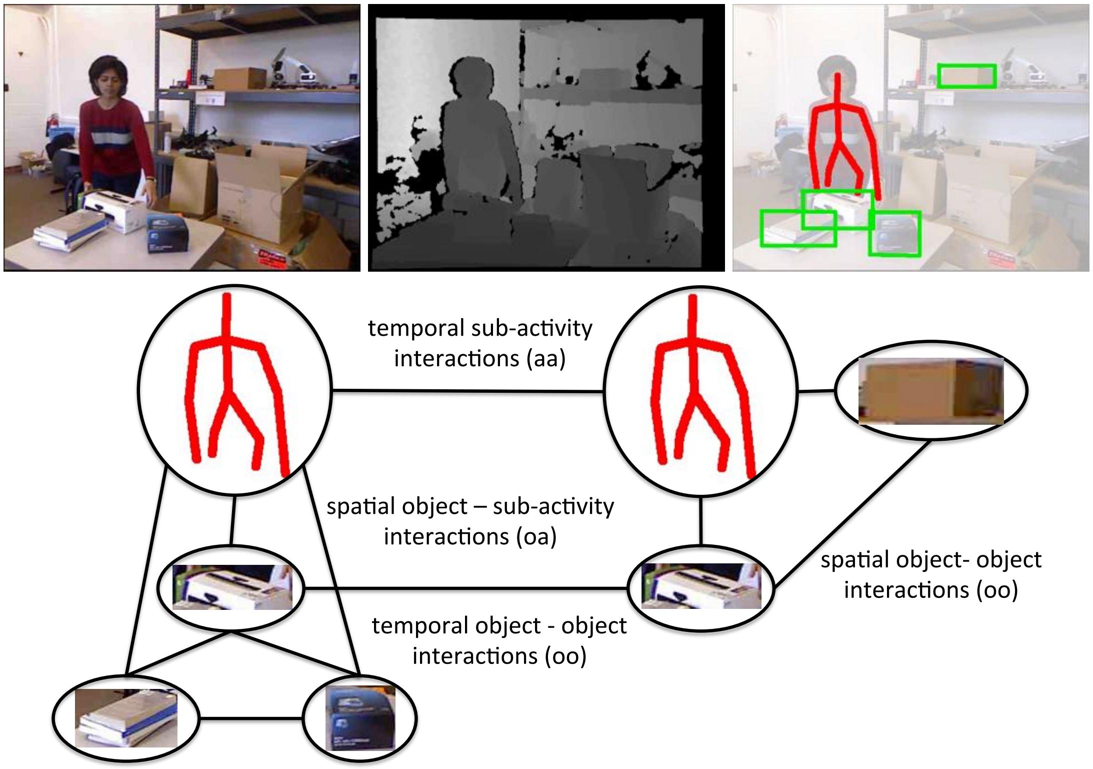

temporal sub-activity interactions (aa)
spatial object – sub-activity interactions (oa)
spatial object- object interactions (oo)
temporal object - object interactions (oo)

Figure 4.2: An example activity from the CAD-120 dataset (top row) and one possible graph structure (bottom row). Top row shows the RGB image (left), depth image (middle), and the extracted skeleton and object information (right). (Graph in the bottom row shows the nodes at only the temporal segment level, the frame level nodes are not shown.)

ods have been proposed to model the temporal structure of low-level features extracted from video, e.g., histograms of spatiotemporal filter responses. This includes both discriminative [107, 142, 40, 189] and generative models [110, 15]. Another approach is to represent activities as collections of semantic attributes [119, 167, 160, 39]. These methods use an intermediate level of representation such as the presence or absence of semantic concepts (e.g., scene types, actions, objects, etc.) in order to generalize to unseen instances. There are also a few recent works which address the task of early recognition [166, 62]. We refer the reader to [2] for a comprehensive survey of the field and discuss works that are closely related to ours.

Some works use human pose for action recognition by detecting local pose features [210] and modeling the spatial configuration between human body parts and objects [49, 211, 75, 194]. Recent availability of inexpensive RGB-D sensors has enabled significant improvement in scene modeling [96, 70] and estimation of human poses [179, 122]. This, together with depth information, has enabled some recent works [214, 141, 185] to obtain good action recognition performance. However, these methods only address detection over small time periods, where temporal segmentation (i.e., knowledge of the spatio-temporal structure) is not an issue.

The model we proposed in Chapter 3 jointly predicts sub-activities and object affordances by taking into account spatio-temporal interactions between human poses and objects over longer time periods. However, we found that not knowing the graph structure (i.e., the correct temporal segmentation) decreased the performance significantly. This is because the boundary between two sub-activities is often not very clear, as people often start performing the next sub-activity before finishing the current sub-activity. Moreover, all these work only detect the activities and affordance after the action is performed. None of these methods can *anticipate* what is going to happen next.

Temporal Segmentation. In activity detection from 2D videos, much previous work has focussed on short video clips, assuming that temporal segmentation has been done apriori. It has been observed that temporal boundaries of actions are not precisely defined in practice, whether they are obtained automatically using weak-supervision [31] or by hand [169]. These works represent the action clips by an orderless bag-of-features and try to improve classification of the action clips by refining their temporal boundaries. However, they only model the temporal extent of actions, not their temporal structure.

Some recent effort in recognizing actions from longer video sequences take an event detection approach [84, 140, 180, 40], where they evaluate a classifier function at many different segments of the video and then predict event presence. Similarly, change point detection methods [209, 53] perform a sequence of change-point analysis in a sliding window along the time dimension. However, these methods only detect *local* boundaries and tend to over-segment complex actions which often contain many changes in local motion statistics.

Some previous works consider joint segmentation and recognition by defining dynamical models based on kinematics [146, 37], but these works do not model the complex human-object interactions. [64] and [63] perform activity classification and clustering respectively but do not consider temporal context. In contrast, our application requires modeling of the temporal context (as compared to just spatial).

In contemporary work, [215] performs joint segmentation and labeling using a structural model which takes into account the activity durations, motion features and context features. They use a nonlinear dynamical model to obtain action segments which are then merged into activities of interest. Wang et al. [202] predict sports moves and human activity in TV episodes by solving a bilinear program to jointly estimate the structure of an MRF graph and perform MAP inference. However, these works do not anticipate the future activities.

Anticipation of Human Actions. Anticipation or forecasting future human actions has been the focus of few recent works. Maximum entropy inverse reinforcement learning was used by [216, 86, 100] to obtain a distribution over possible human navigation trajectories from visual data, and also used to model the forthcoming interactions with pedestrians for mobile robots [216, 100]. However, these works focus only on human actions which are limited to navigation trajectories. Wang et al. [201] propose a latent variable model for inferring unknown human intentions, such as the target ball position in a robot table tennis scenario, and Dragan et al. [30] use inverse reinforcement learning to predict future goal for grasping an object. In comparison, we address the problem of anticipation of human actions at a fine-grained level of how a human interacts with objects in more involved activities such as *microwaving food* or *taking medicine* compared to the generic navigation activities or task-specific trajectories.

Learning Algorithms. Our work uses probabilistic graphical models to capture rich context. Such frameworks as HMMs [67, 139], DBNs [45], CRFs [155, 182], semi-CRFs [168] have been previously used to model the temporal structure of videos and text. While these previous works maintain their template graph structure over time, in our work, new graph structures are possible. Works on semi-Markov models [168, 177] are quite related as they address the problem of finding the segmentation along with labeling. However, these methods are limited since they are only efficient for feature maps that are additive in nature. We build upon these ideas where we use additive feature maps only as a close approximation to the graph structure and then explore the space of likely graph structure by designing moves. We show that this improves performance while being computationally efficient.

For anticipation, we use importance sampling for efficient estimation of the likelihood of the potential future activities. Particle filters have been applied with great success to a variety of state estimation problems including object tracking [85, 61], mobile robot localization [36, 69], people tracking [174], etc. However, the worst-case complexity of these methods grows exponentially in the dimensions of the state space, it is not clear how particle filters can be applied to arbitrary, high-dimensional estimation problems. Some approaches use factorizations of the state space and apply different representations for the individual parts of the state space model. For example, Rao-Blackwellised particle filters sample only the discrete and non-linear parts of a state estimation problem. The remaining parts of the states are solved analytically conditioned on the particles by using Kalman filters [28, 51, 134, 174]. In our work, each of our particles is a CRF that models rich structure and lies in a high-dimensional space.

4.3 Overview

Our goal is to anticipate what a human will do next given the current observation of his pose and the surrounding environment. These observations are from RGB-D videos recorded with a Kinect sensor. From these videos, we obtain the human pose using the Openni's skeleton tracker [1] and extract the tracked object point clouds using SIFT feature matching as described in Chapter 3. Note that we infer the object affordances based on its usage in the activity and do not require the object category labels. We discuss the effect of knowing the object categories on the anticipation performance in Section 4.5.3.

Since activities happen over a long time horizon, with each activity being composed of sub-activities involving different number of objects, we perform segmentation in time, as described in Section 4.7.1, such that each temporal segment represents one sub-activity. We model the activity using a spatio-temporal graph (a CRF), as shown in Figure 4.3-left. The extracted human pose and objects form the nodes in this graph, and the edges between them represent their interactions, as described in Section 4.5.1. However, this graph can only model the present observations. In order to predict the future, we augment the graph with an 'anticipated' temporal segment, with anticipated nodes for sub-activities, objects (their affordances), and the corresponding spatio-temporal trajectories. We call this augmented graph an anticipatory temporal CRF (ATCRF), formally defined in Section 4.5.2.

Our goal is to obtain a distribution over the future possibilities, i.e., a distribution over possible ATCRFs. Motivated by particle filtering algorithm [135], we represent this distribution as a set of weighted particles, where each particle is a sampled ATCRF. Partial observations become available as the sub-activity is being performed and we use these partial observations to improve the estimation of the distribution. Section 4.5.3 describes this approach. Since each of our ATCRF captures strong context over time (which sub-activity follows another) and space (spatial motion of humans and objects, and their interactions), each of our particles (i.e., possible future) is rich in its modeling capacity. Later, our experiments in Section 4.8 will show that this is essential for anticipating human actions.

Anticipated temporal segments are generated based on the available object affordances and the current configuration of the 3D scene. For example, if a person has picked up a coffee mug, one possible outcome could be drinking from

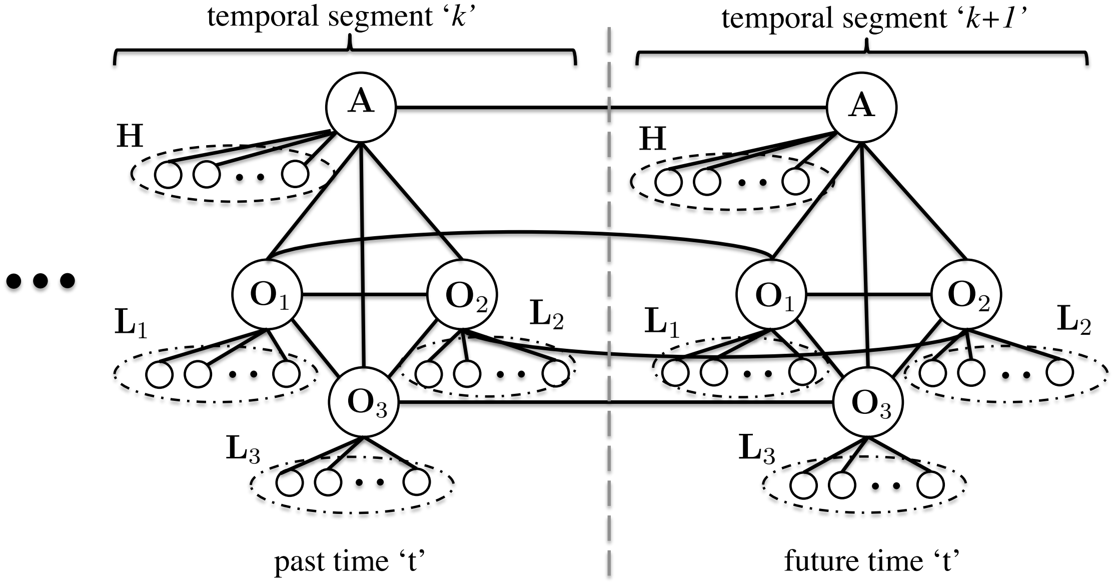


Figure 4.3: An ATCRF that models the human poses $\mathcal{H}$, object affordance labels $\mathcal{O}$, object locations $\mathcal{L}$, and sub-activity labels $\mathcal{A}$, over past time 't', and future time 'd'. Two temporal segments are shown in this figure: k^{th} for the recent past, and $(k+1)^{th}$ for the future. Each temporal segment has three objects for illustration in the figure.

it. Therefore, for each object, we sample possible locations at the end of the anticipated sub-activity and several trajectories based on the selected affordance. The location and trajectory generation are described in Section 4.4.1 and Section 4.4.2 respectively.

The temporal segmentation determines the structure of the ATCRF. It is quite challenging to estimate this structure because of two reasons. First, an activity comprises several sub-activities of varying temporal length, with an ambiguity in the temporal boundaries. Thus a single graph structure may not explain the activity well. Second, there can be several possible graph structures when we are reasoning about activities in the future (i.e., when the goal is to *anticipate* future activities, different from just detecting the past activities). Multiple spatio-temporal graphs are possible in these cases and we need to reason over them in our learning algorithm.

Figure 4.4 shows two possible graph structures for an activity with two objects. We reason about the possible graph structures for both past and future activities. The key idea is to first sample a few segmentations that are close to the ground-truth using our CRF model instantiated with a subset of features, and then explore the space of segmentation by making merge and split moves to create new segmentations. We do so by approximating the graph with only additive features, which lends to efficient dynamic programming, as described in Section 4.7.1.

4.4 Object Affordances

The concept of affordances was proposed by Gibson [43] as all "action possibilities" provided by the environment. Many recent works in computer vision and robotics reason about object functionality (e.g., sittable, drinkable, etc.) instead of object identities (e.g., chairs, mugs, etc.). These works take a recognition based approach to identify the semantic affordance labels [87, 184, 59]. Few recent works explore the physical aspects of affordances based on human interactions [47, 50, 22, 77]. For example, Grabner et al. [47] detect the functionality of the object (specifically, chairs) with respect to possible human poses, Gupta et al. [50] predict stable and feasible human poses given an approximate 3D geometry from an image, etc. In our work, we consider semantic affordances with spatio-temporal grounding which help in anticipating the future activities. Here, we describe how we model the spatio-temporal aspects of affordances.

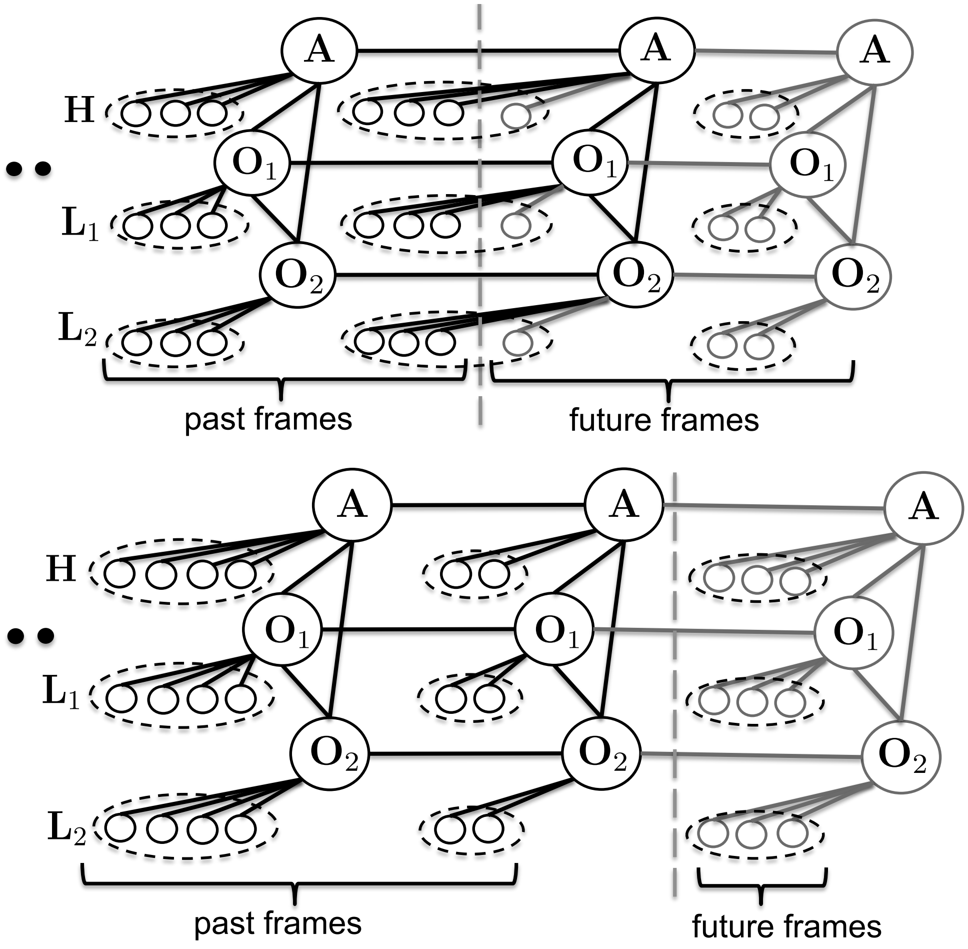


Figure 4.4: Figure illustrating two possible graph structures resulting from two temporal segmentations (top and bottom), with six observed frames in the past and three anticipated frames in the future. This example has one sub-activity node and two object nodes in each temporal segment.

4.4.1 Object Affordance Heatmaps

To represent object affordances we define a potential function based on how the object is being interacted with, when the corresponding affordance is active. The kind of interaction we consider depends on the affordance being considered. For example, when the active affordance of an object is *drinkable*, the object is found near the human's mouth, the interaction considered is the relative po-

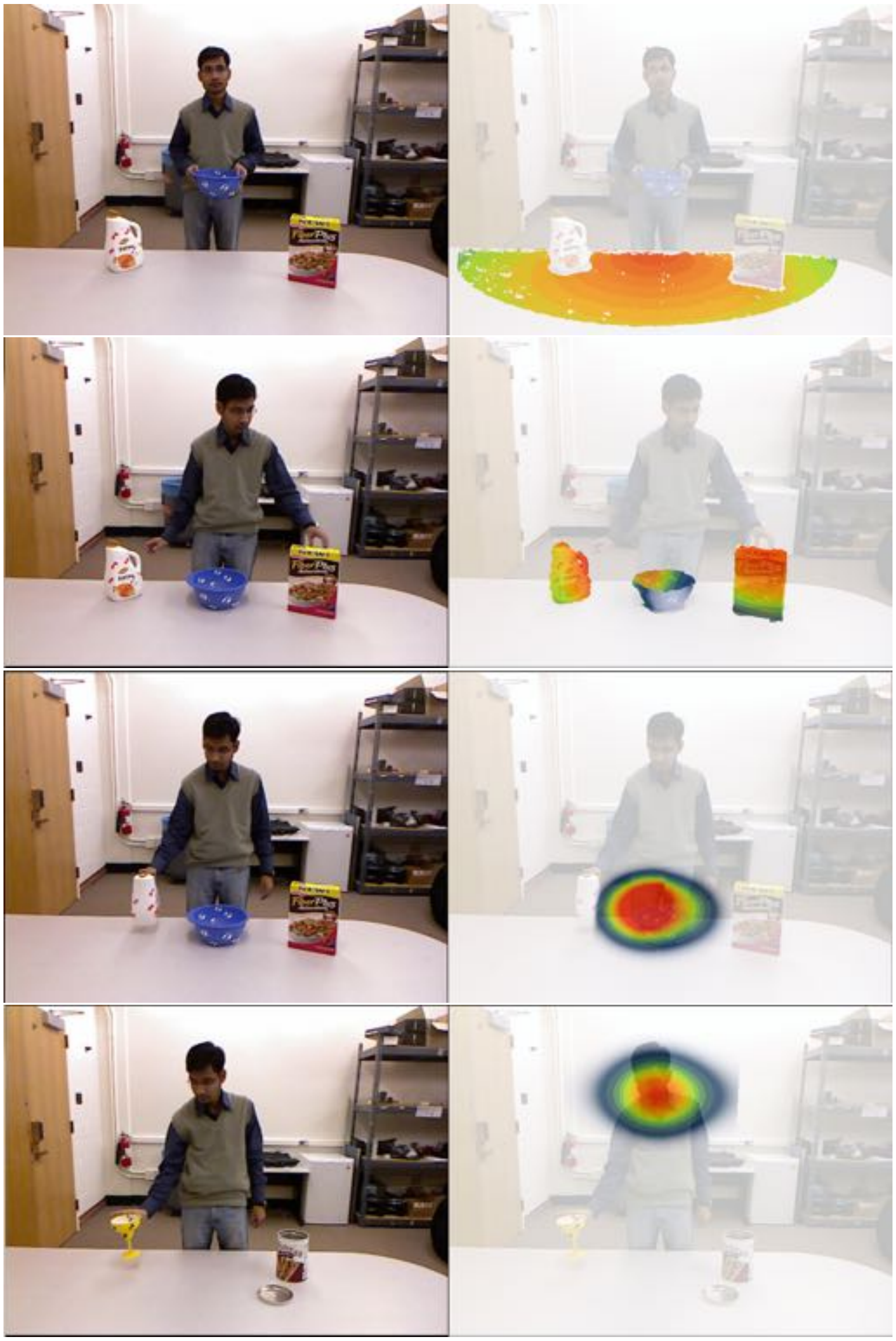


Figure 4.5: **Affordance heatmaps**. The learnt affordance heatmaps for *placeability* (top-left), *reachability* (top-right), *pourability* (bottom-left) and *drinkability* (bottom-right). The red signifies where the affordance is most likely, for example, the red signifies where the object is *placeable* (top-left) and the most likely *reachable* locations on the object (top-right) (See Section 4.4.1).

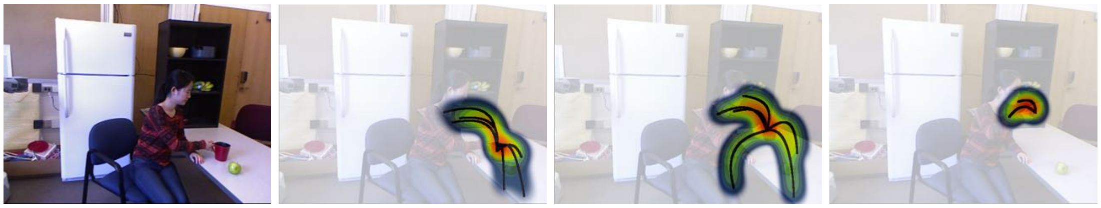


Figure 4.6: **Heatmap of anticipated trajectories** for *moving* sub-activity and how they evolve with time.

sition of the object with respect to the human skeleton. In case of the affordance *placeable*, the interaction is the relative position of the object with respect to the environment, i.e., an object is *placeable* when it is above a surface that provides stability to the object once placed. The general form of the potential function for object affordance o given the observations at time t is:

$$\psi_o = \prod_i \psi_{dist_i} \prod_j \psi_{ori_j} \tag{4.1}$$

where ψ_{dist_i} is the i^{th} distance potential and ψ_{ori_j} is the j^{th} relative angular potential. We model each distance potential with a Gaussian distribution and each relative angular potential with a von Mises distribution. We find the parameters of the affordance potential functions from the training data using maximum likelihood estimation. Since the potential function is a product of the various components, the parameters of each distribution can be estimated separately. In detail, the mean and variance of the Gaussian distribution have closed form solutions, and we numerically estimate the mean and concentration parameter of the von Mises distribution.

We categorize these functions into three groups depending on the potentials used: (1) affordances *drinkable* and *reachable* have one distance potential per skeleton joint and one angular potential with respect to the head orientation, (2) affordances depending on the target object, such as *pourable* which depends on a *pour-to* object, have a distance potential and an angular potential with respect to the target object's location, (3) the rest of the affordances which depend on the environment, such *placeable* and *openable*, have a distance potential with respect to the closest surface and an angular potential with respect to the head orientation.

We generate heatmaps for each affordance by scoring the points in the 3D space using the potential function, and the value represents the strength of the particular affordance at that location. Figure 4.5 shows the heatmaps generated for the *placeable*, *reachable*, *pourable* and *drinkable* affordances. We obtain the future target locations of an object by weighted sampling of the scored 3D points.

4.4.2 Trajectory Generation

Once a location is sampled from the affordance heatmap, we generate a set of possible trajectories in which the object can be moved form its current location to the predicted target location. We use parametrized cubic equations, in particular Bézier curves, to generate human hand like motions [32].

$$\begin{aligned} B(x) = {} & (1-x)^3 L_0 \; + \; 3(1-x)^2 x L_1 \\ & + \; 3(1-x)x^2 L_2 \; + \; x^3 L_3 \,, \quad x \in [0,1] \end{aligned} \tag{4.2}$$

We estimate the control points of the Bézier curves for the proposal distribution component from the trajectories in the training data. A cubic Bézier curve, as shown in Eq. 4.2, is parameterized by a set of four points: the start and end point of the trajectory (L_0 and L_3 respectively), and two control points (L_1 and L_2) which define the shape of the curve. We first transform and normalize the trajectories in the training data so that all of them have the same start and end points. We then estimate the control points of the Bézier curve, one per sub-activity class, which best fit the normalized trajectories. In detail, L_0 and L_3 are the start and end points of the normalized trajectories, respectively, and L_1 and L_2 are estimated using the least square fitting method to minimize the distance between the fitted and the observed normalized trajectories. Figure 4.6 shows some of the anticipated trajectories for *moving* sub-activity.

4.5 Our Approach

Given the observations of a scene containing a human and objects for time t in the past, and its goal is to anticipate future possibilities for time d.

However, for the future d frames, we do not even know the structure of the graph—there may be different number of objects being interacted with depending on which sub-activity is performed in the future. Our goal is to compute a distribution over the possible future states (i.e., sub-activity, human poses and object locations). We will do so by sampling several possible graph structures by augmenting the graph in time, each of which we will call an anticipatory temporal conditional random field (ATCRF). We first describe an ATCRF below.

4.5.1 Modeling Past with a CRF

Following Chapter 3, we discretize time to the frames of the video[3] and group the frames into temporal segments, where each temporal segment spans a set of contiguous frames corresponding to a single sub-activity. Therefore, at time 't' we have observed 't' frames of the activity that are grouped into 'k' temporal segments. For the past t frames, we know the nodes of the CRF but we do not know the temporal segmentation, i.e., which frame level nodes are connected to each of the segment level node. The node labels are also unknown. For a given temporal segmentation, we represent the graph until time t as: $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$, where $\mathcal{E}^t$ represents the edges, and $\mathcal{V}^t$ represents the nodes $\{\mathcal{H}^t, \mathcal{O}^t, \mathcal{L}^t, \mathcal{A}^t\}$: human pose nodes $\mathcal{H}^t$, object affordance nodes $\mathcal{O}^t$, object location nodes $\mathcal{L}^t$, and sub-activity nodes $\mathcal{A}^t$. Figure 4.3-left part shows the structure of this CRF for an activity with three objects.

Our goal is to model the $P(\mathcal{H}^t, \mathcal{O}^t, \mathcal{L}^t, \mathcal{A}^t \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}})$, where $\Phi^t_{\mathcal{H}}$ and $\Phi^t_{\mathcal{L}}$ are the observations for the human poses and object locations until time t. Using the independencies expressed over the graph in Figure 4.3, for a graph $\mathcal{G}^t$, we have:

$$P_{\mathcal{G}^t}(\mathcal{H}^t, \mathcal{O}^t, \mathcal{L}^t, \mathcal{A}^t \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) = P(\mathcal{O}^t, \mathcal{A}^t \mid \mathcal{H}^t, \mathcal{L}^t)\; P(\mathcal{H}^t, \mathcal{L}^t \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) \tag{4.3}$$

The second term $P(\mathcal{H}^t, \mathcal{L}^t \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}})$ models the distribution of true human pose and object locations (both are continuous trajectories) given the observations from the RGB-D Kinect sensor. We model it using a Gaussian distribution. The first term $P(\mathcal{O}^t, \mathcal{A}^t \mid \mathcal{H}^t, \mathcal{L}^t)$ predicts the object affordances and the sub-activities that are discrete labels—this term further factorizes following the

---

[3] In the following, we will use the number of videos frames as a unit of time, where 1 unit of time ≈ 71ms (=1/14, for a frame-rate of about 14Hz in our experiments).

graph structure as:

$$P(\mathcal{O}^t, \mathcal{A}^t \mid \mathcal{H}^t, \mathcal{L}^t) \propto \overbrace{\prod_{o_i \in \mathcal{O}} \Psi_{\mathcal{O}}(o_i | \ell_{o_i})}^{\text{object affordance}} \overbrace{\prod_{a_i \in \mathcal{A}} \Psi_{\mathcal{A}}(a_i | h_{a_i})}^{\text{sub-activity}} \prod_{v_i, v_j \in \mathcal{E}} \overbrace{\Psi_{\mathcal{E}}(v_i, v_j | \cdot)}^{\text{edge terms}} \tag{4.4}$$

Given the continuous state space of $\mathcal{H}$ and $\mathcal{L}$, we rely on the method described in Chapter 3 for powerful modeling using a discriminative framework for the above term. Each node potential function in Eq. (4.4), $\Psi_{\mathcal{O}}(o_i|\ell_{o_i})$ and $\Psi_{\mathcal{A}}(a_i|h_{a_i})$, has the form $\sum_{k \in K} y_i^k \left[w_n^k \cdot \phi_n(i) \right]$, where y_i^k denotes a binary variable representing the node i having label k, K is the set of labels, $\phi_n(i)$ is the node feature map and w_n^k are the corresponding node feature weights. Similarly, each edge potential function, $\Psi_{\mathcal{E}}(v_i, v_j|\cdot)$, has the form $\sum_{(l,k) \in K \times K} y_i^l y_j^k \left[\mathbf{w}_\mathbf{e}^{lk} \cdot \phi_e(i, j) \right]$, where $\phi_e(i, j)$ is the edge feature map and $\mathbf{w}_\mathbf{e}^{lk}$ are the corresponding edge feature weights. The feature maps, $\phi_n(i)$ and $\phi_e(i, j)$, are described in detail in Section 4.6. We can rewrite Eq. (4.4) by taking logarithm on both sides and grouping the node potentials as the following energy function expressed over a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (as illustrated in Figure 4.2):

$$E_{\mathbf{w}}(\mathbf{y}, \Phi(\mathbf{x})) = \sum_{i \in \mathcal{V}} \sum_{k \in K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] + \sum_{(i,j) \in \mathcal{E}} \sum_{(l,k) \in K \times K} y_i^l y_j^k \left[\mathbf{w}_\mathbf{e}^{lk} \cdot \phi_e(i, j) \right] \tag{4.5}$$

Depending on the nodes and edges in the graph $\mathcal{G}$, the appropriate subset of features and class labels are used in energy function given in Eq. (4.5). The learning and inference algorithms are described in Section 4.7.

4.5.2 Modeling one Possible Future with an augmented temporal CRF (ATCRF).

We defined the anticipatory temporal conditional random field as an augmented graph $\mathcal{G}^{t,d} = (\mathcal{V}^{t,d}, \mathcal{E}^{t,d})$, where t is observed time and d is the future anticipation time. $\mathcal{V}^{t,d} = \{\mathcal{H}^{t,d}, \mathcal{O}^{t,d}, \mathcal{L}^{t,d}, \mathcal{A}^{t,d}\}$ represents the set of nodes in the past time t as well as in the future time d. $\mathcal{E}^{t,d}$ represents the set of all edges in the graph (see Figure 4.3). The observations (not shown in the figure) are represented as set of features, $\Phi^t_{\mathcal{H}}$ and $\Phi^t_{\mathcal{O}}$, extracted from the t observed video frames. Note that we do not have observations for the future frames.

In the augmented graph $\mathcal{G}^{t,d}$, we have:

$$P_{\mathcal{G}^{t,d}}(\mathcal{H}^{t,d}, \mathcal{O}^{t,d}, \mathcal{L}^{t,d}, \mathcal{A}^{t,d} \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) = P(\mathcal{O}^{t,d}, \mathcal{A}^{t,d} \mid \mathcal{H}^{t,d}, \mathcal{L}^{t,d})\, P(\mathcal{H}^{t,d}, \mathcal{L}^{t,d} \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) \tag{4.6}$$

The first term is similar to Eq. (4.4), except over the augmented graph, and we can still rely on the discriminatively trained CRF presented in Chapter 3. We model the second term with a Gaussian distribution.

4.5.3 Modeling the Distribution over Future Possibilities with ATCRFs.

There can be several potential augmented graph structures $\mathcal{G}^{t,d}$ because of different possibilities in human pose configurations and object locations that determines the neighborhood graph. Even the number of nodes to be considered

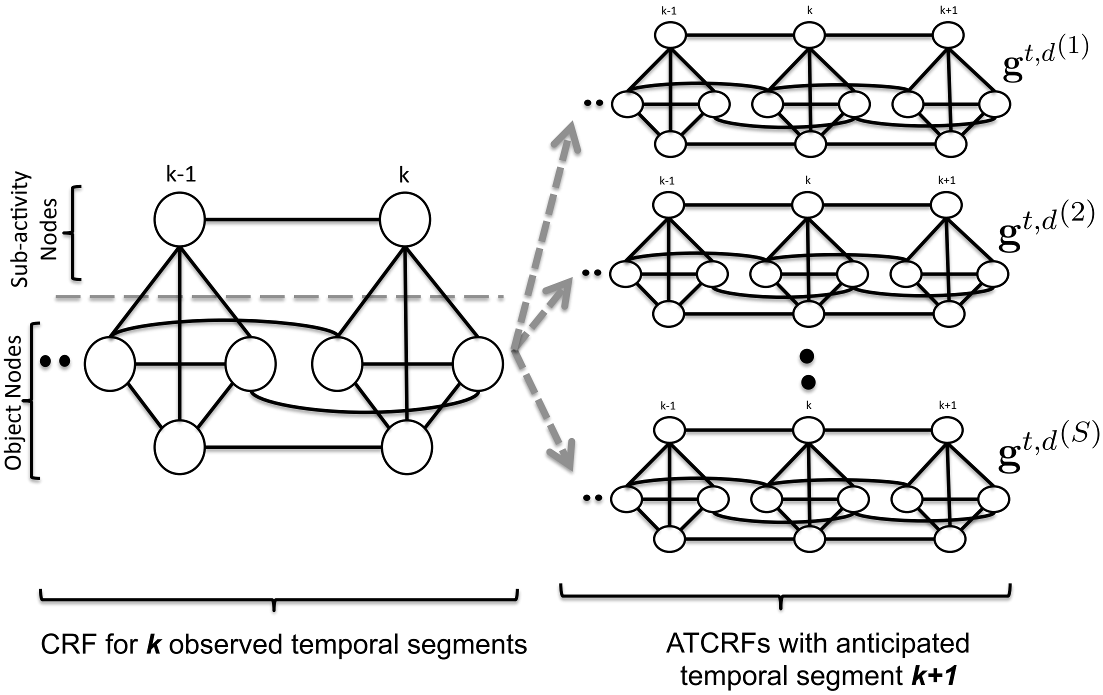


Figure 4.7: Figure showing the process of augmenting the CRF structure to obtain multiple ATCRFs at time *t* for an activity with three objects. The frame level nodes are not shown for the sake of clarity.

in the future changes depending on the sub-activity and the configuration of the environment.

Let $\mathbf{g}^{t,d}$ represent a sample augmented graph structure with particular values assigned to its node variables. I.e., one sample may represent that a person and object move in a certain way, performing a sub-activity with certain object affordances, and another sample may represent a person moving in a different way performing a different sub-activity.

Figure 4.7 shows the process of augmenting CRF structure corresponding to the seen frames with the sampled anticipations of the future to produce multiple ATCRF particles at time *t*. The frame level nodes are not shown in the figure.

The left portion of the figure shows the nodes corresponding to the k observed temporal segments. This graph is then augmented with a set of anticipated nodes for the temporal segment $k+1$, to generate the ATCRF particles at time t. The frame level nodes of $k+1$ temporal segment are instantiated with anticipated human poses and object locations.

The goal is now to compute the distribution over these ATCRFs $\mathbf{g}^{t,d}$, i.e., given observations until time t, we would like to estimate the posterior distribution $p(\mathbf{g}^{t,d}|\Phi_t)$ from Eq. (4.6). However, this is extremely challenging because the space of ATCRFs is infinite, and to even represent the distribution we need an exponential number of labels. We therefore represent the posterior using a set of weighted particles as shown in Eq. (4.7) and choose the weights using importance sampling as shown in Eq. (4.8).

$$p(\mathbf{g}^{t,d}|\Phi_t) \approx \sum_{s=1}^{S} \hat{w}_t^s \delta_{\mathbf{g}^{t,d(s)}}(\mathbf{g}^{t,d}) \tag{4.7}$$

$$\hat{w}_t^s \propto \frac{p(\mathbf{g}^{t,d^{(s)}}|\Phi_t)}{q(\mathbf{g}^{t,d^{(s)}}|\Phi_t)} \tag{4.8}$$

Here, $\delta_x(y)$ is the Kronecker delta function which takes the value 1 if x equals y and 0 otherwise, $\hat{w}_t^s$ is the weight of the sample s after observing t frames, and $q(\mathbf{g}^{t,d}|\Phi_t)$ is the proposal distribution. We need to perform importance sampling because: (a) sampling directly from $p(\mathbf{g}^{t,d}|\Phi_t)$ is not possible because of the form of the distribution in a discriminative framework, and (b) sampling uniformly would be quite naive because of the large space of ATCRFs and most of our samples would entirely miss the likely futures.

We now describe how we sample particles from the proposal distribution $q(\mathbf{g}^{t,d}|\Phi_t)$ and how to evaluate the posterior for the generated samples.

Sampling. In order to generate a particle ATCRF, we need to generate possible human pose and object locations for the d future frames. We write the desired distribution to sample as:

$$\begin{aligned} q(\mathbf{g}^{t,d}|\Phi^t) =& P_{\mathcal{G}^{t,d}}(\mathcal{H}^{t,d}, \mathcal{O}^{t,d}, \mathcal{L}^{t,d}, \mathcal{A}^{t,d} \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) \\ =& P_{\mathcal{G}^t}(\mathcal{H}^t, \mathcal{O}^t, \mathcal{L}^t, \mathcal{A}^t \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) \\ & P(\mathcal{H}^d, \mathcal{L}^d \mid \mathcal{O}^d, \mathcal{A}^d, \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}})\ P(\mathcal{O}^d, \mathcal{A}^d \mid \mathcal{O}^t, \mathcal{A}^t, \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}}) \end{aligned} \tag{4.9}$$

We first sample the affordances, one per object in the scene, and the corresponding sub-activity from the distribution $P(\mathcal{O}^d, \mathcal{A}^d \mid \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}})$. This is a discrete distribution generated from the training data based on the object type (e.g., cup, bowl, etc.) and object's current position with respect to the human in the scene (i.e., in contact with the hand or not). For example, if a human is holding an object of type 'cup' placed on a table, then the affordances *drinkable* and *movable* with their corresponding sub-activities (*drinking* and *moving* respectively) have equal probability, with all others being 0.[4]

Once we have the sampled affordances and sub-activity, we need to sample the corresponding object locations and human poses for the d anticipated frames from the distribution $P(\mathcal{H}^d, \mathcal{L}^d \mid \mathcal{O}^d, \mathcal{A}^d, \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}})$. In order to have meaningful object locations and human poses we take the following approach. We sample a set of target locations and motion trajectory curves based on the sampled affordance, sub-activity and available observations. We then generate the corresponding object locations and human poses from the sampled end point and trajectory curve. The details of sampling the target object location and motion

---

[4] If we know the object type as well, then it helps in generating informative samples based on the discrete distribution learnt from the training data, which can save computation time as fewer samples are sufficient. In our experiments, when object type information is not provided, the anticipation performance (micro P/R) for sub-activities and affordances only drops by a maximum of 0.3% for our full model (Table 4.2-row 6).

trajectory curves are described in Section 4.4.1 and Section 4.4.2 respectively.

Scoring. Once we have the sampled ATCRF particles, we obtain the weight of each sample s by evaluating the posterior for the given sample, $q(\mathbf{g}^{t,d^{(s)}}|\Phi^t)$, as shown in Eq. (4.9) and normalize the weights across the samples.

Note that the aforementioned methods for the affordance and trajectory generation are only for the proposal distribution to sample. The estimated trajectories are finally scored using our ATCRF model.

4.6 Features: Additive and Non-Additive

In this section we describe the feature maps used in the energy function Eq. 4.5. In detail, we use the same features as described in Chapter 3. These features include the node feature maps $\phi_o(i)$ and $\phi_a(j)$ for object node i and sub-activity node j respectively, and edge feature maps $\phi_e(i, j)$ capturing the relations between various nodes. The object node feature map, $\phi_o(i)$, includes the (x, y, z) coordinates of the object's centroid, the coordinates of the object's bounding box and transformation matrix w.r.t. to the previous frame computed at the middle frame of the temporal segment, the total displacement and distance moved by the object's centroid in the set of frames belonging to the temporal segment. The sub-activity node feature map, $\phi_a(j)$, gives a vector of features computed using the noisy human skeleton poses obtained from running Openni's skeleton tracker on the RGBD video. We compute the above described location (relative to the subject's head location) and distance features for each the upper-skeleton joints excluding the elbow joints (neck, torso, left shoulder, left palm, right shoulder and right palm).

The edge feature maps, $\phi_t(i, j)$, include relative geometric features such as the difference in (x, y, z) coordinates of the object centroids and skeleton joint locations and the distance between them. In addition to computing these values at the first, middle and last frames of the temporal segment, we also consider the *min* and *max* of their values across all frames in the temporal segment to capture the relative motion information. The temporal relational features capture the change across temporal segments and we use the vertical change in position and the distance between corresponding object and joint locations. We perform cumulative binning of all the feature values into 10 bins for each feature.

We categorize the features into two sets: additive features, $\Phi^A(\mathbf{x})$, and non-additive features, $\Phi^{NA}(\mathbf{x})$. We compute the additive features for a set of frames corresponding to a temporal segment by adding the feature values for the frames belonging to the temporal segment. Examples of the additive features include distance moved and vertical displacement of an object within a temporal segment. The features that do not satisfy this property are referred to as the non-additive features, for example, maximum and minimum distances between two objects. As we discuss in the next section, additive features allow efficient joint segmentation and labeling by using dynamic programming, but may not be expressive enough.

Non-additive features sometimes provide very useful cues for discriminating the sub-activity and affordance classes. For example, consider discriminating *cleaning* sub-activity from a *moving* sub-activity: here the total distance moved could be similar (an additive feature), however, the minimum and maximum distance moved being small may be strong indicator of the activity being *cleaning*. In fact, when compared to our model learned using only the additive features, the model learned with both additive and non-additive features improves macro precision and recall by 5% and 10.1% for labeling object affordance respectively and by 3.7% and 6.2% for labeling sub-activities respectively.

4.7 Learning and Inference

4.7.1 Inference

The inference problem is to find the best labeling of the past nodes of the CRF for detecting the past sub-activities and affordances and past as well as augmented future nodes of the ATCRF for anticipation. The prediction $\hat{\mathbf{y}}$ is computed as the argmax of an energy function $E_{\mathbf{w}}(\mathbf{y}, \Phi(\mathbf{x}))$.

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} E_{\mathbf{w}}(\mathbf{y}, \Phi(\mathbf{x})) \tag{4.10}$$

For a given temporal segmentation, where the graph structure is fully known, finding the argmax over labelings is a NP hard problem. However, its equivalent formulation as the following mixed-integer program has a linear relaxation which can be solved efficiently as a quadratic pseudo-Boolean optimization problem using a graph-cut method [162] as described in Chapter 3.

$$\hat{\mathbf{y}} = \operatorname*{argmax}_{\mathbf{y}} \max_{\mathbf{z}} \sum_{i \in \mathcal{V}} \sum_{k \in K} y_i^k \left[w_n^k \cdot \phi_n(i) \right] + \sum_{(i,j) \in \mathcal{E}} \sum_{(l,k) \in K \times K} z_{ij}^{lk} \left[\mathbf{w}_{\mathbf{e}}^{lk} \cdot \phi_e(i,j) \right] \tag{4.11}$$

$$\forall i, j, l, k:\ z_{ij}^{lk} \le y_i^l,\ z_{ij}^{lk} \le y_j^k,\ y_i^l + y_j^k \le z_{ij}^{lk} + 1,\ z_{ij}^{lk}, y_i^l \in \{0, 1\} \tag{4.12}$$

Note that the products $y_i^l y_j^k$ have been replaced by auxiliary variables z_{ij}^{lk}. Relaxing the variables z_{ij}^{lk} and y_i^l to the interval $[0, 1]$ results in a linear program that can be shown to always have half-integral solutions (i.e. y_i^l only take values $\{0, 0.5, 1\}$ at the solution) [52]. When we consider the additional constraints that each node can take only one label, the problem can no longer be solved via graph cuts. We compute the exact mixed integer solution including these additional constraints using a general-purpose MIP solver[5] during inference.

However, during inference we only know the nodes in the graph but not the temporal segmentation, i.e., the structure of the graph in terms of the edges connecting frame level nodes to the segment level label nodes. We could search for the best labeling over *all* possible segmentations, but this is very intractable because our feature maps contain non-additive features (that are important as described in Section 4.6).

Efficient Inference with Additive Features. We express the feature set, $\Phi(\mathbf{x})$, as the concatenation of the additive and non-additive feature sets, $\Phi^A(\mathbf{x})$ and $\Phi^{NA}(\mathbf{x})$ respectively. Therefore, by rearranging the terms in Eq. (4.4), the energy function can be written as:

$$E_{\mathbf{w}}(\mathbf{y}, \Phi(\mathbf{x})) = E_{\mathbf{w}}(\mathbf{y}, \Phi^A(\mathbf{x})) + E_{\mathbf{w}}(\mathbf{y}, \Phi^{NA}(\mathbf{x}))$$

We perform efficient inference for the energy term $E_{\mathbf{w}}(\mathbf{y}, \Phi^A(\mathbf{x}))$ by formulating it as a dynamic program (see Eq. (4.13)). In detail, let L denote the max length of a temporal segment, i denote the frame index, s denote the temporal segment spanning frames $(i - l)$ to i, and $(s - 1)$ denote the previous segment. We write the energy function in a recursive form as:

$$V(i, k) = \max_{k', l=1\ldots L} V(i - l, k') + \sum_{k \in K} y_s^k \left[w_n^k \cdot \phi_n^A(s) \right] + \sum_{k \in K} y_s^k \left[\mathbf{w}_{\mathbf{e}}^{lk} \cdot \phi_e^A(s-1, s) \right] \tag{4.13}$$

Here, $\phi_n^A(s)$ and $\phi_e^A(s-1, s)$ denote the additive feature maps and can be effi-

---

[5] http://www.tfinley.net/software/pyglpk/readme.html

ciently computed by using the concept of integral images.[6] The best segmentation then corresponds to the path traced by $\max_a V(t, a)$, where t is the number of video frames. Using $E_{\mathbf{w}}(\mathbf{y}, \Phi^A(\mathbf{x}))$, we find the top-k scored segmentations[7] and then evaluate them using the full model $E_{\mathbf{w}}(\mathbf{y}, \Phi(\mathbf{x}))$ in order to obtain more accurate labelings.

Merge and Split Moves. The segmentations generated by the approximate energy function, $E_{\mathbf{w}}(\mathbf{y}, \Phi^A(\mathbf{x}))$, are often very close to the given ground-truth segmentations. However, since the energy function used is only approximate, it sometimes tends to over-segment or miss the boundary by a few frames. In order to obtain a representative set of segmentation samples, we also perform random merge and split moves over these segmentations, and consider them for evaluating with the full model as well. A merge move randomly selects a boundary and removes it, and a split move randomly chooses a frame in a segment and creates a boundary.

Heuristic Segmentations. There is a lot of information present in the video which can be utilized for the purpose of temporal segmentation. For example, smooth movement of the skeleton joints usually represent a single sub-activity and the sudden changes in the direction or speed of motion indicate sub-activity boundaries. Therefore, we incorporate such information in performing temporal segmentation of the activities. In detail, we use the multiple segmentation hypotheses proposed in Chapter 3. These include graph based segmentation method proposed by [34] adapted to temporally segment the videos. The sum

---

[6] The additive features for temporal segments starting at the first frame and ending at frame l, for $l = 1..t$ are precomputed, i.e., the segment features for a total of t temporal segments are computed. This needs $(n \times t)$ summations, where n is the number of features. Now the segment features for a temporal temporal segment starting and ending at any frame can be computed by n subtractions. Therefore, the total feature computation cost is linear in the number of possible segmentations.

[7] k is 2 in experiments.

of the Euclidean distances between the skeleton joints and the rate of change of the Euclidean distance are used as the edge weights for two heuristic segmentations respectively. By varying the thresholds, different temporal segmentations of the given activity can be obtained. In addition to the graph based segmentation methods, we also use the uniform segmentation method which considers a set of continuous frames of fixed size as the temporal segment. There are two parameters for this method: the segment size and the offset (the size of the first segment). However, these methods often over-segment a sub-activity, and each segmentation would result in a different graph structure for our CRF modeling.

We generate multiple graph structures for various values of the parameters for the above mentioned methods and obtain the predicted labels for each using Eq. (4.10). We obtain the final labeling over the segments by either using the second-step learning method presented in Chapter 3, or by performing voting and taking the label predicted by majority of the sampled graph structures (our experiments in Section 4.8.2 follow the latter). During anticipation, we only consider the best graph structure obtained form the using the additive features and split-merge moves. Algorithm 1 gives the summary of the inference algorithm for anticipation.

4.7.2 Learning

The structure of the graph is fully known during learning. We obtain the parameters of the energy function in Eq. (4.5) by using the cutting plane method [81] as described in Chapter 3. Given M labeled training examples $(\mathbf{x}_1, \mathbf{y}_1), .., (\mathbf{x}_M, \mathbf{y}_M)$,

```
Data: RGB-D video frames
Result: Future sub-activity and affordance anticipations
$t = 0, P = \{\}$ ;
while new frame $f_t$ observed do
    Generate frame features for frame $f_t$ (Section 4.6);
    if temporal segmentation not given then
        Find best segmentation using additive energy $E_{\mathbf{w}}(\mathbf{y}, \Phi^A(\mathbf{x}))$ (solve Eq. 4.13);
        Sample segmentations by split and merge moves (Section 4.7.1);
    end
    Compute segment features $\phi_n$ and $\phi_e$ (Section 4.6);
    Compute $\hat{\mathbf{y}}$, best labeling of the past-CRF (Eq. 4.11-4.12);
    for each object do
        Sample possible future affordance and sub-activity from the discrete distribution $P(O^d, \mathcal{A}^d | \Phi^t_{\mathcal{H}}, \Phi^t_{\mathcal{L}})$;
        Sample future object location based on the affordance heatmaps $\psi_o$;
        Generate corresponding object trajectory and human poses for $d$ future frames;
        Augment the past-CRF to generate an ATCRF particle $\mathbf{g}^{t,d^{(s)}}$ ;
        $P = P \cup \{\mathbf{g}^{t,d^{(s)}}\}$;
    end
    for each particle $\mathbf{g}^{t,d^{(s)}} \in P$ do
        for each augmented frame do
            Generate frame features (Section 4.6);
        end
        if temporal segmentation not given then
            Find best segmentation using additive energy $E_{\mathbf{w}}(\mathbf{y}, \Phi^A(\mathbf{x}))$ (solve Eq. 4.13);
            Sample segmentations by split and merge moves (Section 4.7.1);
        end
        Compute segment features $\phi_n$ and $\phi_e$ (Section 4.6);
        Compute $\hat{\mathbf{y}}$, best labeling for the ATCRF particle (Eq. 4.11-4.12);
        Compute weight $\hat{w}^s_t$ (Eq. 4.8);
    end
    $P$ = top-k scored particles in $P$;
    $A_t$ = future sub-activity and affordance labels of top-3 particles based on $E_{\mathbf{w}}(\mathbf{y}, \Phi(\mathbf{x}))$;
    $t = t + 1$;
end
```

Algorithm 1: Summary of our inference method.

we optimize the regularized upper bound on the training error.

$$R(h) = \frac{1}{M}\sum_{m=1}^{M}\Delta(\mathbf{y}_m, \hat{\mathbf{y}}_m),$$

where $\hat{\mathbf{y}}_m$ is the optimal solution of Eq. (4.10) and $\Delta(\mathbf{y}, \hat{\mathbf{y}})$ is the loss function defined as

$$\Delta(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{i\in\mathcal{V}_o}\sum_{k\in K_o}|y_i^k - \hat{y}_i^k| + \sum_{i\in\mathcal{V}_a}\sum_{k\in K_a}|y_i^k - \hat{y}_i^k|.$$

To simplify notation, note that Eq. (4.11) can be equivalently written as $\mathbf{w}^T\Gamma(\mathbf{x}, \mathbf{y})$ by appropriately stacking the w_n^k and $\mathbf{w}_{\mathbf{t}}^{lk}$ into $\mathbf{w}$ and the $y_i^k\phi_a(i)$, $y_i^k\phi_o(i)$ and $z_{ij}^{lk}\phi_t(i,j)$ into $\Gamma(\mathbf{x}, \mathbf{y})$, where each z_{ij}^{lk} is consistent with Eq. (4.12) given $\mathbf{y}$. Training can then be formulated as the following convex quadratic program [81]:

$$\begin{aligned}
\min_{w,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi \\
s.t. \quad & \forall\bar{\mathbf{y}}_1, ..., \bar{\mathbf{y}}_M \in \{0, 0.5, 1\}^{N\cdot K}: \\
& \frac{1}{M}\mathbf{w}^T\sum_{m=1}^{M}[\Gamma(\mathbf{x}_m, \mathbf{y}_m) - \Gamma(\mathbf{x}_m, \bar{\mathbf{y}}_m)] \geq \Delta(\mathbf{y}_m, \bar{\mathbf{y}}_m) - \xi,
\end{aligned} \tag{4.14}$$

where N is the number of temporal segments per example and K is the total number class labels. While the number of constraints in this QP is exponential in M, N and K, it can nevertheless be solved efficiently using the cutting-plane algorithm [81]. The algorithm needs access to an efficient method for computing

$$\bar{\mathbf{y}}_m = \underset{\mathbf{y}\in\{0,0.5,1\}^{N\cdot K}}{\operatorname{argmax}}\left[\mathbf{w}^T\Psi(\mathbf{x}_m, \mathbf{y}) + \Delta(\mathbf{y}_m, \mathbf{y})\right]. \tag{4.15}$$

Due to the structure of $\Delta(.,.)$, this problem is identical to the relaxed prediction problem in Eqs. (4.11)-(4.12) and can be solved efficiently using graph cuts.

4.8 Experiments

In this section we describe the detailed evaluation of our approach on both offline data as well as realtime robotic experiments. We first give the details of the dataset in Section 4.8.1. We then present the detection results in Section 4.8.2, anticipation results in Section 4.8.3 and the robotic experiments for anticipation in Section 4.8.4.

4.8.1 Data

We use the CAD-120 dataset, which has 120 RGB-D videos of four different subjects performing 10 high-level activities. The data is annotated with object affordance and sub-activity labels and includes ground-truth object categories, tracked object bounding boxes and human skeletons. The set of high-level activities are: {*making cereal*, *taking medicine*, *stacking objects*, *unstacking objects*, *microwaving food*, *picking objects*, *cleaning objects*, *taking food*, *arranging objects*, *having a meal*}, the set of sub-activity labels are: {*reaching*, *moving*, *pouring*, *eating*, *drinking*, *opening*, *placing*, *closing*, *scrubbing*, *null*} and the set of affordance labels are: {*reachable*, *movable*, *pourable*, *pourto*, *containable*, *drinkable*, *openable*, *placeable*, *closable*, *scrubbable*, *scrubber*, *stationary*}. We use all sub-activity classes for prediction of observed frames but do not anticipate *null* sub-activity.

Table 4.1: **Results on CAD-120 dataset for *detection*,** showing average micro precision/recall, and average macro precision and recall for affordances, sub-activities and high-level activities. Computed from 4-fold cross validation with testing on a new human subject in each fold. Standard error is also reported.

With ground-truth segmentation.

	Object Affordance			Sub-activity			High-level Activity		
	micro	macro		micro	macro		micro	macro	
method	*P/R*	Prec.	Recall	*P/R*	Prec.	Recall	P/R	Prec.	Recall
chance	8.3 (0.0)	8.3 (0.0)	8.3 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)
max class	65.7 (1.0)	65.7 (1.0)	8.3 (0.0)	29.2 (0.2)	29.2 (0.2)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)
Our model: all features	**93.9** (0.4)	**89.2** (1.3)	**82.5** (2.0)	**89.3** (0.9)	**87.9** (1.8)	**84.9** (1.5)	**93.5** (3.0)	**95.0** (2.3)	**93.3** (3.1)
Our model: only additive features	92.0 (0.5)	84.2 (2.2)	72.4 (1.2)	86.5 (0.6)	84.2 (1.3)	78.7 (1.9)	90.3 (3.8)	92.8 (2.7)	90.0 (3.9)

Without ground-truth segmentation.

Our DP seg.	83.6 (1.1)	70.5 (2.3)	53.6 (4.0)	**71.5** (1.4)	71.0 (3.2)	60.1 (3.7)	80.6 (4.1)	86.1 (2.5)	80.0 (4.2)
Our DP seg. + moves	84.2 (0.9)	72.6 (2.3)	58.4 (5.3)	71.2 (1.1)	70.6 (3.7)	61.5 (4.5)	83.1 (5.2)	**88.0** (3.4)	**82.5** (5.4)
heuristic seg. (Chapter 3)	83.9 (1.5)	75.9 (4.6)	64.2 (4.0)	68.2 (0.3)	71.1 (1.9)	62.2 (4.1)	80.6 (1.1)	81.8 (2.2)	80.0 (1.2)
Our DP seg. + moves + heur. seg.	**85.4** (0.7)	**77.0** (2.9)	**67.4** (3.3)	70.3 (0.6)	**74.8** (1.6)	**66.2** (3.4)	83.1 (3.0)	87.0 (3.6)	82.7 (3.1)

4.8.2 Detection Results

For comparison, we follow the same train-test split described in Chapter 3 and train our model on activities performed by three subjects and test on activities of a *new subject*. We report the results obtained by 4-fold cross validation by averaging across the folds. We consider the overall micro accuracy (P/R), macro precision and macro recall of the detected sub-activities, affordances and overall activity. Micro accuracy is the percentage of correctly classified labels. Macro precision and recall are the averages of precision and recall respectively for all classes.

Table 4.1 shows the performance of our proposed approach on object affordance, sub-activity and high-level activity labeling for past activities. Rows 3-4 show the performance for the case where ground-truth temporal segmentation

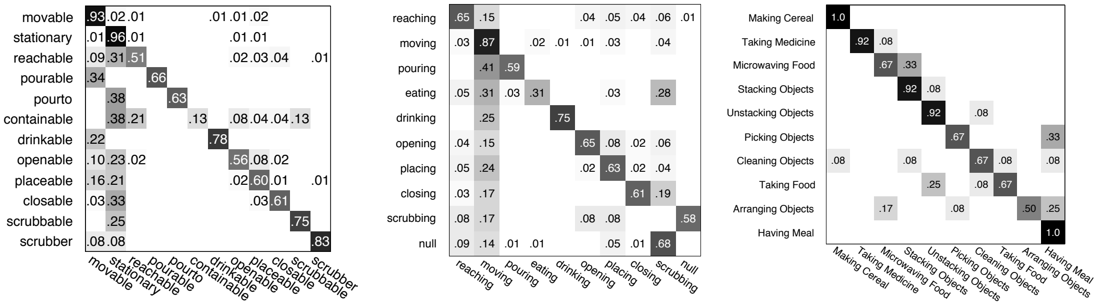


Figure 4.8: **Confusion matrix** for affordance labeling (left), sub-activity labeling (middle) and high-level activity labeling (right) of the test RGB-D videos.

is provided and rows 5-8 show the performance for the different methods when no temporal segmentation is provided. With known graph structure, the model using the the full set of features (row 3) outperforms the model which uses only the additive features (row 4): macro precision and recall improve by 5% and 10.1% for labeling object affordance respectively and by 3.7% and 6.2% for labeling sub-activities respectively. This shows that additive features bring us close, but not quite, to the optimal graph structure.

When the graph structure is not known, the performance drops significantly. Our graph sampling approach based on the additive energy function (row 5) achieves 83.6% and 71.5% micro precision for labeling object affordance and sub-activities, respectively. This is improved by sampling additional graph structures based on the Split and Merge moves (row 6). Finally, combining these segmentations with the other heuristically generated segmentations presented in Chapter 3, our method obtains the best performance (row 8).

Figure 4.8 shows the confusion matrix for labeling affordances, sub-activities and high-level activities using our method (row 8). Note that there is a strong diagonal with a few errors such as *pouring* misclassified as *moving*, and *picking* *objects* misclassified as *having a meal*. Figure 4.9 shows the labeling output of the different methods. The bottom-most row show the ground-truth segmentation, top-most row is the labeling obtained when the graph structure is provided, followed by three heuristically generated segmentations. The fifth row is the segmentation generated by our sampling approach and the sixth and seventh rows are the labeling obtained by combining the multiple segmentations using a simple max-voting and by the multi-segmentation learning, respectively. Note that some sub-activity boundaries are more ambiguous (high variance among different methods) than the others. Our method has an end-to-end (including feature computation cost) frame rate of 4.3 frames/sec compared to 16.0 frames/sec of Chapter 3.

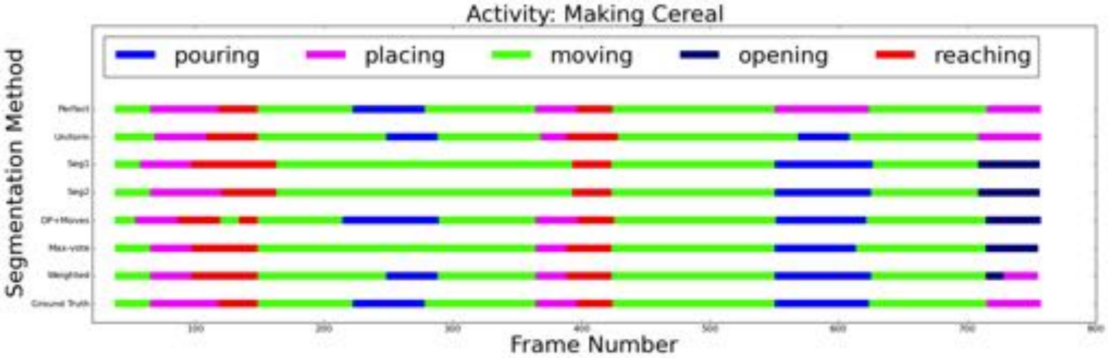


Figure 4.9: **Illustration of the ambiguity in temporal segmentation.** We compare the sub-activity labeling of various segmentations. Here, *making cereal* activity comprises the sub-activities: *reaching*, *moving*, *pouring* and *placing* as colored in red, green, blue and magenta respectively. The x-axis denotes the time axis numbered with frame numbers. It can be seen that the various individual segmentation methods are not perfect.

4.8.3 Anticipation Results

Baseline Algorithms. We compare our method against the following baselines: *1) Chance.* The anticipated sub-activity and affordance labels are chosen at random.

2) Nearest Neighbor Exemplar. It first finds an example from the training data which is the most similar to the activity observed in the last temporal segment. The sub-activity and object affordance labels of the frames following the matched frames from the exemplar are predicted as the anticipations. To find the exemplar, we perform a nearest neighbor search in the feature space for the set of frames, using the node features described in Chapter 3.

3) Co-occurrence Method. The transition probabilities for sub-activities and affordances are computed from the training data. The observed frames are first labelled using the CRF model from Chapter 3. The anticipated sub-activity and affordances for the future frames are predicted based on the transition probabilities given the inferred labeling of the last frame.

4) ATCRF without $\{\mathcal{H}, \mathcal{L}\}$ *anticipation (ATCRF-discrete).* Our ATCRF model with only augmented nodes for discrete labels (sub-activities and object affordances).

5) ATCRF. Our method that samples the future nodes (both segment and frame level) as described in Section 4.5.3, and uses a fixed temporal structure, which in this case is the segmentation output of Chapter 3.

Evaluation: We follow the same train-test split described in Chapter 3 and train our model on activities performed by three subjects and test on activities of a *new subject*. We report the results obtained by 4-fold cross validation by averag-

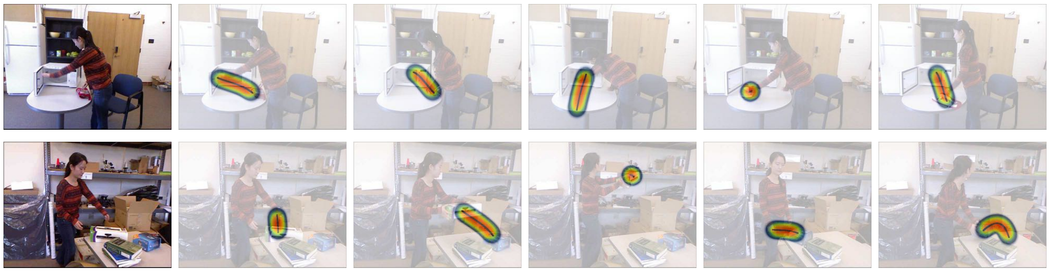


Figure 4.10: **Highest scored future anticipations** for *cleaning objects* activity (top-row) and *arranging objects* activity (bottom-row).

ing across the folds. We consider the following metrics:

1) Labeling Metrics. For detecting and anticipating labels (for sub-activity and affordances), we compute the overall micro accuracy (P/R), macro precision, macro recall and macro F1 score. Micro accuracy is the percentage of correctly classified labels. Macro precision and recall are the averages of precision and recall respectively for all classes.

2) Robot Anticipation Metric. It is important for a robot to plan ahead for multiple future activity outcomes. Therefore, we measure the accuracy of the anticipation task for the top three predictions of the future. If the actual activity matches one of the top three predictions, then it counts towards positive.

3) Trajectory Metric. For evaluating the quality of anticipating trajectories, we compute the modified Hausdorff distance (MHD) as a physical measure of the distance between the anticipated object motion trajectories and the true object trajectory from the test data.[8]

---

[8] The MHD allows for local time warping by finding the best local point correspondence over a small temporal window. When the temporal window is zero, the MHD is same as the Euclidean distance between the trajectories. We normalize the distance by the length of the trajectory in order to compare across trajectories of different lengths. The units of the MHD are centimeters.

Table 4.2: **Anticipation Results of Future Activities and Affordances,** computed over 3 seconds in the future (similar trends hold for other anticipation times).

model	Anticipated Sub-activity			Anticipated Object Affordance		
	micro P/R	macro F1-score	robot anticipation metric	micro P/R	marco F1-score	robot anticipation metric
chance	10.0 ± 0.1	10.0 ± 0.1	30.0 ± 0.1	8.3 ± 0.1	8.3 ± 0.1	24.9 ± 0.1
Nearest-neighbor	22.0 ± 0.9	10.6 ± 0.6	48.1 ± 0.5	48.3 ± 1.5	17.2 ± 1.0	60.9 ± 1.1
CRF + co-occurence	28.6 ± 1.8	11.1 ± 0.4	34.6 ± 2.8	55.9 ± 1.7	11.6 ± 0.4	62.0 ± 1.8
ATCRF-discrete	34.3 ± 0.8	12.2 ± 0.2	44.8 ± 1.1	59.5 ± 1.5	12.4 ± 0.3	67.6 ± 1.3
ATCRF	47.7 ± 1.6	37.9 ± 2.6	69.2 ± 2.1	66.1 ± 1.9	36.7 ± 2.3	71.3 ± 1.7
Ours-full	**49.6** ± 1.4	**40.6** ± 1.6	**74.4** ± 1.6	**67.2** ± 1.1	**41.4** ± 1.5	**73.2** ± 1.0

Table 4.2 shows the frame-level metrics for anticipating sub-activity and object affordance labels for 3 seconds in the future on the CAD-120 dataset. We use the temporal segmentation algorithm from Chapter 3 for obtaining the graph structure of the observed past frames for all the baseline methods. ATCRF (row 5) outperforms all the baseline algorithms and achieves a significant increase across all metrics. Our full model (row 6), which estimates the graph structure for both past and the future, improves the anticipation performance further. Figure 4.10 shows the highest scored anticipations for the *cleaning objects* activity. We will now study our results on anticipation in the form of the following questions:

How does the performance change with the duration of the future anticipation? Figure 4.11 shows how the macro F1 score and the *robot anticipation metric* changes with the anticipation time. The average duration of a sub-activity in the CAD-120 dataset is around 3.6 seconds, therefore, an anticipation duration of 10 seconds is over two to three sub-activities. With the increase in anticipation duration, performance of the others approach that of a random chance baseline, the performance of our ATCRF declines. It still outperforms other baselines for all anticipation times.

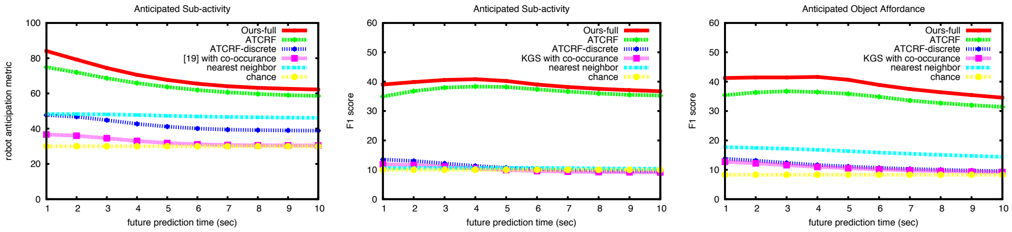


Figure 4.11: Plots showing how *robot anticipation metric* and macro F1 score changes with the future anticipation time for all methods.

How does the performance change with the duration of the past observations? Figure 4.12 shows how the macro F1 score changes with the past observation time and future anticipation time. The algorithm has lower performance when predicting longer into the future, but this improves as more observations become available. Therefore, context from the past helps in anticipating longer into the future.

How good are the anticipated trajectories? Since trajectories are continuous variables, we perform two types of estimation: MAP, where we take the highest scored particle generated by our model, and MLE where we take the weighted sum. Figure 4.13 shows how these distance errors, averaged over all the moving sub-activities in the dataset, change with the progress of the sub-activity. Figure 4.6 shows the sampled trajectories along with the heatmap corresponding to the distribution of trajectories. At the beginning of the sub-activity the anticipations correspond to moving the cup to other places on the table and near the mouth to drink. As the sub-activity progresses, depending on the current position of the cup, a few new target locations become probable, such as moving the cup on to the lap (such instances are observed in the training data). These new

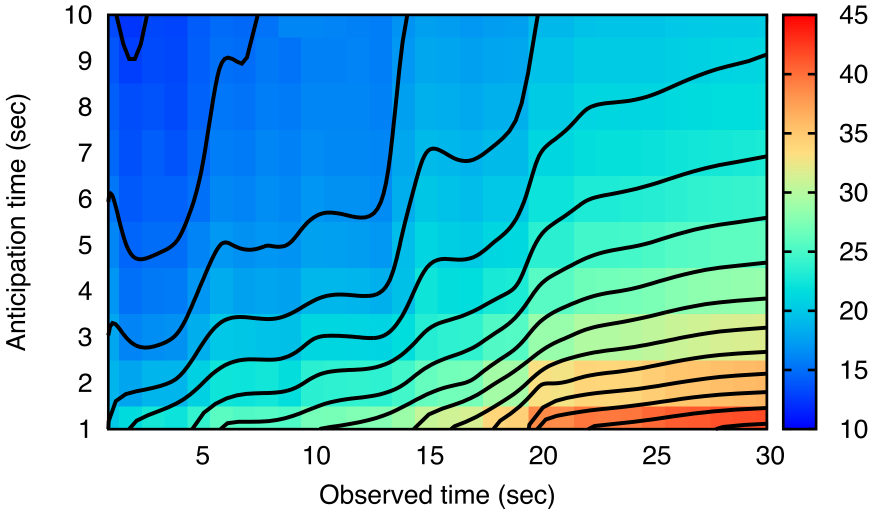


Figure 4.12: Plot showing how macro F1 score depends on observed time and anticipation time.

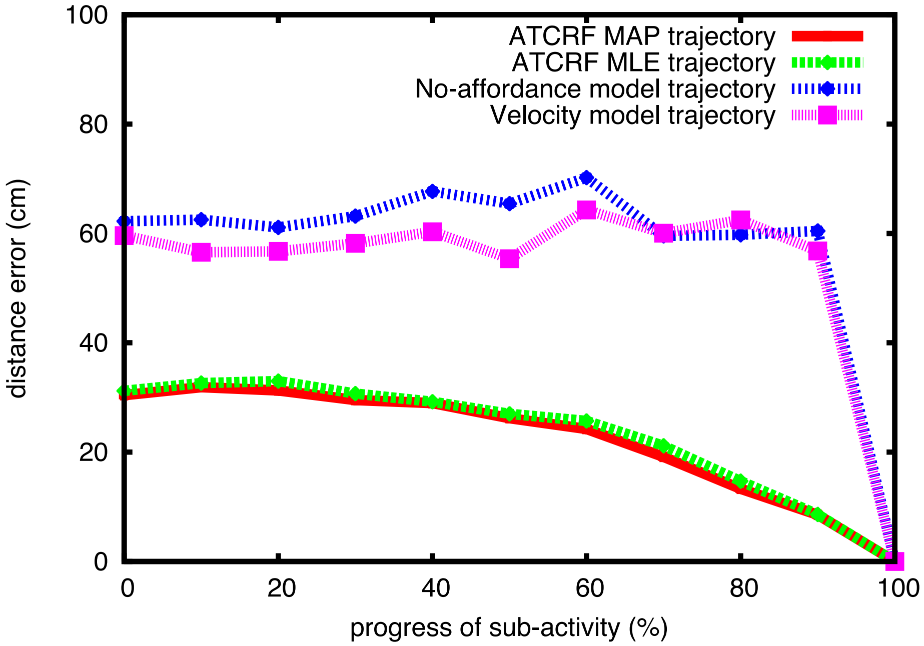


Figure 4.13: Plot showing how the trajectory distance error (MHD) changes with the progress of the activity for our ATCRF (top particle and local mean) and other baselines (Kalman Filter velocity model using object affordance as target, and one without object affordance information).

Table 4.3: **Online Detection Results** of Past Activities and Affordances.

model	Past Sub-activity Detection			Past Object Affordance Detection		
	micro	macro		micro	macro	
	P/R	Prec.	Recall	P/R	Prec.	Recall
chance	10.0 (0.1)	10.0 (0.1)	10.0 (0.1)	8.3 (0.1)	8.3 (0.1)	8.3 (0.1)
CRF-online	80.3 (1.5)	78.0 (1.3)	68.1 (2.6)	89.6 (0.8)	80.7 (2.8)	67.8 (1.4)
ATCRF-discrete	84.0 (1.3)	72.2 (2.3)	60.7 (2.3)	87.7 (1.0)	67.9 (2.4)	48.9 (2.6)
ATCRF	**84.7** (1.4)	**80.6** (1.0)	**75.6** (2.4)	**92.3** (0.7)	**84.8** (2.3)	**77.1** (1.1)

possibilities tend to increase the distance measure as can be seen in the plot of Figure 4.13. However, on observing more frames, the intent of the human is inferred more accurately resulting in better anticipated trajectories, for example in Figure 4.6-last frame, anticipating only moving to drink trajectories.

Effect of anticipation on detection of past activities. Table 4.3 shows the detection results of the sub-activities and object affordances of the past temporal segments, computed in an online fashion. When we label each past segment, we observe that segment's features but not the future. The online metrics are computed by aggregating performance on the recent past of three segments. (CRF from Chapter 3 was used to label a segment given past, present, as well as the future.) In this experiment, we assumed ground-truth segmentation and object tracks for consistent comparison across the methods. If we instead use our algorithm to segment, the overall performance drops, however similar trends hold. We see that both the anticipation methods (rows 3-4) improve the detection results over the one that does not anticipate (row 2). This shows that anticipating the future can improve present and past performance on detection.

Effect of unknown activities on anticipation performance. For activities not present in the training data, our approach generates most likely anticipations based on the affordances of the objects present in the environment and the detected past sub-activities. However, our approach cannot anticipate a sub-activity on which the anticipation model was not trained. In fact, around 14% of the segments in our dataset are sub-activities which do not belong to the nine sub-activity categories used to train our anticipation model. These sub-activities include various background actions such as *checking time*, *standing still*, etc., which are not relevant to the high-level activity being performed. We label these segments as *null* sub-activities and include them for learning the energy function in Eq. 4.4. Therefore, even though our model is unable to anticipate such sub-activities in the future, they are correctly detected as *null* sub-activities. This allows us to ignore such irrelevant sub-activities and proceed to anticipate the most likely future. However, if *null* sub-activities are performed very often, the performance of our anticipation model would go down further.

4.8.4 Robotic Experiments

In this section we show how future activity predictions can help the robot perform appropriate actions in response to what the human is going to do next. By incorporating such reactive responses, the robot can better assist humans in tasks which they are unable to perform as well as work along side the humans much more efficiently.

We use a PR2 robot to demonstrate the following anticipatory response scenarios: (i) Robot is instructed to refill water glasses for people seated at a table, but when it sees a person reaching a glass to drink, it waits for him to finish

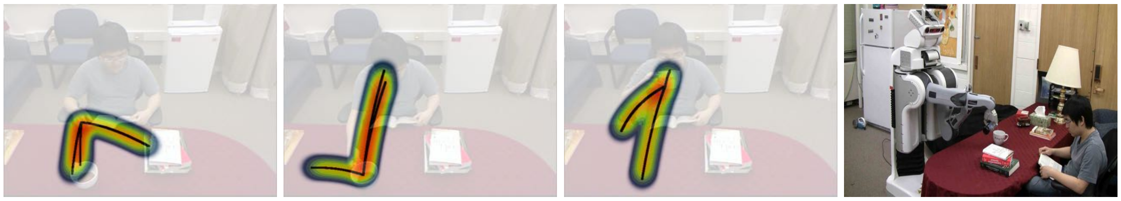


Figure 4.14: **Robot Anticipatory Response** for refilling water task. See Figure 4.1 for opening fridge door task.

drinking before refilling, in order to avoid spilling, and (ii) Robot opens the fridge door when a person approaches the fridge to place something inside the fridge. PR2 is mounted with a Kinect as its main input sensor to obtain the RGB-D video stream. We used the OpenRAVE libraries [25] for programing the robot to perform the pre-programmed tasks described in the aforementioned scenarios by incorporating the anticipations generated with our ATCRFs. Figure 4.1 and Figure 4.14 show the snapshots of the robot observing the human, the anticipated actions and the response executed by the robot.

In our experiments, on the first scenario, we evaluate the success rate which is defined as the percentage of times the robot identifies the correct response. We have a new subject (not seen in the training data) performing the interaction task multiple times in addition to other activities which should not effect the robot's response, such as reaching for a book, etc. We considered a total of 10 interaction tasks which involve four objects including the cup, and 5 of these tasks were to reach for the cup and drink from it. The robot is given an instruction to pour water in the cup at four random time instants during each interaction tasks (40 total pour instructions). The robot makes a decision whether to execute the pouring task or not, based on the anticipated activity and object affordance. The robot considers the three top scored anticipations for taking the decision following the *robot anticipation metric*.

We obtain a success rate of 85%, which is the fraction of times the robot correctly identifies its response ('pour' or 'not pour'). Out of the 6 failed instances, 3 instances are false-negatives, i.e., the robot anticipated an interaction with the cup when no interaction occurred in future. Videos showing the results of our robotic experiments and code are available at: `http://pr.cs.cornell.edu/anticipation/`.

4.9 Conclusion

In this Chapter, we considered the problem of detecting the past human activities as well as anticipating the future using object affordances. We showed how the anticipation of future activities can be used by a robot to perform look-ahead planning of its reactive responses. We modeled the human activities and object affordances in the past using a rich graphical model (CRF), and extended it to include future possible scenarios. Each possibility was represented as a potential graph structure and labeling over the graph (which includes discrete labels as well as human and object trajectories), which we called ATCRF. We used importance sampling techniques for estimating and evaluating the most likely future scenarios.

The structure of the ATCRF was obtained by first considering the potential graph structures that are close to the ground-truth ones by approximating the graph with only additive features. We then designed moves to explore the space of likely graph structures. We showed that anticipation can improve performance of detection of even past activities and affordances. We also extensively evaluated our algorithm, against baselines, on the tasks of anticipating activity and affordance labels as well as the object trajectories.

Our work shows a considerable advance by improving the state-of-the-art results on both the detection and anticipation tasks. We have focused on the algorithms for estimating the graph structure for both past and future activities while using given noisy skeleton and object tracks. Improvements to human pose estimation and object perception would further improve these results. In our experiments, we see that there is still a large gap between the detection performance with and without ground-truth temporal segmentation. Incorporating additional priors about the activities in future work would improve the estimation of the graph structure. Also, we see that the anticipation accuracies fall rapidly with future prediction time. We believe that modeling larger temporal-range dependencies and hierarchical structure of activities is an interesting direction to explore for obtaining better anticipation.

CHAPTER 5

COLLABORATIVE PLANNING FOR HUMAN-ROBOT TEAMS

Robots are being incorporated into human workspaces where they perform tasks with humans – assistive settings in nursing homes [54, 157], collaborative assembly line manufacturing [143], or in other outdoor applications [198]. The challenge here is two-fold: the robots often have to operate in *contextually-rich environments*, where they have to perform tasks involving manipulation of objects, and they have to work closely *with humans* performing the same task (see Fig. 5.1).

Collaborative tasks are more challenging as compared to both reactive and role-based tasks. In collaborative tasks, the goal of the robot is to perform actions along side humans in order to achieve the goal of the task. For example, if the task is to set the dinner table, the various actions involved are reaching for the objects (e.g., plates, cups and spoons), and moving them to appropriate locations on the table. The robot can perform any action in order to achieve the goal as opposed to a role-based scenario where the robot has a pre-assigned role of setting plates or cups, etc. It needs to plan its actions by taking into account the actions of the human. In order to achieve this, there are three aspects we need to address: (i) model the contextually-rich environment to reason about what can be done and how, (ii) perceiving the human's actions and anticipating their future moves, and (iii) plan robot's actions taking into account the inherent uncertainty in the human actions.

In Chapters 3 and 4, we presented a perception algorithm for modeling the spatio-temporal relations of activities which allows us to detect the past actions and anticipate the future actions. However, the robotic responses were only re-

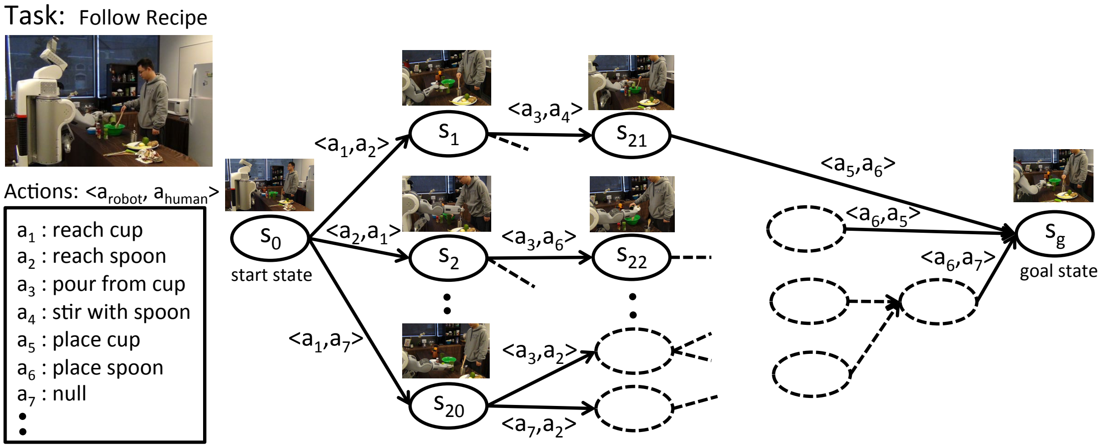


Figure 5.1: **Robot performing collaborative task with human:** The human and the robot are asked to follow a recipe – pour the ingredients in to the bowl and stir. At every time step both the agents execute an action and change the state of the environment. The robot needs to plan its actions by taking in to account what actions the human would perform, where some human actions are more likely than the others based on the human's strategy.

active and hand-designed. In related works, [144] consider collaboration for assembling tasks with pre-assigned roles for human and robot, where they do not explicitly model anticipation. [125] anticipate human actions to minimize penetration of robot in human workspace and [199] introduced social-affordances for planning. In comparison, we look at a more generic collaborative task planning problem, where the role of robot and human are indistinguishable.

In this chapter, we formulate the collaborative task completion problem as a two-agent planning problem, where we model the ambiguities in perception as well as in the human's choice of actions. Unlike planning for multi-robot scenarios, where one has control over all agent behaviors [9, 196], the human does not perform his actions according to a fixed strategy. Humans tend to follow their habits when possible in a familiar environment, but will also try to adapt in response to the other agents in the environment. Therefore, our problem of robot-human collaborative planning can be viewed as a two-agent cooperative Markov game, where the goal of each game is to complete a pre-specified activity in a given environment. We aim to learn the optimal policy for the robot while taking into account the various human behaviors or strategies.

In detail, we represent the contextually rich environment in terms of the object affordances and incorporate them as the states of our collaborative Markov decision process. We propose a distributed Q-learning algorithm to learn the policies for both the agents. We model the human's actions in several ways—taking the ϵ-optimal action according to the MDP model, taking actions based on past habits as seen in a RGB-D video dataset, and taking appropriate actions by adapting to the environment and robot actions. Each human behavior results in exploring a different subspace of states by the robot, resulting in a different robot policy as shown in Fig. 5.1. Therefore, during learning we first estimate how adaptive the human is in the given environment and then jointly estimate the robot and human policies.

We evaluate our approach on five high-level activities in 60 environments from the Cornell activity dataset as well as in a user study. We predict the current object affordances from RGB-D videos and use our algorithm to plan appropriate actions by the robot to be performed along with the human. We compare our approach against the baselines on several metrics, and find that our approach performs better collaborative planning. Specifically, our robot policy learnt with an adaptive human model completes the tasks 36.5% faster as compared to 13.8% when the human agent is not modeled explicitly

5.1 Related Work

Our approach of anticipatory planning has three main aspects: human-robot interaction, perception in contextually-rich environments, and planning algorithms. We now review the relevant works specific to these aspects.

Human-robot collaboration. Many tasks are parallelizable or involve complex interactions with objects in the environment, and can be more efficiently completed if human and robot collaborate. Some recent works have addressed this problem of collaboration in human-robot teams. Nikolaidis et al. [144] consider collaboration for assembling tasks with pre-assigned roles for human and robot. Mainprice et al. [125] anticipates human actions to minimize penetration of robot in human workspace. Uyanik et al. [199] introduced social-affordance where robot's action depends on help from human. As opposed to them, in our work the role of robot and human are indistinguishable, and for task completion they interact with multiple objects performing different activities.

Another aspect of human-robot collaboration is the interaction between the agents and their compatibility. Some works [126, 181, 130] encode the compatibility in the form of constraints on the distance of robot from user, the visibility of robot and user arm comfort. Strabala et al. [183] and Cakmak et al. [13] consider handover tasks where the robot reasons about its location w.r.t. human and handover configuration. We differ from these in that, in our tasks both human and robot are active participants and collaborate towards a common goal.

Affordances. The concept of affordances was described by J.J. Gibson [42] as the "Action possibilities in the environment in relation to the action capabilities of an actor". Affordances have been widely used in robotics for obtaining a functional understanding of the scene as well as enabling robots to interact and manipulate objects. These works range from predicting opportunities for interaction with an object by using only visual cues [184, 59, 5] to observing effects of exploratory behaviors [136, 133, 60, 83]. For instance, Sun et al. [184] proposed a probabilistic graphical model that leverages visual object categorization for learning affordances. Katz et al. [83] propose a framework for learning to manipulate objects in clutter by choosing robot actions based on object affordances.

There is some recent work in interpreting human actions and interaction with objects [120, 3, 90, 78, 93] in context of learning actions from demonstrations. Lopes et al. [120] use context from objects in terms of possible grasp affordances to focus the attention for recognition. Aksoy et al. [3] propose a dynamic graph sequence representation, constructed from human demonstrations, for robot manipulation. Affordances have also been used in planning (e.g., [121, 197]). In this work, we use object affordances to represent the state of the environment, and these affordances evolve as the objects are used in an activity (see Chapter 3).

Multi-agent Reinforcement Learning. The multi-agent reinforcement learning (MARL) literature focus on multiple autonomous agents learning how to solve dynamic tasks online. Besides single-agent reinforcement learning, MARL has strong connections with game theory, evolutionary computation, and optimization theory. We refer the reader to [12] for a survey of the works in this area and discuss some relevant ideas here. Many multi-agent algorithms exist for different tasks which range from fully cooperative setting [109, 88] to fully competitive setting [116]. When collaborating with humans, the robot needs to be aware of the human's behavior, which might not always be fully cooperative.

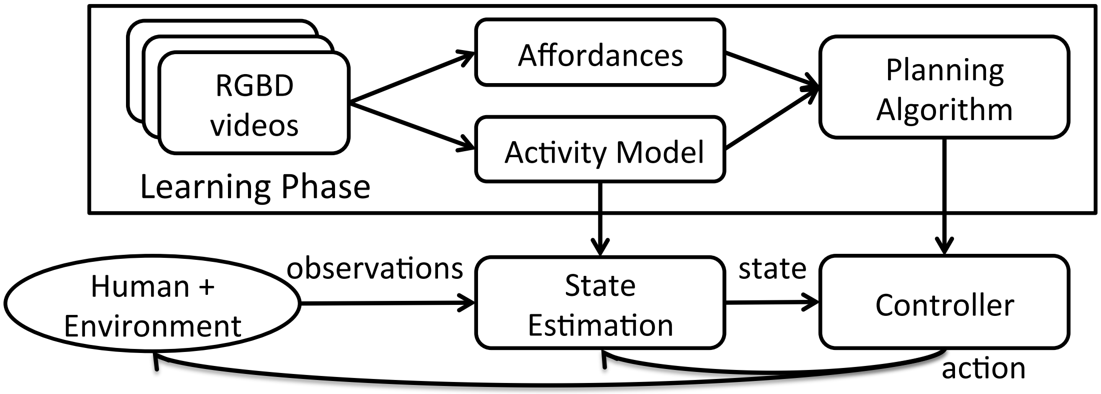


Figure 5.2: **System Overview.** The planning algorithm uses the affordance and activity models, learnt from videos of people performing various activities, to find the best action for the robot to perform for any given state of the environment. When working with the human, the robot first uses the learnt activity model to estimate the state of the environment and then executes the best action based on its learned policy.

Adaptation of agents has been studied previously [192, 153, 203], where an agent's adaptation depends on the degree of awareness of other agent's behavior maintained by the learning algorithms. These algorithms use some form of opponent modeling to keep track of the other agent's policies [14, 68]. There is a tradeoff between the stability (convergence) of the algorithms and the degree of adaptability. We build upon some of these ideas and propose a two-agent reinforcement learning algorithm, which models the various human behaviors allowing the robot to learn an adaptive policy.

5.2 Approach

Our goal is to learn which actions a robot can perform in order to collaborate with the human and assist in the task. As illustrated in Fig. 5.2, we first learn the spatio-temporal structure of activities using a conditional random field (CRF) from RGB-D videos of people performing these activities. We model the sub-activities and affordances of the objects, how they change over time, and how they relate to each other (for details see Chapter3). We then learn a Q-value function in simulation using the learnt activity and affordance models. When working with the human, the robot first estimates the state of the environment by detecting the object affordances and human actions, and then chooses an appropriate action and executes it.

In detail, we consider a robot r working with a human h in an environment having objects O. The goal is to learn a policy for the robot, π^r, which maps the current environment to an action. We formulate the collaborative task planning problem as a Markov decision process (MDP) with two agents – the human and the robot. We define the following:

- State Space $\mathcal{S}$: Let $\mathbf{s_t} = \{s_t^1, ..., s_t^n\}$ denote the state of the environment, where s_t^i denotes the state of the i^{th} object at time t and n denotes the number of objects.
- Action Space $\mathcal{A}$: Let $a_t = \langle a_t^h, a_t^r \rangle$ denote the joint action at time t, where a_t^h and a_t^r denote the human and robot actions respectively.
- Robot's policy π^r: $\mathcal{S} \times \mathcal{A}^r \rightarrow [0, 1]$, where $\mathcal{A}^r$ denotes the set of possible robot actions. $\pi^r(\mathbf{s}, a^r)$ specifies the probability of choosing action a^r in state $\mathbf{s}$.

We address the following challenging aspects of this problem: (i) Defining an efficient state-action representation that captures the contextually rich environments for performing complex activities, (ii) Learning task models which specify the effect of actions on the environment and the actions required for completing the task, and (iii) Modeling human's actions for learning the robot's policy.

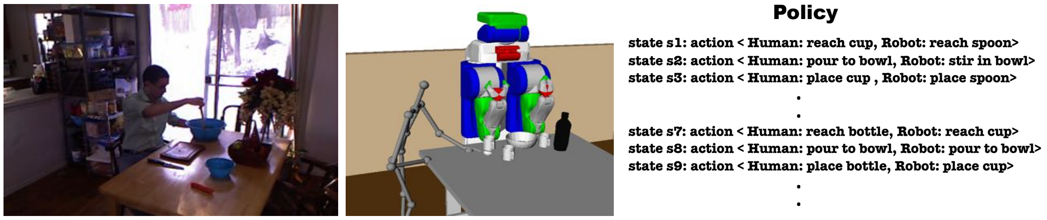
5.2.1 Collaborative Markov Decision Processes

We use RGB-D videos of a single human performing the activities to define the state-action representation and learn the task model of the activities.[1] Once we have the set of states, set of actions and the task model, we can solve the MDP using dynamic programming techniques [154]. However, with large joint state-action space, computing the optimal policy is computationally very expensive and therefore, we take the model-free approach of Q-learning and learn the Q-functions offline with the help of the learnt task models. When collaborating with humans, the robot choses actions greedily with respect to its learnt Q-function. We fix the robot's policy after the offline learning, however, one can also further refine the Q-functions on-the-fly while working with humans in the real world. We now describe the details of our collaborative MDP algorithm.

State-Action Representation: We represent the environment in terms of the object affordances, which leads to an efficient state action space for planning. For example in Fig. 5.3, the state of the environment is represented in terms of the affordance labels of the objects in the scene, i.e., the bowl is *stirrable*, the spoon is the *stirrer* and the rest of the objects are *stationary*. The *stir* action corresponds to the temporal motion trajectory of the spoon from the grounded stir affordance. On performing the stir action, the spoon becomes *placeable*, thus changing the

---

[1] Such data is easier to collect for a wide variety of activities in a variety of environments [200, 97] as compared to collecting data of humans working with robots.



Policy

state s1: action < Human: reach cup, Robot: reach spoon>
state s2: action < Human: pour to bowl, Robot: stir in bowl>
state s3: action < Human: place cup , Robot: place spoon>
.
.
state s7: action < Human: reach bottle, Robot: reach cup>
state s8: action < Human: pour to bowl, Robot: pour to bowl>
state s9: action < Human: place bottle, Robot: place cup>
.
.

Figure 5.3: **Collaborative planning by the robot.** In order to collaborate with the human on a *recipe following* task, the robot learns the activity model from RGB-D videos of human preparing a recipe (left), represents the environment via affordances and uses our planning algorithm (middle) to generate a policy for jointly performing the activity with the human (right).

state of the environment.

Task Model: State Transitions and Rewards. The affordance-based representation of the environment allows for factored representation of the transition and reward functions. That is, it is sufficient to specify the state transitions with respect to only a subset of affordances that are effected by an action. For example, a move action would change only the state of the movable object where as a pour action would change the state of the pourable and the pour-to objects. We assume that each action can be completed in one time step and hence given the nature of activities and affordances, the state transitions are deterministic. That is, on performing a valid action, the affordance of the object changes to another fixed affordance. The reward function allows us to specify valid actions at any given affordance state, where all valid actions receive a fixed positive reward and non-valid actions will incur a negative cost. We compute these task models (i.e., transition and the reward functions) for each high-level activity from the labeled RGB-D videos of a human performing the activity in various environments.

Learning Robot Policy. Given the deterministic nature of the state transitions,

```
INPUT: State space $S$, Action space $A$
Initialize environment to start state;   $R \leftarrow 0$;   $i \leftarrow 0$;
loop
    if goal state then
        return  $Q$-functions, $R$
    end if
    Sample $a_h$ from $\pi^h$ and $a_r$ from $\pi^r$
    Take action $(a_h, a_r)$ and observe $r$, $\mathbf{s}'$
    Update $Q$-functions as in Eq. 5.2
    $R \leftarrow R + \gamma^i * r$
    $i \leftarrow i + 1$
end loop
```

Algorithm 2: RUN-EPISODE (Q, π^h, π^r)

we use the distributed Q-learning [109] algorithm to learn the local value functions $q_t^h(s,a)$ and $q_t^r(s,a)$ for the human and the robot respectively. Each agent assumes that other agents are acting optimally and only updates their local Q-functions when it results in an increase. This ensures that the local Q-value always captures the maximum of the joint-action Q-values. Therefore at each iteration, the local Q-functions are updated as in Eq. 5.1 while maintaining the invariants in Eq. 5.2.

$$q_{t+1}^j(\mathbf{s}_t, a_t^j) = \max\{q_t^j(\mathbf{s}_t, a_t^j),\ R(\mathbf{s}_t, a_t^h, a_t^r) + \gamma \max_{a \in \mathcal{A}^j} q_t^j(\mathbf{s}_{t+1}, a)\},\ j \in \{r, h\} \tag{5.1}$$

$$q_t^r(\mathbf{s}, a) = \max_{a^h \in \mathcal{A}^h} Q_t(\mathbf{s}, a^h, a^r = a); \quad q_t^h(\mathbf{s}, a) = \max_{a^r \in \mathcal{A}^r} Q_t(\mathbf{s}, a^h = a, a^r) \tag{5.2}$$

Our collaborative distributed Q-learning algorithm is summarized in Algorithm 2. Here, an episode is defined as the sequence of actions performed by the robot and human from the initial configuration to the goal configuration.

5.3 Models of Human Behavior

Many studies on human behavior have shown that there are primarily two systems which drive the way humans think – the first being fast, intuitive and emotional; and the second system which is slower, more deliberative and logical [82]. This also applies to our problem of performing collaborative tasks, where humans can either think fast and perform activities following their habits or think more carefully about collaborating by taking into account what the robot can do. Therefore, the actions chosen by the human can range from fully cooperative, when humans are thinking for collaboration, to somewhat adversarial when their habits conflict with the robot's actions. Modeling these various types of human behavior becomes extremely important for collaboration.

Previous works in game theory literature study such senarios in the setting of general sum Markov games, where the types of opponent behaviors have been roughly classified into fixed strategies or best-response strategies [117]. In the fixed strategy case, the opponent always executes a fixed unknown policy and Q-learning finds the best response with respect to the fixed opponent. In the second case, it is assumed that the opponent adapts and chooses the best response so that it is mutually beneficial to both agents. Following these ideas, we model the following behaviors of a human agent:

- ***Habit-following human.*** In this model, we consider the perceptual data of the human from RGB-D videos, and assume that the human follows close to what he has done in the training videos. This is a fixed strategy behavior, where the human has a preferred way of performing activities and follows the same approach even when working with a robot. Let D be the set of activity videos and let $c(\mathbf{s}, a)$ be the number of times the human performed action a when in state $\mathbf{s}$ in D. The policy followed by a habit-following human, $\pi_d^h(\mathbf{s}, a)$, is defined as

$$\pi_d^h(\mathbf{s}, a_i) = \begin{cases} c(\mathbf{s}, a_i) / \sum_a c(\mathbf{s}, a) & \text{if } \mathbf{s} \in D \\ 1/n & \text{if } \mathbf{s} \notin D \\ 0 & \text{otherwise} \end{cases}$$

where n is the number of possible actions in state $\mathbf{s}$.

- ***ϵ-optimal human.*** In this model, we assume that the human takes the best action according to the value function most of the time, but makes a random choice ϵ fraction of the time. Here, human chooses a response that is mutually beneficial most of the time, according to the value function learnt so far.[2] This is equivalent to the ϵ-greedy exploration strategy which was shown to have better convergence properties [204] compared to always choosing the action greedily. The human policy is defined as

$$\pi_\epsilon^h(\mathbf{s}, a_i) = \begin{cases} (1 - \epsilon) + (\epsilon/n) & \text{if } a_i = argmax_a(q^h(\mathbf{s}, a)) \\ \epsilon/n & \text{otherwise} \end{cases}$$

- ***Adaptive human.*** In the real world, when collaborating, humans usually adapt to other agents while trying to maintain their preferences or habits. That is, they follow their habits when possible in familiar situations, but when faced with new situations while working with the robot, they adapt

---

[2] The *ϵ-optimal human* behavior can differ from the *habit-following human* behavior, even when the reward model for learning the *ϵ-optimal human's* value function is extracted from the same data as the actions of *habit-following human*. There are two reasons for this: (i) The test environment is not present in the training data, and therefore, the reward function learnt from the training environments might not capture all valid ways of performing the activity in the test environment. This would lead to differences in what a human might do and the policy learnt from an incomplete reward function; (ii) Humans can follow a different reward model when working alone as compared to when collaborating with others. Since we have adapted the reward function learnt from a single-agent scenario to a two-agent scenario, it is possible for the optimal-human policy to deviate from the habit-following human.

```
INPUT: State space $S$ , Action space $A$, Data $D$
Initialize $\pi^h$ and $\pi^r$ uniformly
$Q \leftarrow 0; \eta \leftarrow 0.5$
while burn-in period do
    Sample $s \sim Bern(\eta); s \in \{d, \epsilon\}$
    Update $\pi^h_s$
    $Q, R \leftarrow$RUN-EPISODE($Q, \pi^h_s, \pi^r$)
end while
Update $\eta$
loop
    Update $\pi^h_a$ using Eq. 5.3
    $Q, R \leftarrow$RUN-EPISODE($Q, \pi^h_a, \pi^r$)
end loop
return Robot's Policy $\pi_r$
```

Algorithm 3: Learn Robot Policy

and try to perform the action that is beneficial to both for completing the activity. We model this behavior by computing the probability of the human choosing one of the above two behaviors and define the human policy as

$$\pi^h_a(\mathbf{s}, a) = \eta * \pi^h_d(\mathbf{s}, a) + (1 - \eta) * \pi^h_\epsilon(\mathbf{s}, a); \quad \forall \mathbf{s}, a \tag{5.3}$$

where η denotes the probability of the human to follow habits.

During test time, when the robot is collaborating with the human on a new task, it should choose actions from the policy which is learnt with matching human behavior. One approach is to assume that the opponent type is known and fixed, and use the policy learnt with that type when executing the activities.

Some works try to identify the opponents strategy on the fly and adapt accordingly. Such an approach requires the robot to perform activities with humans for long time durations, which is not very practical in most scenarios. In contrast to these approaches, we present an algorithm (Algorithm 3) which adaptively selects the human's actions for exploration during the learning phase.

We need to estimate the Q-values along with the value of η, which is the probability with which the human follows his habits. This probability depends on the human's familiarity of the environment as well as the cost of deviating from the optimal policy. Therefore, we model this probability as a function of the joint reward obtained when the human follows one of the two extreme behaviors, *habit-following* and *ϵ-optimal*, throughout the activity. Therefore, during an initial burn-in period, we sample the behavior uniformly and fix the behavior throughout an episode and learn the Q-values. We maintain a score for each of the behaviors, denoted by w_d and w_ϵ for the *habit-following* and *ϵ-optimal* behaviors respectively. At the end of Q-learning episode, we compute the normalized joint reward and update the corresponding score value as in Eq. 5.4.

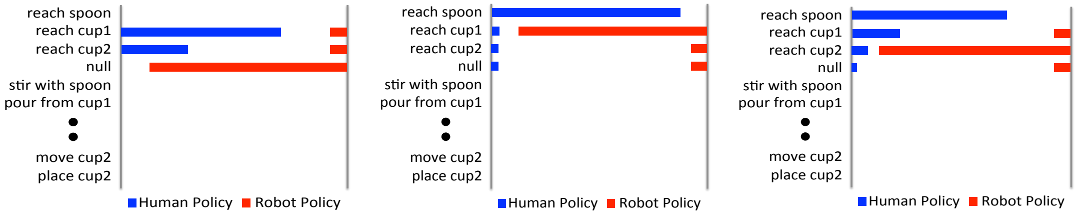
$$\hat{R} \leftarrow \frac{R - \frac{r_{min}}{1-\gamma}}{\frac{r_{max}}{1-\gamma} - \frac{r_{min}}{1-\gamma}} \; ; \;\; w \leftarrow w + \alpha(\hat{R} - w) \tag{5.4}$$

where r_{max} and r_{min} are the maximum and minimum reward at any given time, respectively, γ is the discount factor and α is the learning rate. At the end of the burn-in period we compute the value of η as:

$$\eta \leftarrow \frac{e^{(w_d)}}{e^{(w_d)} + e^{(w_\epsilon)}} \tag{5.5}$$

We then continue learning the Q-values for the adaptive human by updating the human policy according to Eq. 5.3 using the estimated value of η.

Effect of the human behavior on the learned robot policies. Fig. 5.4 illustrates



(a) Habit-following Human. (b) ϵ-optimal Human. (c) Adaptive Human.

Figure 5.4: **Illustration of policies learnt with different human behaviors:** Each figure shows the learnt probability distributions of the various possible actions at the start state of the *following recipe* activity. Blue and red bars represent the probability of choosing the corresponding actions by the human and robot respectively.

different human and robot policies corresponding to the different possible human behaviors we consider. Here, we consider an environment in which there are two cups, a bowl and a spoon, and the robot can only reach the cups and the bowl where as the human can reach all objects. The goal of the activity is to follow a recipe involving transfer of the ingredients from the cups to the bowl and mix them with the spoon. In the training videos, at the beginning of the activity, the human reaches the first cup more often than the second cup as shown by the policy for the start state in Fig. 5.4-(a). The corresponding learned robot policy is to not do any action as reaching for a cup could result in a conflict.

Instead of following habits, if the human tries to optimizes for the joint reward, he would reach for the spoon and let the robot use the cups, which allows them to perform the activity together and complete it sooner. Fig. 5.4-(b) shows the policies corresponding to the ϵ-*optimal* human behavior. Given this environment, following habits turns out to be less rewarding and therefore a human would try to adapt more. This is reflected in the estimated value of η which is low for this particular scenario. Fig. 5.4-(c) shows our learnt *adaptive human* policy and the corresponding robot policy. Note that even small changes in human behavior can result in significant changes in the robot's actions.

5.4 Experiments

We test our proposed algorithm and other baseline methods for generating collaborative plans for several household activities. We evaluate the learnt robot policies on both an activity dataset as well as in interaction with real humans. In this section we describe the data, experimental setup and the results.

Data: In order to evaluate our affordance and anticipatory planning models we expanded the CAD-120 dataset to CAD-250 dataset, which has 130 additional RGB-D activity videos which contain more interesting object affordances and activities which allow human-robot collaboration. The sub-activities in the CAD-250 dataset include {*moving, stirring, pouring, drinking, cutting, eating, cleaning, reading, answering phone, wearing, exercising, hammering, measuring*} and the corresponding affordances are {*movable, stirrable, pourable, pourto, drinkable, cuttable, edible, cleanable, cleaner, readable, hearable, wearable, exercisable, hammer, hammerable, measurer, measurable*}.

We evaluated our planning algorithm on 60 RGB-D videos from the CAD-250 dataset which allow for collaboration. These activities include *two recipe making tasks*, *setting dinner table*, *cleaning house*, and *loading shelves*. These activities were performed by four subjects where each high-level activity is performed three times by each subject in a different environment. For each activity video, we labeled the sub-activities and the object affordances.

Baselines: We compare our method against the following baselines:

Table 5.1: **Collaborative Planning Evaluation.** Metrics computed for the collaborative plans generated on our RGB-D dataset.

Model	% time saving					% conflicts
	Recipe	Setting	Cleaning	Loading	Overall	
Human Expert Plans	36.8	53.1	16.4	42.4	37.2	0
Chance	3.3	10.5	-33.1	23.7	1.1	3.7
Mental-model MDP[144]	-2.6	30.4	-5.1	32.3	13.8	6.4
Our Model – ϵ-optimal human	27.5	45.6	18.3	30.8	31.2	13.5
Our Model – habit following human	28.4	48.1	18.6	41.4	33.4	11.9
Our Model – adaptive human	**32.8**	**48.5**	**22.9**	**41.9**	**36.5**	13.7

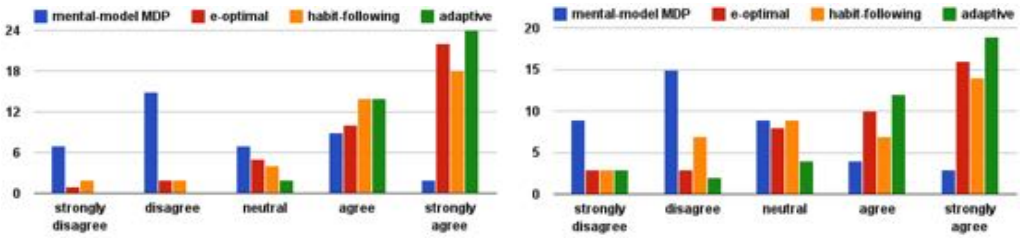
- **Human Expert**: A human expert manually designed collaborative plans for each activity in the dataset.
- **Chance**: This algorithm chooses actions uniformly at random from the set of possible actions.
- **Mental-model MDP [144]**: We follow Nikolaidis et al. and define a MDP to model the robot's mental model [144]. In this approach, the human actions are incorporated into the state transition function and the policy specifies only the robot's actions. Therefore, we use the same state and action spaces and reward function as described in our approach with only one agent and compute the transition function from the state action sequences from the training data. Note that in our adaptation of [144] we fix the transition function learned from the data and do not perform any cross training iterations as the roles are fully exchangeable in our collaborative setting.

5.4.1 Evaluation on Data

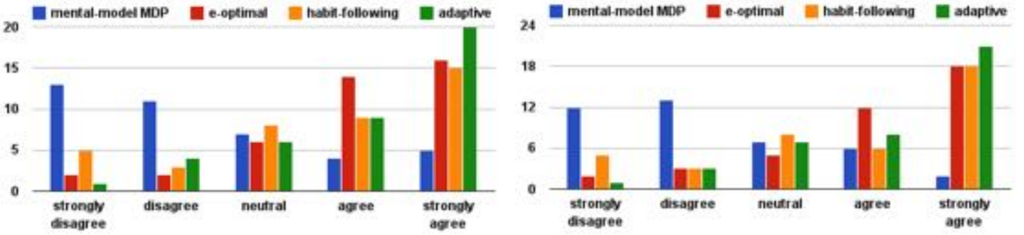
We evaluate the generated collaborative plans on the following two metrics: (i) Percentage time saving: The percentage of savings in time for task completion is computed as $\frac{n_h - n_c}{n_h} * 100$, where n_h denotes the number of time steps taken if only human performs the task and n_c denote the number of steps to task completion following the collaborative plan. (ii) Percentage conflicts: The percentage of time steps robot's chosen action conflicted with that of the human.

For each activity video in the dataset, we give the environment extracted from the first frame and the goal state as input to the planning algorithm. We perform leave-one-out cross-validation and use the rest of the activity videos for learning the task model and the robot policies as described in Section 5.2.1 and Section 5.3. The sequence of human actions are taken from the test video and executed together with the robot actions specified by the learnt robot policy. Table 5.1 shows the results averaged for each high-level activity as well as for all activities in the dataset.

As can be seen in Table 5.1, our algorithm allows for more collaboration between human and the robot, resulting in higher savings in the time required for task completion compared to the baseline algorithms. When the robot chooses actions uniformly at random (Chance baseline), it sometimes chooses action sequences that help in achieving the goal sooner, but can also perform undesirable actions requiring additional time to complete the activity. Therefore, on average it does not result in any savings in the execution time. These results show that modeling the human actions along with the contextually rich environments is very important for collaborative planning.



(a) The robot was collaborative and helped. (b) The robot did the right things.



(c) I am satisfied working with the robot. (d) I will work with this robot again.

Figure 5.5: **User Study Results:** The subjects collaborated with the robot on two different tasks in a total of eight different activities. They rated their experience based on four different criteria. The plots (a)-(d) show the comparison of the user ratings for four different robot policies – mental-model MDP [144], and our collaborative MDP trained with *ϵ-optimal human*, *habit-following human* and *adaptive human*.

5.4.2 User Study

Experiment setup. We performed an user study with five subjects to evaluate the learned robot policies. We considered two high-level activities – *setting table* and *making recipe*, and four different environments for each activity. The subjects were asked to work with the robot to complete the tasks in a simulator. We re-created the environments in OpenRAVE [24] and provided an interface to the subjects to select an action they wish to execute. At every time instant, the users were shown the current state of the environment, and were asked to choose an action. The robot also selects an action based on the current state using its learned policy. Both the human and robot actions are then executed in the simulator. After completing each task, the users were asked to rate the following statements on Likert scale from 1 (strongly disagree) to 5 (strongly agree).

(a) The robot was collaborative and helped in the activity.

(b) The robot did the right thing at the right time.

(c) I am satisfied working with the robot.

(d) I will work with this robot again in future.

In this study, we compared the robot policies generated by the mental-model MDP [144] and our method learned with the three human behaviors. Therefore, every user performed each task four times, resulting in a total of 640 ratings.

Results. Fig. 5.5 shows the comparison of the user ratings for the four different robot policies on the four criteria mentioned above. Users rated the robot trained with our collaborative MDP model significantly higher ($p < 0.001$) than the robot using the mental-model MDP on all four criteria. For the robot policies learnt with our collaborative MDP using different human behaviors, when asked if they thought the robot did the right action at the right time, the users rated the robot trained with *adaptive human* higher than others ($p = 0.08$). For other criteria, there is no significant difference in the user ratings, however, as can be seen in Fig. 5.5, there is a slight preference for the robot trained with *adaptive human*. Table 5.2 summarizes the two metrics on the collaborative plans generated in the user study. The users completed the tasks faster when working with robot trained with the *adaptive human* as compared to others.

Table 5.2: **Collaborative Planning Evaluation for User Study.** Metrics computed for the collaborative plans generated when working with humans during the user study.

Model	% time saving			% conflicts
	Recipe	Setting	Overall	
Mental-model MDP[144]	-0.9	13.9	6.5	2.7
Our Model – ϵ-optimal human	34.3	46.5	40.4	4.6
Our Model – habit following human	16.5	48.9	32.7	4.4
Our Model – adaptive human	**38.2**	**52.7**	**45.5**	4.6

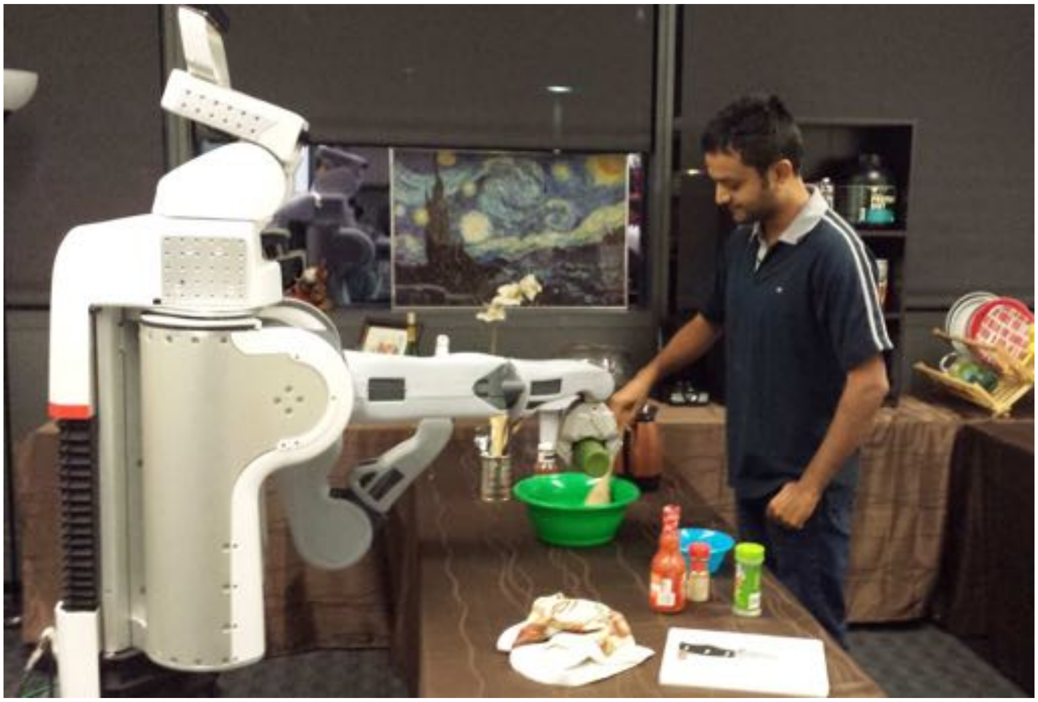


Figure 5.6: Robot and human collaborating to prepare a recipe.

5.4.3 Robot Experiment

We have also used the learned robot policy on our Kodiak (PR2) robot to work with a human on a *following recipe* task. Fig. 5.6 shows the robot collaborating with human to prepare a recipe, where the robot is executing the pour action as the human is stirring, based on its learnt policy. Videos showing the human and robot collaborating are available at: `http://pr.cs.cornell.edu/collaborativeplanning/`.

5.4.4 Discussion

We discuss the results of the evaluation on our dataset as well as the user study in the light of the following questions.

What is the advantage of our collaborative MDP model over a single-agent MDP model? The mental-model MDP doesn't model the human explicitly as an agent but incorporates the effect of human actions in to the state transition probabilities. We observe a large variation in the performance of the mental-model MDP [144] baseline across the types of activities – it performs well on the setting table and loading shelf tasks, but takes longer to complete the recipe and cleaning tasks. It is interesting to note that the setting table and loading shelf tasks have a smaller action space as compared to the cleaning and recipe tasks. Therefore, given limited training data, the mental-model MDP is sensitive to the estimated state transition probabilities and fails when the action space is large. On the other hand, our collaborative MDP approach, which models the actions of human explicitly, overcomes this problem and performs significantly better on all tasks.

How important is modeling human behavior for collaboration? As humans tend to have specific preferences for executing tasks, the robot policy learnt with *habit-following human* strategy, which incorporates these preferences into planning, achieves an additional 2.2% saving in time compared to the *ϵ-optimal hu-*

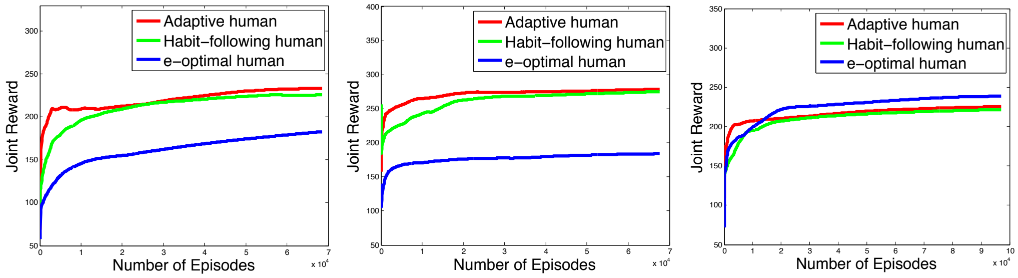


Figure 5.7: **Joint reward received during *Q*-learning.** Plots showing the joint reward as a function of the number of *Q*-learning episodes for three test environments.

man. However, when tested with new humans, whose habits were never seen in the training data, the robot policy learnt with *ϵ-optimal* human performs better (see Table 5.2). Modeling human as an adaptive agent always performs better and results in more collaboration – increasing the savings in the task completion time by 3.1% when working with a familiar human (seen in the training data) and by 5.1% when working with a new human.

We also study how the joint reward evolves over the *Q*-learning episodes during training. At the end of each episode, we use the learned robot policy to perform the activity with a human following the policy π_d^h corresponding to the test environment. Fig. 5.7 shows the joint reward received by the human and the robot as a function of the number of training episodes. We see that the policy learnt with the *adaptive human* converges to the highest joint reward much faster in most cases. However, incorrect estimation of adaptation probability causes the *adaptive human* to perform sub-optimally in some cases (Fig. 5.7-right).

How often does the robot conflict with human? The savings in task completion time increase as a result of the robot's increased participation in the task. This also leads to an increase in the percentage of conflicts between the robot's and the human's actions. However, our model learnt with the *habit-following* *human* strategy reduces the percentage of conflicts compared to other baselines as it models the human's preferences. The percentage of conflicts again increase in case of *adaptive human* due to increased participation of the robot in the activity. When a conflict occurs, the preference is given to the human and the robot stops executing the action and chooses a new action in the next time step. In our current model we prefer plans with increased collaboration and do not penalize conflicts heavily. However, it is possible to modify the reward function to incorporate this, and we plan to explore this in future work.

5.5 Conclusion

In this Chapter, we considered the problem of anticipatory planning for human robot teams, for enabling robots to work along side humans in contextually rich environments to accomplish complex tasks. We proposed a two agent collaborative MDP model and learn robot policies by taking into account the actions that can be performed by the human. We represented the contextually rich environments in terms of the object affordances and learn the activity model from RGB-D videos of a human performing the activities. We used this learned task model in a distributed Q-learning algorithm to learn the robot policy for a given new environment. We model the different possible human behaviors – taking the ϵ-optimal action according to the MDP model, taking actions based on past habits, and taking appropriate actions by adapting to the environment and robot actions. We tested our collaborative MDP model on the activity dataset as well as while directly interacting with humans in a user study. We show that explicitly modeling the human actions in the MDP formulation results in learning better robot policies. We also showed that changes in the human behavior can lead to significant changes in desirable robot actions. Therefore, modeling human behavior is essential for collaborative planning.

CHAPTER 6

CONCLUSION

One of the key goals of machine perception is understanding people and their surroundings. The various challenges for achieving this goal include reasoning about the spatiotemporal aspects, modeling human intent, robustness to noisy and incomplete data, and scaling the algorithms to large number of classes and large datasets, etc.

In this dissertation we described new learning algorithms that address these challenges. Our learning algorithms leverage the 3D structure as well as the the spatial and temporal interactions between objects and humans in order to model the human environments. We proposed a conditional random field model that naturally captures the spatial and temporal relationships between the objects and the humans in the environment.

Conditional random field models capture the relations between labels of all pairs of nodes connected by an edge in the graph. In these models the number of parameters grow very quickly with the increase in the number of labels. For labeling tasks with a large number of classes, the model's parsimony becomes important. One of the key ideas in this work was to design appropriate sparse clique potentials leading to reduced number of model parameters.

Estimating the spatio-temporal structure of activities is quite challenging. Humans are very dexterous in using objects and often overlap their activities in order to achieve efficiency. This results in an ambiguity in the temporal segmentation and defining an exact boundary between sub-activities is often hard even for humans. We addressed this problem by developing an algorithm that considers labelings over alternate possible segmentations as latent variables in our learning algorithm. The key idea is to generate approximate initial segmentations that are close to the ground-truth segmentation using our CRF model with only additive features which lends to efficient dynamic programming.

For anticipation, we represented each possible future using an anticipatory temporal conditional random field (ATCRF). We augmented the graph structure of the CRF corresponding to the past with anticipated future nodes using the object affordance and motion trajectories learnt form the data. We then considered each ATCRF as a particle and represent the distribution over the potential futures using a set of particles. Our models admit efficient approximate inference and we achieve efficient sampling of graph structures through use of affordance representations.

We have shown how these algorithms have enabled some very challenging tasks. They have significantly extended the state-of-the-art in 3D scene and human activity understanding. They have enabled a PR2 robot to assist in daily activities by understanding what the people are doing. The ability to anticipate what a person might do next allowed the robot to perform reactive responses. For example, the robot does not pour a drink when a person is reaching for a cup to move it. Finally, by taking into account the human habits and behavior, the robot was able to work alongside the person and collaborate on various complex tasks such as following a recipe.

BIBLIOGRAPHY

[1] Openni. `https://github.com/OpenNI/OpenNI`.

[2] J. K. Aggarwal and M. S. Ryoo. Human activity analysis: A review. *ACM Comp Surveys (CSUR)*, 2011.

[3] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. Learning the semantics of object-action relations by observation. *IJRR*, 30 (10), 2011.

[4] E.E. Aksoy, A. Abramov, F. Worgotter, and B. Dellen. Categorizing object-action relations from semantic scene graphs. In *ICRA*, 2010.

[5] A. Aldoma, F. Tombari, and M. Vincze. Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes. In *ICRA*, 2012.

[6] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Contextually guided semantic labeling and search for 3d point clouds. *IJRR*, 2012.

[7] D. Anguelov, B. Taskar, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3d scan data. In *CVPR*, 2005.

[8] Dhruv Batra and Ashutosh Saxena. Learning the right model: Efficient max-margin learning in laplacian crfs. In *CVPR*, 2012.

[9] Jos Bento, Nate Derbinsky, Javier Alonso-Mora, and Jonathan S. Yedidia. A message-passing algorithm for multi-agent trajectory planning. In *NIPS*, 2013.

[10] Mario Bollini, Stefanie Tellex, Tyler Thompson, Nicholas Roy, and Daniela Rus. Interpreting and executing recipes with a cooking robot. In *ISER*, 2012.

[11] E. Boros and P.L. Hammer. Pseudo-boolean optimization. *Discrete Applied Mathematics*, 123(1-3):155–225, 2002.

[12] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE SMC, Part C*, 2008.

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

[14] David Carmel and Shaul Markovitch. Opponent modeling in multi-agent systems. In *Adaption And Learning In Multi-Agent Systems*, volume 1042. 1996.

[15] 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 *CVPR*, 2009.

[16] Changhyun Choi and Henrik I Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *IJRR*, 31(4):498–519, 2012.

[17] Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The moped framework: Object recognition and pose estimation for manipulation. *IJRR*, 2011.

[18] Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The MOPED framework: Object Recognition and Pose Estimation for Manipulation. *IJRR*, 2011.

[19] Alvaro Collet Romea, Siddhartha Srinivasa, and Martial Hebert. Structure discovery in multi-modal data : a region-based approach. In *ICRA*, 2011.

[20] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, 2004.

[21] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[22] Vincent Delaitre, David Fouhey, Ivan Laptev, Josef Sivic, Abhinav Gupta, and Alexei Efros. Scene semantics from long-term observation of people. In *ECCV*, 2012.

[23] J. Denzler and C.M. Brown. Information theoretic sensor data selection for active object recognition and state estimation. *IEEE PAMI*, 24(2):145 –157, feb 2002. ISSN 0162-8828. doi: 10.1109/34.982896.

[24] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, CMU, RI, August 2010.

[25] Rosen Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010. URL `http://www.programmingvision.com/rosen_diankov_thesis.pdf`.

[26] Cristian Dima, Nicolas Vandapel, and Martial Hebert. Classifier fusion for outdoor obstacle detection. In *ICRA*, 2004.

[27] S.K. Divvala, D. Hoiem, J.H. Hays, A.A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, 2009.

[28] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-blackwellised particle filtering for dynamic bayesian networks. In *UAI*, 2000.

[29] B. Douillard, D. Fox, F. Ramos, and H. Durrant-Whyte. Classification and semantic mapping of urban environments. *IJRR*, 30(1):5–32, 2011.

[30] A. Dragan and S. Srinivasa. Formalizing assistive teleoperation. In *RSS*, 2012.

[31] O. Duchenne, I. Laptev, J. Sivic, F. Bach, and J. Ponce. Automatic annotation of human actions in video. In *ICCV*, 2009.

[32] J. J. Faraway, M. P. Reed, and J. Wang. Modelling three-dimensional trajectories by using bezier curves with application to hand motion. *JRSS Series C*, 56:571–585, 2007.

[33] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008.

[34] P. F. Felzenszwalb and D.P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004.

[35] Thomas Finley and Thorsten Joachims. Training structural svms when exact inference is intractable. In *ICML*, 2008.

[36] D. Fox. Kld-sampling: Adaptive particle filters. In *NIPS*, 2001.

[37] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky. Nonparametric Bayesian learning of switching linear dynamical systems. In *NIPS 21*, 2009.

[38] E.B. Fox, E.B. Sudderth, M.I. Jordan, and A.S. Willsky. Bayesian Nonparametric Inference of Switching Dynamic Linear Models. *IEEE Transactions on Signal Processing*, 59(4), 2011.

[39] Y. Fu, T. M. Hospedales, T. Xiang, and S. Gong. Attribute learning for understanding unstructured social activity. In *ECCV*, 2012.

[40] A. Gaidon, Z. Harchaoui, and C. Schmid. Actom sequence models for efficient action detection. In *CVPR*, 2011.

[41] J. Gall, A. Fossati, and L. van Gool. Functional categorization of objects using real-time markerless motion capture. In *CVPR*, 2011.

[42] J. J. Gibson. *The Ecological Approach to Visual Perception*. 1979.

[43] J.J. Gibson. *The Ecological Approach to Visual Perception*. Houghton Mifflin, 1979.

[44] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *ICCV*, 2009.

[45] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *ICCV*, 2003.

[46] Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y. Ng, and Daphne Koller. Integrating visual and range data for robotic object detection. In *ECCV workshop Multi-camera Multi-modal (M2SFA2)*, 2008.

[47] H. Grabner, J. Gall, and L. Van Gool. What makes a chair a chair? In *CVPR*, 2011.

[48] E. Guizzo and E. Ackerman. The rise of the robot worker. *Spectrum, IEEE*, 49(10):34 –41, October 2012.

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

[50] Abhinav Gupta, Scott Satkin, Alexei Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011.

[51] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Trans. SP*, 2002.

[52] P.L. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0–1 optimization. *Mathematical Programming*, 28(2):121–155, 1984.

[53] Z. Harchaoui, F. Bach, and E. Moulines. Kernel change-point analysis. In *NIPS*, 2008.

[54] Kelsey P Hawkins, Phillip M Grice, Tiffany L Chen, Chih-Hung King, and Charles C Kemp. Assistive mobile manipulation for self-care tasks around the head. In *Computational Intelligence in Robotic Rehabilitation and Assistive Technologies (CIR2AT), 2014 IEEE Symposium on*, pages 16–25. IEEE, 2014.

[55] V. Hedau, D. Hoiem, and D.A. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, 2010.

[56] G. Heitz, S. Gould, A. Saxena, and D. Koller. Cascaded classification models: Combining models for holistic scene understanding. In *NIPS*, 2008.

[57] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *ECCV*, 2008.

[58] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using kinect-style depth cameras for dense 3d modeling of indoor environments. *IJRR*, 31(5):647–663, 2012.

[59] T. Hermans, J. M. Rehg, and A. Bobick. Affordance prediction via learned object attributes. In *ICRA Workshop on SPME*, 2011.

[60] T. Hermans, J. M. Rehg, and A. Bobick. Decoupling behavior, perception, and control for autonomous learning of affordances. In *ICRA*, 2013.

[61] R. Hess and A. Fern. Discriminatively trained particle filters for complex multi-object tracking. In *CVPR*, 2009.

[62] M. Hoai and F. De la Torre. Max-margin early event detectors. In *CVPR*, 2012.

[63] M. Hoai and F. De la Torre. Maximum margin temporal clustering. In *AISTATS*, 2012.

[64] M. Hoai, Z. Lan, and F. De la Torre. Joint segmentation and classification of human actions in video. In *CVPR*, 2011.

[65] Minh Hoai and Fernando De la Torre. Maximum margin temporal clustering. In *Proceedings of International Conference on Artificial Intelligence and Statistics*, 2012.

[66] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *CVPR*, 2006.

[67] S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. In *ICCV*, 2003.

[68] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *ICML*, 1998.

[69] P. Jensfelt, D. Austin, O. Wijk, and M. Andersson. Feature based condensation for mobile robot localization. In *ICRA*, 2000.

[70] Z. Jia, A. Gallagher, A. Saxena, and T. Chen. 3d-based reasoning with blocks, support, and stability. In *CVPR*, 2013.

[71] Zhaoyin Jia, Ashutosh Saxena, and Tsuhan Chen. Robotic object detection: Learning to improve the classifiers using sparse graphs for path planning. In *IJCAI*, pages 2072–2078, 2011.

[72] Zhaoyin Jia, Ashutosh Saxena, and Tsuhan Chen. Robotic object detection: Learning to improve the classifiers using sparse graphs for path planning. In *IJCAI*, 2011.

[73] Y. Jiang and A. Saxena. Hallucinating humans for learning robotic placement of objects. In *ISER*, 2012.

[74] Y. Jiang, Z. Li, and S. Chang. Modeling scene and object contexts for human action retrieval with few examples. *IEEE Trans Circuits & Sys Video Tech*, 2011.

[75] Y. Jiang, M. Lim, and A. Saxena. Learning object arrangements in 3d scenes using human context. In *ICML*, 2012.

[76] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *IJRR*, 2012.

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

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

[79] Yun Jiang, Stephen Moseson, and Ashutosh Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *ICRA*, 2011.

[80] Yun Jiang, Marcus Lim, Changxi Zheng, and Ashutosh Saxena. Learning to place new objects in a scene. *IJRR*, 2012.

[81] T. Joachims, T. Finley, and C.N.J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009. ISSN 0885-6125.

[82] Daniel Kahneman. *Thinking, fast and slow*. 2011.

[83] D. Katz, A. Venkatraman, M. Kazemi, J. A. Bagnell, and A. Stentz. Perceiving, learning, and exploiting object affordances for autonomous pile manipulation. In *RSS*, 2013.

[84] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *ICCV*, October 2007.

[85] Z. Khan, T. Balch, and F. Dellaert. Mcmc-based particle filtering for tracking a variable number of interacting targets. *IEEE PAMI*, 2005.

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

[87] H. Kjellström, J. Romero, and D. Kragic. Visual object-action recognition: Inferring object affordances from human demonstration. *CVIU*, 115(1): 81–90, 2011.

[88] S. Koenig, P. Keskinocak, and C. A Tovey. Progress on agent coordination with cooperative auctions. In *AAAI*, 2010.

[89] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[90] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto. Robot learning from demonstration by constructing skill trees. *IJRR*, 31, 2012.

[91] H. S. Koppula and A. Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *ICML*, 2013.

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

[93] H. S. Koppula and A. Saxena. Physically grounded spatio-temporal object affordances. In *ECCV*, 2014.

[94] H. S. Koppula and A. Saxena. Anticipatory planning for human-robot teams. In *ISER*, 2015.

[95] H. S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. *IEEE PAMI*, 2015.

[96] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011.

[97] H. S. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 2013.

[98] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with EM-based reinforcement learning. In *IROS*, 2010.

[99] Michael Krainin, Peter Henry, Xiaofeng Ren, and Dieter Fox. Manipulator and object tracking for in-hand 3d object modeling. *IJRR*, 30(11):1311–1327, 2011.

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

[101] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[102] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *ICRA*, 2011.

[103] K. Lai, L. Bo, X. Ren, and D. Fox. Sparse distance learning for object recognition combining rgb and depth information. In *ICRA*, 2011.

[104] Jean-Franois Lalonde, Nicolas Vandapel, Daniel F. Huber, and Martial Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *J. Field Robotics*, pages 839–861, 2006.

[105] Catherine Laporte and Tal Arbel. Efficient discriminant viewpoint selection for active bayesian recognition. *IJCV*, 68:267–287, 2006. ISSN 0920-5691. URL `http://dx.doi.org/10.1007/s11263-005-4436-9`.

[106] Catherine Laporte, Rupert Brooks, and Tal Arbel. A fast discriminant approach to active object recognition and pose estimation. In *ICPR*, 2004.

[107] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[108] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008.

[109] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *ICML*, 2000.

[110] B. Laxton, L. Jongwoo, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *CVPR*, 2007.

[111] David Changsoo Lee, Abhinav Gupta, Martial Hebert, and Takeo Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010.

[112] Bastian Leibe, Nico Cornelis, Kurt Cornelis, and Luc Van Gool. Dynamic 3d scene analysis from a moving vehicle. In *CVPR*, 2007.

[113] C. Li, A. Saxena, and T. Chen. θ-mrf: Capturing spatial and semantic structure in the parameters for scene understanding. In *NIPS*, 2011.

[114] C. Li, A. Kowdle, A. Saxena, and T. Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. *IEEE PAMI*, 34(7):1394–1408, 2012.

[115] W.Q. Li, Z.Y. Zhang, and Z.C. Liu. Action recognition based on a bag of 3d points. In *Workshop on CVPR for Human Communicative Behavior Analysis*, 2010.

[116] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *ICML*, 1994.

[117] M. L. Littman. Friend-or-foe q-learning in general-sum games. In *ICML*, 2001.

[118] J. Liu, J. Luo, and M. Shah. Recognizing realistic actions from videos "in the wild". In *CVPR*, 2009.

[119] J. Liu, B. Kuipers, and S. Savarese. Recognizing human actions by attributes. In *CVPR*, 2011.

[120] M. Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE SMC, Part B,*, 2005.

[121] C. Lorken and J. Hertzberg. Grounding planning operators by affordances. In *Int'l conf Cog Sys*, 2008.

[122] D. L. Ly, A. Saxena, and H. Lipson. Co-evolutionary predictors for kinematic pose inference from rgbd images. In *GECCO*, 2012.

[123] Daniel Ly, Ashutosh Saxena, and Hod Lipson. Co-evolutionary predictors for kinematic pose inference from rgbd images. In *GECCO*, 2012.

[124] Jeremy Ma, Timothy H Chung, and Joel Burdick. A probabilistic framework for object search with 6-dof pose estimation. *IJRR*, 30(10):1209–1228, 2011. doi: 10.1177/0278364911410090.

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

[126] J. Mainprice, E. A. Sisbot, L. Jaillet, J. Cortés, R. Alami, and T. Siméon. Planning human-aware motions using a sampling-based costmap planner. In *ICRA*, 2011.

[127] S. Maji, L. Bourdev, and J. Malik. Action recognition from a distributed representation of pose and appearance. In *CVPR*, 2011.

[128] P. Matikainen, R. Sukthankar, and M. Hebert. Model recommendation for action recognition. In *CVPR*, 2012.

[129] D. Meger, A. Gupta, and J.J. Little. Viewpoint detection models for sequential embodied object category recognition. In *ICRA*, 2010.

[130] E. Meisner, V. Isler, and J. Trinkle. Controller design for human-robot interaction. *Autonomous Robots*, 24(2), 2008.

[131] Stephen Miller, Jur van den Berg, Mario Fritz, Trevor Darrell, Ken Goldberg, and Pieter Abbeel. A geometric approach to robotic laundry folding. *IJRR*, 2011.

[132] J.-K. Min and S.-B. Cho. Activity recognition based on wearable sensors using selection/fusion hybrid ensemble. In *SMC*, 2011.

[133] B. Moldovan, M. van Otterlo, P. Moreno, J. Santos-Victor, and L. De Raedt. Statistical relational learning of object affordances for robotic manipulation. In *Latest Adv Inductive Logic Prog.*, 2012.

[134] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *AAAI*, 2002.

[135] M. Montemerlo, S. Thrun, and W. Whittaker. Conditional ficle filters for simultaneous mobile robot localization and people-tracking. In *ICRA*, 2002.

[136] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor. Learning object affordances: From sensory–motor coordination to imitation. *Trans Robotics*, 2008.

[137] Daniel Munoz, Nicolas Vandapel, and Martial Hebert. Onboard contextual classification of 3-d point clouds with learned high-order markov random fields. In *ICRA*, 2009.

[138] K. P. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. *NIPS*, 2003.

[139] P. Natarajan and R. Nevatia. Coupled hidden semi markov models for activity recognition. In *WMVC*, 2007.

[140] M. H. Nguyen, L. Torresani, F. De la Torre, and C. Rother. Weakly supervised discriminative localization and classification: a joint learning process. In *ICCV*, 2009.

[141] B. Ni, G. Wang, and P. Moulin. Rgbd-hudaact: A color-depth video database for human daily activity recognition. In *ICCV Workshop on CDC4CV*, 2011.

[142] J. Niebles, C. Chen, and L. Fei-fei. Modeling temporal structure of decomposable motion segments for activity classification. In *ECCV*, 2010.

[143] S. Nikolaidis and J. Shah. Human-robot teaming using shared mental models. In *HRI, Workshop on Human-Agent-Robot Teamwork*, 2012.

[144] S. Nikolaidis and J. Shah. Human-robot cross-training: Computational formulation, modeling and evaluation of a human team training strategy. In *HRI*, 2013.

[145] S. Nikolaidis and J. Shah. Human-robot cross-training: computational formulation, modeling and evaluation of a human team training strategy. In *HRI*, 2013.

[146] S. Oh, J.M. Rehg, T. Balch, and F. Dellaert. Learning and inferring motion patterns using parametric segmental switching linear dynamic systems. *IJCV*, 2008.

[147] Anand Panangadan, Maja J. Mataric, and Gaurav S. Sukhatme. Tracking and modeling of human activity using laser rangefinders. *IJSR*, 2010.

[148] A. K. Pandey and R. Alami. Mightability maps: A perceptual level decisional framework for co-operative and competitive human-robot interaction. In *IROS*, 2010.

[149] A. K. Pandey and R. Alami. Taskability graph: Towards analyzing effort based agent-agent affordances. In *RO-MAN*, sept. 2012.

[150] O. Pele and M. Werman. A linear time histogram metric for improved sift matching. In *ECCV*, 2008.

[151] Roland Perko and Ales Leonardis. A framework for visual-context-aware object detection in still images. *Computer Vision and Image Understanding*, 114:700–711, June 2010. ISSN 1077-3142.

[152] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012.

[153] R. Powers and Y. Shoham. New criteria and a new algorithm for learning in multi-agent systems. In *NIPS*, 2004.

[154] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[155] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden-state conditional random fields. *IEEE PAMI*, 2007.

[156] Morgan Quigley, Siddharth Batra, Stephen Gould, Ellen Klingbeil, Quoc V. Le, Ashley Wellman, and Andrew Y. Ng. High-accuracy 3d sensing for mobile manipulation: Improving object detection and door opening. In *ICRA*, 2009.

[157] Ulrich Reiser, Theo Jacobs, Georg Arbeiter, Christopher Parlitz, and Kerstin Dautenhahn. Care-o-bot3 - vision of a robot butler. In *Your Virtual Butler*, Lecture Notes in Computer Science, pages 97–116. 2013.

[158] B. Ridge, D. Skočaj, and A. Leonardis. Unsupervised learning of basic object affordances from object properties. In *CVWW*, 2009.

[159] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *CVPR*, 2012.

[160] M. Rohrbach, M. Regneri, Mykhaylo A., S. Amin, M. Pinkal, and B. Schiele. Script data for attribute-based recognition of composite activities. In *ECCV*, 2012.

[161] Benjamin Rosman and Subramanian Ramamoorthy. Learning spatial relationships between objects. *IJRR*, 30(11):1328–1342, 2011.

[162] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary mrfs via extended roof duality. In *CVPR*, 2007.

[163] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. Towards 3d point cloud based object maps for household environments. *Robot. Auton. Syst.*, 56:927–941, 2008.

[164] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *IROS*, 2009.

[165] R.B. Rusu, G. Bradski, R. Thibaux, and J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 2010.

[166] M.S. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011.

[167] S. Sadanand and J.J. Corso. Action bank: A high-level representation of activity in video. In *CVPR*, 2012.

[168] S. Sarawagi and W. W. Cohen. Semi-markov conditional random fields for information extraction. In *NIPS*, 2004.

[169] S. Satkin and M. Hebert. Modeling the temporal extent of actions. In *ECCV*, 2010.

[170] A. Saxena, S.H. Chung, and A. Ng. Learning depth from single monocular images. In *NIPS*, 2005.

[171] A. Saxena, S.H. Chung, and A.Y. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 76(1):53–69, 2008.

[172] A. Saxena, J. Driemeyer, and A.Y. Ng. Robotic grasping of novel objects using vision. *IJRR*, 2008.

[173] Ashutosh Saxena, Min Sun, and Andrew Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE PAMI*, 31(5):824–840, 2009.

[174] D. Schulz, D. Fox, and J. Hightower. People tracking with anonymous and id-sensors using rao-blackwellised particle filters. In *IJCAI*, 2003.

[175] R. Shapovalov and A. Velizhev. Cutting-plane training of non-associative markov network for 3d point cloud segmentation. In *3DIMPVT*, 2011.

[176] Roman Shapovalov, Alexander Velizhev, and Olga Barinova. Non-associative markov networks for 3d point cloud classification. In *ISPRS Commission III symposium - PCV 2010*, 2010.

[177] Q. Shi, L. Wang, L. Cheng, and A. Smola. Human action segmentation and recognition using discriminative semi-markov models. *IJCV*, 2011.

[178] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[179] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *IEEE TPAMI*, 2012.

[180] T. Simon, M. H. Nguyen, F. De la Torre, and J. F. Cohn. Action unit detection with segment-based svms. In *CVPR*, 2010.

[181] E. A. Sisbot, L. F. Marin, and R. Alami. Spatial reasoning for human robot interaction. In *IROS*, 2007.

[182] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Conditional models for contextual human motion recognition. In *ICCV*, 2005.

[183] K. W. Strabala, M. K. Lee, A. D. Dragan, J. L. Forlizzi, S. .S. Srinivasa, M. Cakmak, and V. Micelli. Towards seamless human-robot handovers. *JHRI*, 2013.

[184] J. Sun, J. L. Moore, A. Bobick, and J. M. Rehg. Learning visual object categories for robot affordance prediction. *IJRR*, 2009.

[185] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from rgbd images. In *ICRA*, 2012.

[186] Jae Y. Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Human activity detection from rgbd images. In *AAAI workshop on Pattern, Activity and Intent Recognition (PAIR)*, 2011.

[187] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgbd images. In *ICRA*, 2012.

[188] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother. A comparative study of energy minimization methods for markov random fields with smoothness-based priors. *IEEE PAMI*, pages 1068–1080, 2008.

[189] K. Tang, L Fei-Fei, and D. Koller. Learning latent temporal structure for complex event detection. In *CVPR*, 2012.

[190] B. Taskar, V. Chatalbashev, and D. Koller. Learning associative markov networks. In *ICML*, 2004.

[191] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *NIPS*, 2003.

[192] G. Tesauro. Extending q-learning to general adaptive multi-agent systems. In *NIPS*, 2004.

[193] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2):169–191, 2003.

[194] S. Tran and L. S. Davis. Event modeling and recognition using markov logic networks. In *ECCV*, 2008.

[195] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

[196] M. Turpin, N. Michael, and V. Kumar. Trajectory planning and assignment in multirobot systems. In *WAFR*, 2012.

[197] E. Ugur, E. Sachin, and E. Oztop. Affordance learning from range data for multi-step planning. In *Epirob*, 2009.

[198] Christopher Urmson, Joshua Anhalt, Hong Bae, J. Andrew (Drew) Bagnell, Christopher R. Baker, Robert E Bittner, Thomas Brown, M. N. Clark, Michael Darms, Daniel Demitrish, John M Dolan, David Duggins, David Ferguson , Tugrul Galatali, Christopher M Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Sascha Kolski, Maxim Likhachev , Bakhtiar Litkouhi, Alonzo Kelly , Matthew McNaughton,

Nick Miller, Jim Nickolaou, Kevin Peterson, Brian Pilnick, Ragunathan Rajkumar, Paul Rybski, Varsha Sadekar, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod M Snider, Joshua C Struble, Anthony (Tony) Stentz, Michael Taylor , William (Red) L. Whittaker, Ziv Wolkowicki, Wende Zhang, and Jason Ziglar. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):425–466, 2008.

[199] K. F. Uyanik, Y. Caliskan, A. K. Bozcuoglu, S. Kalkan O. Yuruten, and E. Sahin. Learning social affordances and using them for planning. In *CogSys*, 2013.

[200] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Mining actionlet ensemble for action recognition with depth cameras. In *CVPR*, 2012.

[201] Z. Wang, M. Deisenroth, H. B. Amor, D. Vogt, B. Scholkopf, and J. Peters. Probabilistic modeling of human movements for intention inference. In *RSS*, 2012.

[202] Z. Wang, Q. Shi, C. Shen, and A. van den Hengel. Bilinear programming for human activity recognition with unknown mrf graphs. In *CVPR*, 2013.

[203] T. Wongpiromsarn, A. Ulusoy, C. Belta, E. Frazzoli, and D. Rus. Incremental synthesis of control policies for heterogeneous multi-agent systems with linear temporal logic specifications. In *ICRA*, 2013.

[204] M. Wunder, M. L. Littman, and M. Babes. Classes of multiagent q-learning dynamics with epsilon-greedy exploration. In *ICML*, 2010.

[205] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *ICCV*, 2009.

[206] Z. Xing, J. Pei, G. Dong, and P. S. Yu. Mining Sequence Classifiers for Early Prediction. In *SIAM ICDM*, 2008.

[207] Xuehan Xiong and Daniel Huber. Using context to create semantic 3d models of indoor environments. In *BMVC*, 2010.

[208] Xuehan Xiong, Daniel Munoz, J. Andrew Bagnell, and Martial Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *ICRA*, 2011.

[209] X. Xuan and K. Murphy. Modeling changing dependency structure in multivariate time series. In *ICML*, 2007.

[210] W. Yang, Y. Wang, and G. Mori. Recognizing human actions from still images with latent poses. In *CVPR*, 2010.

[211] B. Yao and L. Fei-Fei. Modeling mutual context of object and human pose in human-object interaction activities. In *CVPR*, 2010.

[212] B. Yao, X. Jiang, A. Khosla, A.L. Lin, L.J. Guibas, and L. Fei-Fei. Action recognition by learning bases of action attributes and parts. In *ICCV*, 2011.

[213] C. Yu and T. Joachims. Learning structural svms with latent variables. In *ICML*, 2009.

[214] H. Zhang and L. E. Parker. 4-dimensional local spatio-temporal features for human activity recognition. In *IROS*, 2011.

[215] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury. Context-aware modeling and recognition of activities in video. In *CVPR*, 2013.

[216] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *IROS*, 2009.