HALLUCINATED HUMANS:
LEARNING LATENT FACTORS
TO MODEL 3D ENVIRONMENTS

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
Yun Jiang

August 2015

HALLUCINATED HUMANS:
LEARNING LATENT FACTORS
TO MODEL 3D ENVIRONMENTS

Yun Jiang, Ph.D.

Cornell University 2015

The ability to correctly reason about human environment is critical for personal robots. For example, if a robot is asked to tidy a room, it needs to detect object types, such as shoes and books, and then decides where to place them properly. Sometimes being able to anticipate human-environment interactions is also desirable. For example, the robot would not put any object on the chair if it understands that humans would sit on it.

The idea of modeling object-object relations has been widely leveraged in many scene understanding applications. For instance, the object found in front of a monitor is more likely to be a keyboard because of the high correlation of the two objects. However, as the objects are designed by humans and *for* human usage, when we reason about a human environment, we reason about it through an interplay between the environment, objects and humans. For example, the objects, monitor and keyboard, are strongly spatially correlated only because a human types on the keyboard while watching the monitor. The key idea of this thesis is to model environments not only through objects, but also through latent human poses and human-object interactions.

We start by designing a generic form of human-object interaction, also referred as 'object affordance'. Human-object relations can thus be quantified through a function of object affordance, human configuration and object con-

figuration. Given human poses and object affordances, we can capture the relations among humans, objects and the scene through Conditional Random Fields (CRFs). For scenarios where no humans present, our idea is to still leverage the human-object relations by hallucinating potential human poses.

In order to handle the large number of latent human poses and a large variety of their interactions with objects, we present Infinite Latent Conditional Random Field (ILCRF) that models a scene as a mixture of CRFs generated from Dirichlet processes. In each CRF, we model objects and object-object relations as existing nodes and edges, and hidden human poses and human-object relations as latent nodes and edges. ILCRF generatively models the distribution of different CRF structures over these latent nodes and edges.

We apply the model to the challenging applications of 3D scene labeling and robotic scene arrangement. In extensive experiments, we show that our model significantly outperforms the state-of-the-art results in both applications. We test our algorithm on a robot for arranging objects in a new scene using the two applications aforementioned. We further extend the idea of hallucinating static human poses to anticipating human activities. We also present learning-based grasping and placing approaches for low-level manipulation tasks in complimentary to the high-level scene understanding tasks.

BIOGRAPHICAL SKETCH

Yun Jiang was born in Changsha, a beautiful southern city in China which is known for hot food and even hotter summers. With both of her parents working in the computer science department of a university, she grew up with piles of dot-matrix printouts as her drawing papers. At that time, she had no idea those mysterious characters on the paper are what called programs. Neither did she anticipate that one day she would fall in love with it, just like her parents.

Not until her senior high, Yun started learning programming and soon became fascinated about algorithms. She enjoyed solving those programming puzzles and participated many programming contests ever since. When she was admitted by Shanghai Jiao Tong University, she did not think twice of majoring in computer science.

In her junior year, she joined a machine learning research lab and built a Naive-Bayes spam detector with other students. A project, though as basic as that, exposed to her the scientific research frontier and motivated her to pursue a higher degree. Since 2009, Yun has been studying machine learning for her doctorate degree in computer science at Cornell University.

This thesis is dedicated to my parents, who have been and constinue to be my support and inspiration.

ACKNOWLEDGEMENTS

First and foremost, I thank my advisor, Prof. Ashutosh Saxena, for his guidance and support throughout the years. He taught me every detail of academic research, from how to initiate an original idea, design convincing and exhaustive experiments, to how to present the story to different audience. He has always been patient and encouraging me to explore new ideas and even to make mistakes. I am also thankful for his self demonstration on how to work hard and more importantly how to work smart. I am extremely grateful for the opportunity to work with him.

I am also grateful to the members of my thesis committee for their insightful and constructive suggestions on my work which has inspired me to improve it tremendously.

I owe very much to my colleagues and collaborators from Robot learning lab at Cornell: Hema Koppula, Ian Lenz, Jaeyong Sung, Ozan Sener, Ashesh Jain, Dipendra K Misra, Chenxia Wu. I thank them for their generous help on robotic experiments and feedback to my projects. I will always miss the discussions on our weekly group meeting and the fun we had when cleaning the lab after every deadline. I would like to thank Marcus Lim and Changxi Zheng for their hard work on our learning-to-place project. I thank Stephen Moseson for his help on my very first interaction with a robot. I also thank my other collaborators, including John Amend, Gaurab Basu, Alejandro Perez, Mevlana Gemici and others. Last but not least, I thank all robots in our lab, especially Panda, Polar and Kodiak for their unconditional day-and-night support.

I especially thank my dear friend and office mate, Yue Gao, who has given my so many supports and inspirations to both work and life.

Finally, I thank my parents, Shenglan Deng and Weiguo Jiang, for their years of love and faith in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

We make the world we live in and shape our own environment.

Orison Swett Marden (1894).

That the human environments and the objects in it are designed for human usage, is so deeply ingrained in us that when we think about a human environment, we think it through the interplay between these elements. For example, consider a typical office scene in Fig. 1.1, with a chair, table, monitor and keyboard. This scene can be described through many object-object relations, such as chair-in-front-of-table, monitor-on-table, keyboard-on-table, and so on. This particular configuration can also be naturally explained by a sitting human pose in the chair and working with the computer.

Both object-object and human-object relations are essential in reasoning our environments. While at the first blush, introducing human poses may seem to complicate the model, it actually simplifies it to a more parsimonious model. The reason for this is that the set of relevant human poses could be far smaller than the collection of all objects. Therefore, for n objects, we only need to model how they are used by humans, i.e. $O(n)$ relations, as compared with modeling $O(n^2)$ if we were to model object-object relations.

In the following, we first motivate the necessity of capturing both object and human context in one model. We then briefly describe how our proposed learning model can capture them efficiently, followed by showing how we can apply this learning idea to many applications, including scene understanding, human activity anticipation and robotic manipulations.

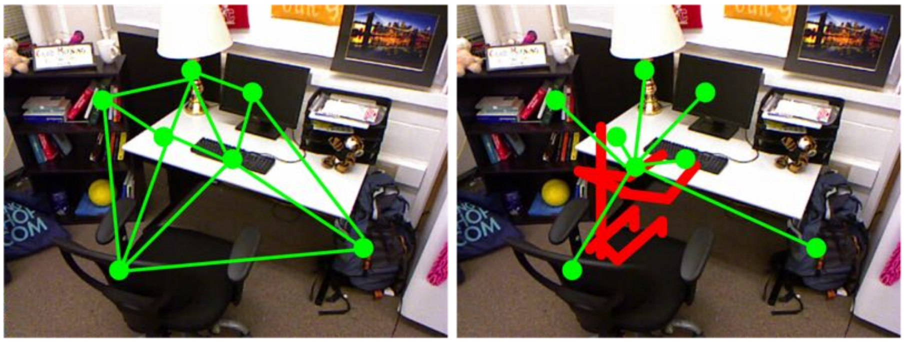


Figure 1.1: Left: Previous approaches model the relations between observable entities, such as the objects. Right: In this thesis, we consider the relations between the objects and hidden humans. Our key hypothesis is that even when the humans are never observed, the human context is helpful.

1.1 Object and Human Context

In this thesis, we argue that a human environment is constructed under two types of relations: *object-object* and *human-object relations*.

When only considering object-object relations, Conditional random fields (CRFs) are a natural choice, as each object can be modeled as a node in a Markov network and the edges in the graph can reflect the object-object relations. CRFs and their variants have thus been applied to many scene modeling tasks (e.g., [114, 1, 102]).

On the other hand, human-object relations which include possible human poses and human-object interactions (or *object affordances*) are not trivial to model because of several reasons: First, humans are not always observable, but we still want to model them as latent factors for making the scene as it is. Second, there can be any number of possible humans in a scene—e.g., some sitting on the couch/chair, some standing by the shelf/table; Third, there can be

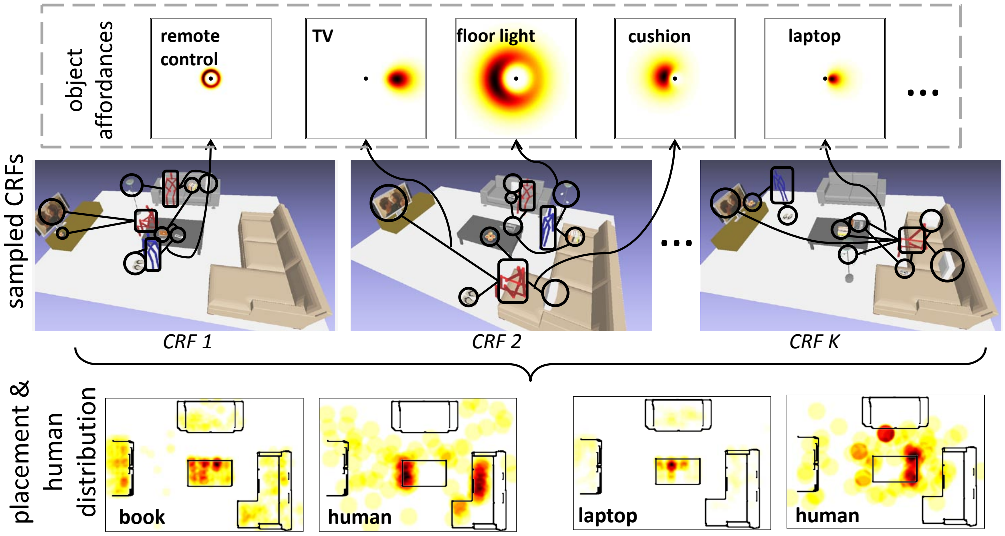


Figure 1.2: An example of instantiated ILCRF for scene arrangement. Top row shows learned object affordances in top-view heatmaps (it shows the probability of the object's location, given a human pose in the center facing to the *right*). Middle row shows a total of K CRFs sampled from our ILCRF algorithm—each CRF models the scene differently. Bottom row shows the distribution of the objects and humans (in the top view of the room) computed from the sampled CRFs.

various types of human-object interactions in a scene, such as watching TV in distance, eating from dishes, or working on a laptop, etc; Fourth, an object can be used by different human poses, such as a book on the table can be accessed by either a sitting pose on the couch or a standing pose nearby; Last, there can be multiple possible usage scenarios in a scene (e.g., see Figure 1.2-middle row). Therefore, we need models that can incorporate latent factors, latent structures, as well as different alternative possibilities.

1.2 Non-parametric Learning

In order to handle those challenges, we propose infinite latent conditional random fields (ILCRFs) to capture both object and human context in a given environment.

Intuitively, An ILCRF is a mixture of CRFs where each CRF can have two types of nodes: existing nodes (e.g., object nodes, which are given in the graph and we only have to infer the value) and latent nodes (e.g., human nodes, where an unknown number of humans may be hallucinated in the room). The relations between the nodes (e.g., object-object edges and human-object edges) could also be of different types. Unlike traditional CRFs, where the structure of the graph is given, the structure of our ILCRF is sampled from Dirichlet Processes (DPs) [133]. DPs are widely used as nonparametric Bayesian priors for mixture models, the resulting DP mixture models can determine the number of components from data, and therefore is also referred as infinite mixture models. ILCRFs are inspired by this, and we call it 'infinite' as it can sidestep the difficulty of finding the correct number of latent nodes as well as latent edge types. Our learning and inference methods are based on Gibbs sampling that samples latent nodes, existing nodes, and edges from their posterior distributions.

To demonstrate the generality of our ILCRFs, we instantiate two specific ILCRFs for two applications in this thesis: scene labeling where the objective is to identify objects in a scene, and scene arrangement where the objective is to find proper placements (including 3D locations and orientations) of given objects in a scene. Despite the disparity of the tasks at the first look, we relate them through one common hidden cause—imaginary humans and object affordances. For both tasks, our ILCRF models each object placement as an existing node, hallucinated human poses as latent nodes and spatial relationships among objects or between objects and humans as edges. We demonstrate in the experiments that this unified model achieves the state-of-the-art results on both synthetic and real datasets. More importantly, we perform an exhaustive analysis on how our model captures different aspects of human context in scenes, in comparisons with numerous baselines. We further demonstrate that by using the two applications together, a robot successfully identified the class of objects in a new room, and placed several objects correctly in it.

1.3 Robotic Manipulations

In addition to a better understanding of our environments, we further show that human context is also critical for personal robots performing daily tasks in human environments.

"Tidy my room." "Put the dishes away." — While these tasks would have been easy for *Rosie* robot from *The Jetsons* TV show, they are quite challenging for our robots to perform. Not only would they need to have the basic manipulation skills of picking up and placing objects, but they would also have to perform them in a way that respects human preferences, such as not placing a laptop in a dish-rack or placing the dishes under the bed.

In order to autonomously perform common daily tasks such as setting up a dinner table, arranging a living room or organizing a closet, a personal robot should be able to figure out where and how to place objects. However, this is particularly challenging because there can potentially be a wide range of objects and placing environments. Some of them may not have been seen by the robot before. For example, to tidy a disorganized house, a robot needs to decide *where* the best place for an object is (e.g., books should be placed on a shelf or a table and plates are better inserted in a dish-rack), and *how* to place the objects in an area (e.g. clothes can be hung in a closet and wine glasses can be held upside down on a stemware holder). In addition, limited space, such as in a cabinet, raises another problem of how to *stack* various objects together for efficient storage. Determining such a placing strategy, albeit rather natural or even trivial to (most) people, is quite a challenge for a robot.

While ILCRFs can infer high-level arrangements of objects, we still need the specific location and orientation for robots placing each object. To find placements that are both stable and preferred, we encode human preferences about placements as well as the geometric relationship between objects and their placing environments by designing appropriate features. We then utilize a graphical model that has two substructures to capture the stability and the semantic preferences respectively. The model also incorporates stacking and constraints that keeps the placing strategy physically feasible. We use max-margin learning for estimating the parameters in our graphical model. The learned model is then used to score the potential placing strategies. Given a placing task, although inferring the best strategy (with the highest score) is provably NP-complete, we express the inference as an integer linear programming (ILP) problem which is then solved efficiently using an linear programming (LP) relaxation.

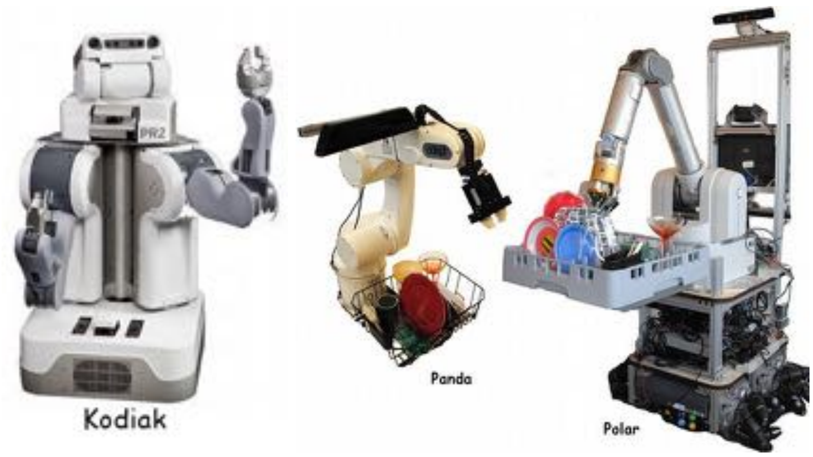


Figure 1.3: Our three robotic platforms used for testing our algorithm: Kodiak (left) is a PR2 robot equipped with a Kinect sensor on top. PANDA (PersonAl Non-Deterministic Arm, in the middle) is an Adept arm with a parallel-plate gripper, mounted with a Kinect. POLAR (PersOnaL Assistant Robot, on right) is a 7-DOF Barrett arm mounted on a Segway Omni base, with a three-fingered hand and a Kinect on top.

1.3.1 Robot Platforms and System

In this work, we perform our robotic experiments mostly on three different robots in our lab: Kodiak, Panda and Polar. Kodiak is a standard PR2 robot (Fig. 1.3 left).

Our PANDA (PersonAl Non-Deterministic Arm) robot (Fig. 1.3 middle) is a 6-DOF Adept Viper s850 arm equipped with a parallel-plate gripper and a Kinect sensor. The arm, together with the gripper, has a reach of 105cm. The arm has a repeatability of 0.03mm in XYZ positioning, but the estimated repeatability with the gripper is 0.1mm. The Kinect sensor-arm calibration was accurate up to an average of 3mm. The arm weighs 29kg and has a rated payload of 2.5kg, but our gripper can only hold up to 0.5kg. The Adept Viper is an industrial arm and has no force or tactile feedback, so even a slight error in positioning can result in a failure to place.

Our POLAR (PersOnaL Assistant Robot) robot (Fig. 1.3 right) is equipped with a 7-DOF WAM arm (by Barrett Technologies) and a three-fingered hand mounted on a Segway RMP 50 Omni base. A Kinect sensor is mounted on top. The arm has a positioning accuracy of ±0.6mm. It has a reach of 1m and a payload of 3kg. The hand does not have tactile feedback. The mobile base can be controlled by speed and has a positioning accuracy of only about 10cm.

Our primary sensor is a depth camera that gives an RGB image together with the depth value at each pixel. In our experiments, we used a Microsoft Kinect sensor which has a resolution of 640x480 for the depth image and an operation range of 0.8m to 3.5m.

1.4 Modeling High-Dimensional Humans

So far, we have mostly focussed on hallucinating static human poses. A very ambitious but natural step to take next would be to hallucinate human motions. As a matter of fact, the ability to anticipate possible future moves of a human is a necessary social skill for humans as well as for robots that work in assembly-line environments (e.g., Baxter) or in homes and offices (e.g., PR2). With such a skill, the robots can work better with humans by performing appropriate tasks and by avoiding conflict. For instance, Koppula et. al. [75] used anticipation in assistive robotic settings, such as in the tasks of opening doors for people or serving drinks to people.

Human activity anticipation is a very challenging task, especially in *un-*

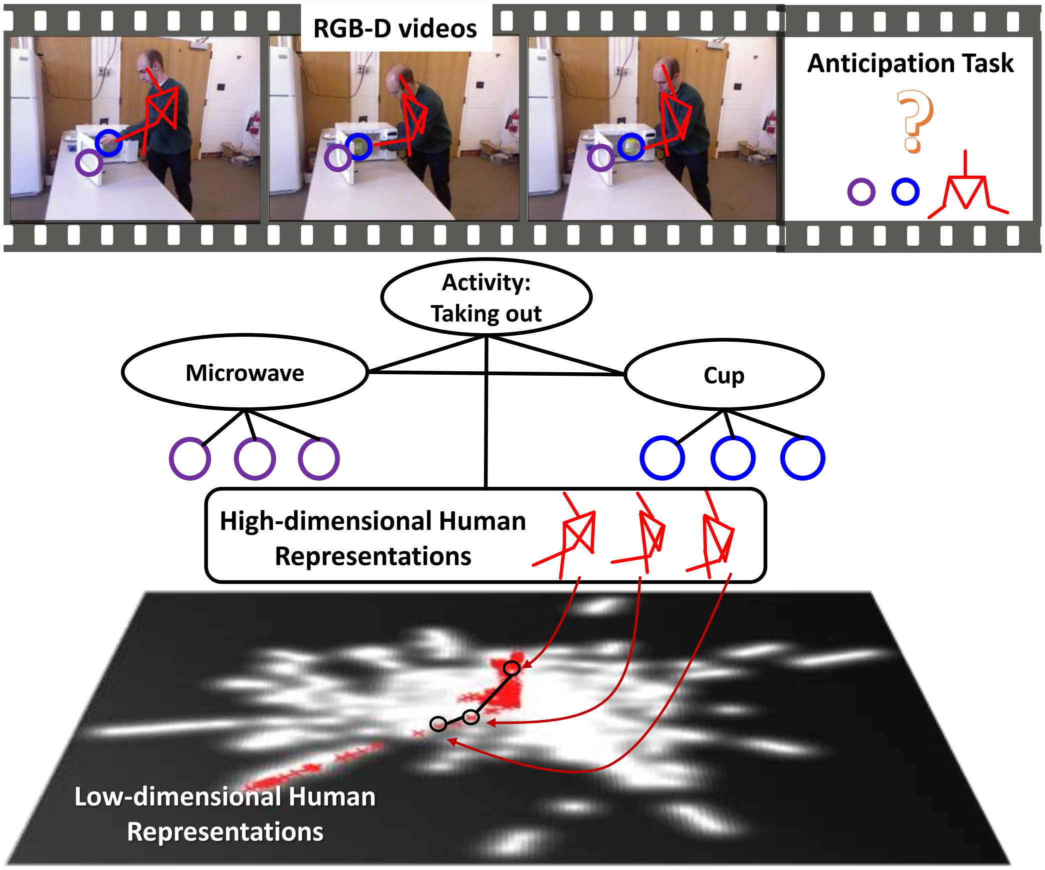


Figure 1.4: Given an RGB-D video of a human interacting with the environment, we are interested in predicting the future: what activity will he perform and how the environment and human pose will change. The key idea in this work is to compactly represent the high-dimensional human configuration in a low-dimensional space so that we can model relations between the human, activity and objects more effectively in a graphical model.

structured environments with a large variety of objects and activities. Koppula et. al. [75] have shown that the rich context (such as object-object and human-object spatial relations) is important for predicting high-level human activities. However for anticipation and robotic planning, predicting *detailed* human motions is also crucial. In this work, our goal is to model the detailed human motions, along with the rich context, in anticipating the human activities. We specifically focus on *how to represent (and learn with) high-dimensional human con-*

figurations and their temporal dynamics.

Recently, high-dimensional description of human motions is widely available through motion capture data or RGB-D cameras (e.g., Kinect), where a human configuration is specified by the joint locations and orientations and often has more than 30 degrees of freedom. While it captures human kinematics and dynamics accurately, modeling human motions in such space (much higher than 30 DOF when considering velocities and accelerations) often requires a detailed musculo-skeletal human model and a large number of spatial and timing constraints to produce smooth and realistic motions [16].

Such a high-DOF model does not lend itself to use in learning models where rich modeling of the human with the environment is needed. Therefore, some works assume a few static human poses are representative enough [34, 37, 54, 58] or simplify a human configuration to a 2D point for navigation task [5, 69, 148, 78] or to a 3D trajectory of one hand while keeping the rest body static neglecting kinematic constraints [75, 76]. In these works, human motions are under-represented and would fail when a more elaborate human motion prediction is required.

1.5 Organization of this Thesis

The rest of this thesis is organized as follows: Chapter 2 describes how to model 3D scenes in terms of object context only. Chapter 3 describes how we define two important elements in human context: human representations and object affordances. Chapter 4 presents our non-parametric learning model ILCRFs. Chapter 5 describes how to instantiate ILCRFs in two different applications and the corresponding experimental results. Chapter 6 extends ILCRF to anticipate human activity. Chapter 7 describes how we can place novel objects using learning approaches. Finally, we conclude this thesis in Chapter 8.

1.6 First Published Appearances of Described Contributions

Most of the contributions presented in this thesis have appeared as publications: [53, 54, 58, 57] for Chapter 2-5; [59] for Chapter 6; [55, 60, 63] for Chapter 7.

Other contributions are not discussed in this thesis because of the scope, including [56, 61, 62].

CHAPTER 2

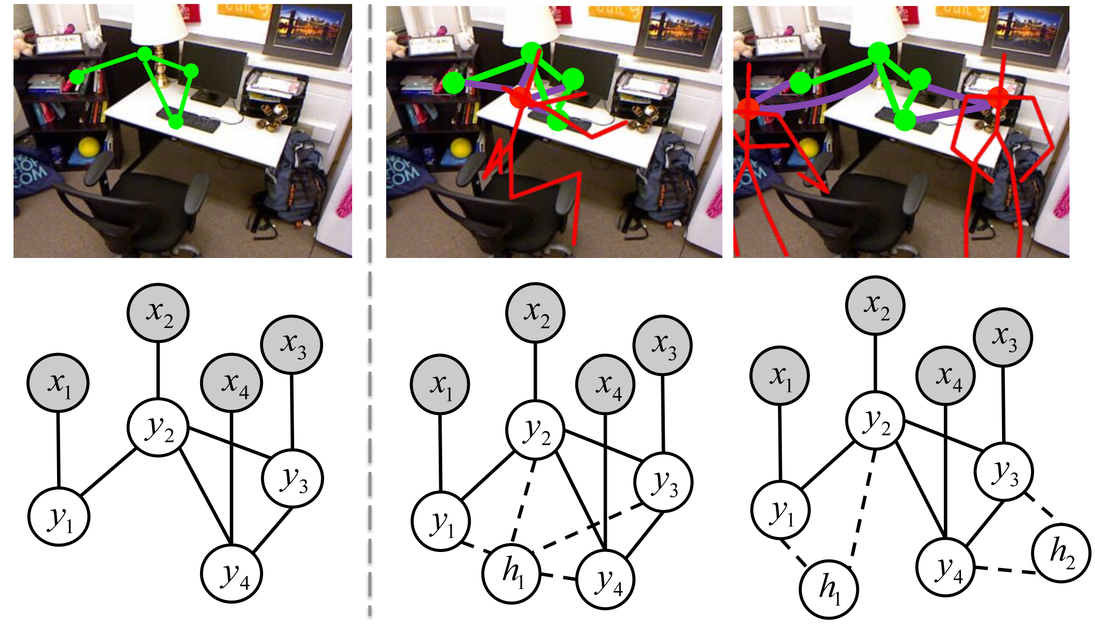
MODELING 3D SCENES

With the availability of stereo/range camera nowadays, such as Kinect, we have more access than ever to RGBD data which has both RGB color and depth information of a 3D scene. Such RGBD data not only allows an algorithm to capture objects self properties, such as shape and orientation, but also makes reasoning objects pairwise relations more easily, such as depth ordering and spacial relations. Thus, in this thesis we focus on reasoning 3D data, such as RGBD images or videos of a given scene which could be either static or dynamic.

In this chapter, we establish the problem of modeling 3D scenes first. We then introduce a popular approach which models object-object relations and some related works in the literature. In the next chapter, we will show how to augment the model to admit human-object relations also.

2.1 Problem Formulation

Given a scene, we are interested in objects that are (or could be) in it, such as identifying the object class in the scene labeling task. The scene is represented as an RGB-D point cloud. We first segment the point cloud based on smoothness and continuity of surfaces using the approach in [73]. We use $\mathcal{X} = \{x_1, \ldots, x_N\}$ to denote N segments of interest, and $\mathcal{Y} = \{y_1, \ldots, y_N\}$ to denote the corresponding labels. In the task of scene labeling, for instance, x_i is the observed appearance features and locations of the i^{th} segment/object in the scene, and y_i is an integer between 1 and M representing the class label, such as monitor, chair, floor, etc.



(a) Modeling objects using CRF

(b) Modeling humans as latent nodes

Figure 2.1: **Graphical models for scene modeling.** (a) Conditional random field (CRF) has been used to capture objects and their relations in a scene. (b) In our work, we model possible human configurations as latent nodes in a scene in order to capture human and object relations in a scene.

We model the correspondance between $\mathcal{X}$ and $\mathcal{Y}$ through probabilistic distribution $P(\mathcal{Y}|\mathcal{X})$, and the objective of the labeling task is to find the optimal labels given observations, namely,

$$\mathcal{Y}^* = \arg\max_{\mathcal{Y}} P(\mathcal{Y}|\mathcal{X})$$

A simple and naive solution is to treat objects independently: $y_i^* = \arg\max_y P(y|x_i)$. In this way, we can label the object class using its own shape/appearance features such as HOG [15]. However, these methods would suffer from noisy local features and the lack of *context* of the whole scene.

2.2 CRFs for Object Context

There are many works trying to capture the context from object-object relations, which can be naturally modeled through conditional random fields (CRFs) [145, 1, 125, 123, 114, 102]. A CRF is a network where each node can be modeled as an object and each edge reflects the relationship between the linked two objects. An example is shown in Fig. 2.1-(a).

Definition 1. *$CRF(\mathcal{X}, \mathcal{Y}, E_Y)$* is a conditional random field if that, when conditioned on $\mathcal{X}$, random variables $\mathcal{Y}$ follow the Markov property with respect to the graph E_Y: The absence of an edge between nodes y_i and y_j implies that they are independent given other nodes. ■

Thus, the likelihood of $\mathcal{Y}$ given $\mathcal{X}$ is given by: $P(\mathcal{Y}|\mathcal{X}) \propto \prod_{c \in C} \psi_c(X_c, Y_c)$, where C is all the maximum cliques, and $Y_c \in \mathcal{Y}$ and $X_c \in \mathcal{X}$ are in the same clique c. ψ is the potential functions. Following [73, 1], we use log-linear node (ψ^o) and edge potentials (ψ^{oo}) to capture object-object context:

$$\begin{aligned} P(\mathcal{Y}|\mathcal{X}) &\propto \prod_{i=1}^{N} \psi^o(x_i, y_i) \prod_{(y_i, y_j) \in E_Y} \psi^{oo}(x_i, x_j, y_i, y_j) \\ &= \exp \sum_i \sum_{k=1}^{M} \mathbf{1}_{\{y_i = k\}} (\theta^o_k)^\top \phi^o(x_i) \\ &\times \exp \sum_{ij} \sum_{kl} \mathbf{1}_{\{y_i = k\}} \mathbf{1}_{\{y_j = l\}} (\theta^{oo}_{kl})^\top \phi^{oo}(x_i, x_j) \end{aligned} \tag{2.1}$$

where ϕ^o and ϕ^{oo} are object's own and pairwise features. For example, in our experiments, ϕ^o includes local features such as histograms of HSV colors, normal and dimensions of the segment's bounding box. ϕ^{oo} includes features such as difference of HSV colors, displacement or co-planarity of the two segments [1]. θ^o_k and θ^{oo}_{kl} are parameters to learn for each class k and each pair of classes (k, l).

2.3 Related Work

There is a significant body of work that captures the relations between different parts of the object [24] and between different objects [73]. In the past, 3D layout or depths have been used for improving object detection (e.g., [118, 121, 41, 85, 43, 88]), where an approximate 3D geometry is inferred from 2D images. Recent works [145, 73, 1, 125] address the problem of labeling 3D point clouds. Reasoning in 3D allows an algorithm to capture stronger context, such as shape, stability and orientation of the objects [55, 52, 56].

CHAPTER 3

HALLUCINATED HUMANS

> *"We bear in mind that the object being worked on is going to be ridden in, sat upon, looked at, talked into, activated, operated, or in some other way used by people individually or en masse."* Dreyfuss [21].

While modeling object-object relations has been a popular approach in scene understanding, we hypothesize that such relations could only be an artifact of certain hidden factors, such as humans. In fact, even when no human is present in an indoor scene, the potential human-object interactions give such a strong cue for scene understanding that we want to model it as latent variables in our algorithms. Moreover, modeling human-object relations is parsimonious and efficient as compared to modeling the pairwise object-object relationships: For n objects, we only need to model how they are used by humans, i.e., $O(n)$ relations, as compared with modeling $O(n^2)$ if we were to model object to object context naively.

In this chapter, we first define the representation of human configurations and human-object relations (referred as 'object affordances' in the rest of the thesis). Then we show how to incorporate the human context into the CRF we just described.

3.1 Human Representations

A human configuration, denoted by h, is comprised of a pose type, location and orientation. The pose type, as shown in Fig. 3.1, is specified by relative positions of 15 body joints, such as head, torso, shoulders, hips, etc. In this work, we consider six static poses extracted from real human activity dataset: We collected all poses in Cornell Activity Dataset-60 [129], and clustered them using k-means algorithm giving us six types of skeletons. Each pose could be at any X-Y-Z location and in different orientations $\in [0, 2\pi)$ inside the scene.

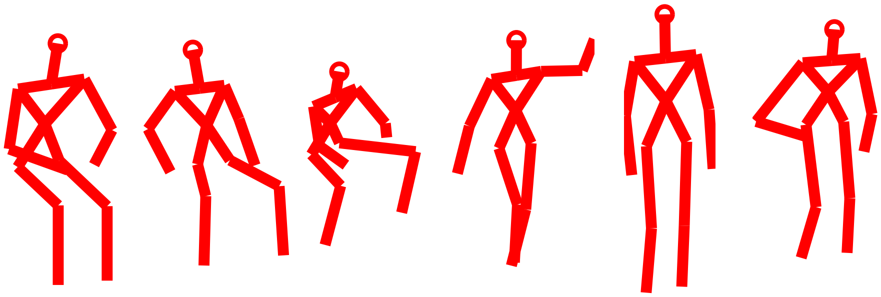


Figure 3.1: Six types of human poses extracted from Kinect 3D data.

3.2 Object Affordances

A human can use the objects at different distances and orientations from the human body. For example, small hand-held devices are typically held close to and in front of the human. Objects such as a TV and decoration pieces are typically placed at a distance. The human-object spatial relations can be a strong hint of the object class as well as where to place the object. Therefore, we define the object affordance as the probability distribution of the object's relative 3D location with respect to a human pose h. An example of the laptop's affordance is shown in Fig. 3.2: Given a centered sitting human pose h, the distribution of a laptop is projected onto a top-view and side-view heatmaps, indicating that the laptop is most likely to appear right ahead of human hands.

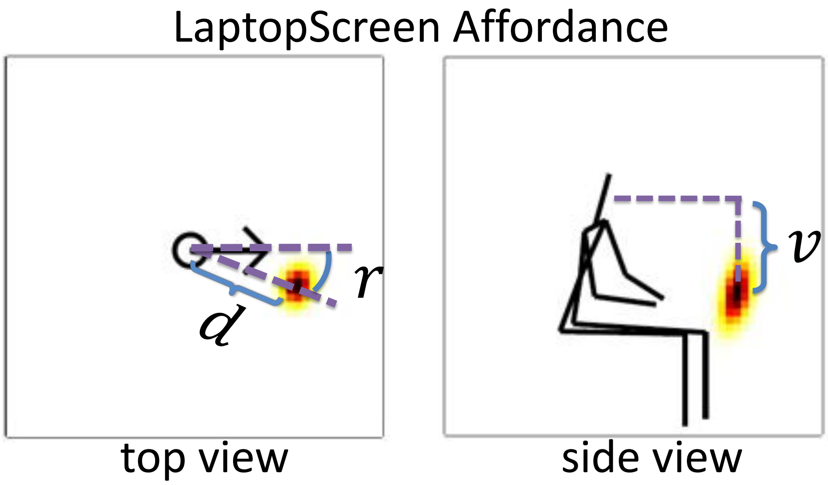


Figure 3.2: The affordance of a laptop screen, visualized in top- and side-view.

In detail, we define an object affordance as a product of terms capturing the preferred distance and orientation from the object to the human pose:

$$\psi^{ho}(x, y, h; \Theta) = \psi_{\text{dist}}\psi_{\text{rel}}\psi_{\text{ori}}\psi_{\text{vert}}. \tag{3.1}$$

We now describe each term below:

Distance preference. Some objects are preferred to be at a certain distance from humans, such as a TV or a laptop. This preference, encoded as ψ_{dist}, includes how far the object should be and how strong this bias is. Let $d(x, y, h)$ be the Euclidean distance between the human and object, as shown in Fig. 3.2. The distance follows a log-normal distribution, i.e.,

$$\psi_{\text{dist}}(x, y, h; \mu_d, \sigma_d) = \frac{\exp(-\frac{(\ln d(x,y,h)-\mu_d)^2}{2\sigma_d^2})}{d(x, y, h)\sigma_d\sqrt{2\pi}}. \tag{3.2}$$

Relative angular preference. There is a preference for objects to be located at a certain angle with respect to human poses. For example, people will sit in front of a laptop, but prefer the mouse to be on their right (or left). Let $r(x, y, h)$ be relative angle from the object to human (as shown in Fig. 3.2-left), and we assume it follows a von Mises distribution:

$$\psi_{\text{rel}}(x, y, h; \mu_r, \kappa_r) = \frac{\exp(\kappa \cos(r(x, y, h) - \mu_r)}{2\pi I_0(\kappa_r)} \tag{3.3}$$

Orientation preference. There is a preference for objects to be oriented at a certain angle with respect to the human pose (e.g., a monitor should also be facing towards the skeleton when located in front of the skeleton). We use $o(x, y, h)$ to denote the difference between orientations of the object and human (regardless their relative angle), i.e., $o(x, y, h) = |o(x, y) - o(h)|$. Similarly to ψ_{rel}, we also use a von Mises distribution for this term:

$$\psi_{\text{ori}}(x, y, h; \mu_o, \kappa_o) = \frac{\exp(\kappa \cos(o(x, y, h) - \mu_o)}{2\pi I_0(\kappa_o)}. \tag{3.4}$$

Vertical difference preference. ψ_{vert} is a Gaussian distribution of the object's relative height to a human pose. Let $v(x, y, h)$ be the vertical distance between the human and object, as shown in Fig. 3.2-right. We it follows a normal distribution:

$$\psi_{\text{vert}}(x, y, h; \mu_v, \sigma_v) = \frac{\exp(-\frac{(v(x,y,h)-\mu_v)^2}{2\sigma_v^2})}{v(x, y, h)\sigma_v\sqrt{2\pi}} \tag{3.5}$$

In this way, we specify one object affordance ψ^{ho} using one set of parameters $\Theta = \{\mu_d, \sigma_d, \mu_r, \kappa_r, \mu_o, \kappa_o, \mu_v, \sigma_v\}$.

3.3 Human Context: a Double-Edged Sword

The human context is very important for understanding our environment. In fact, even when no human is present in an indoor scene, the potential human-object interactions give such a strong cue for scene understanding that we want to model it as latent variables in our algorithms.

3.3.1 Modeling Observed Human Context in CRFs

Let us first consider how to model human context when both human configurations and object affordances are given.

Suppose K human configurations are given in a scene, each of which is specified by a pose type, location and orientation, such as the sitting pose in Fig. 2.1-(b). We model each human pose as a node in the graph, $\mathcal{H} = \{h_1, \ldots, h_K\}$ and each human-object relationship as an edge (y_i, h_{z_i}) where $z_i \in \{1, \ldots, K\}$ denotes which human pose is using the i^{th} object.[1] For example, in the second case in Fig. 2.1-(b), $z_1 = z_2 = 1$ and $z_3 = z_4 = 2$. We use $\mathcal{Z} = \{z_1, \ldots, z_N\}$ to denote these human configuration correspondances.

Suppose we are also given M different object affordances, $\Psi = \{\psi_1^{ho}, \ldots, \psi_M^{ho}\}$ where each ψ_k^{ho} is defined in (3.1) with parameter Θ_k. For each object i, we use $\omega_i \in \{1, \ldots, M\}$ to denote its correspondent affordance (and $\Omega = \{\omega_1, \ldots, \omega_N\}$ for all objects). In other words, we associate the edge (y_i, h_{z_i}) with the potential $\psi_{\omega_i}^{ho}(x_i, y_i, h_{z_i})$.

Given such a CRF with known human context (specified by $\mathcal{G} = \{\mathcal{H}, \mathcal{Z}, \Psi, \Omega\}$, the likelihood now is,

$$\begin{aligned} P(\mathcal{Y}|\mathcal{X}, \mathcal{G}) \quad \propto \quad & \prod_{i=1}^{N} \psi^{o}(x_i, y_i) \prod_{(y_i, y_j)} \psi^{oo}(x_i, x_j, y_i, y_j) \\ & \prod_{i=1}^{N} \psi_{\omega_i}^{ho}(x_i, y_i, h_{z_i}) \end{aligned} \tag{3.6}$$

where the first two terms are defined in Eq. (2.1) and the last one is in Eq. (3.1).

How to model hidden human context? More often humans are not present

[1] We assume a human pose can interact with multiple objects at the same time but each object is used by only one human pose.

in the scene, nevertheless, the potential human-object relations are valuable information for scene understanding. Such latent nature of human context, combined with the enormous space of possible human configurations and object affordances, can lead to an ill-posed problem. For example, one potential explanation of the scene could be humans floating in the air and prefer stepping on every object as the affordance!

However, the human context cannot be easily harnessed because the space of possible human configurations and object affordances is rather large. Furthermore, the humans are not always observable and such latent nature leads to an ill-posed problem while using it. For example, one potential explanation of the scene could be humans floating in the air and prefer stepping on every object as the affordance! The key to modeling the large space of latent human context lies in building *parsimonious* models and providing *priors* to avoid physically-impossible models.

3.3.2 Model Parsimony

While there are infinite number of human configurations in a scene and countless ways to interact with objects, only a few human poses and certain common ways of using objects are needed to explain most parts of a scene. These could be shared across objects and be instantiated to numerous forms in reality. We will do so by representing them as 'topics,' according to which objects in a scene are generated. This is analogous to the document topics [133, 62], except that in our case topics will be continuous distributions and factored. Similar to document topics, our human-context topics can be *shared* across objects and scenes. As a result, the model's complexity, i.e., the number of parameters, is significantly reduced. We describe the two types of topics below:

Human Configuration Topics. In a scene, there are certain human configurations that are used more commonly than others. For instance, in an office a sitting pose on the chair and a few poses standing by the desk, shelf and whiteboard are more common. Most of the objects in an office are arranged for these human configurations.

Object Affordance Topics. An object affordance, despite its large variety, can often be represented as a mixture of several commonly shared object-affordance topics. For example, both using a keyboard and reading a book require a human pose to be close to objects. However, when books are not in use, they can be stored away from humans. Therefore, the affordance of a book would be a mixture of a 'close-to' and a 'spread-out' topic.

3.3.3 Physics-Based Priors

In order to obtain meaningful human-configuration and object-affordance topics, we impose prior that follows physics and conventions to those topics.

Human Configuration Prior. Our hallucinated human configurations need to follow basic physics. Encoding physics-based notions about objects has been shown to be successful in 3D geometric interpretation [125, 55]. We consider the following two properties as priors for the generated human configurations [37]:

1) Kinematics. We perform collision check so that the human pose is kinematically feasible. *2) Dynamics.* We check if the human skeleton is supported by the nearby environments to ensure its stability.

Object Affordance Prior. In general, it is more likely for an object to be close to humans while being used. Furthermore, most objects' affordance should be symmetric in their relative orientation to the humans' left or right. We encode this information in the design of the function quantifying affordances and as Bayesian priors in the estimation of the function's parameters.

CHAPTER 4

NON-PARAMETRIC LEARNING ALGORITHMS

In this thesis, we propose a type of mixture CRFs—infinite latent conditional random fields (ILCRFs), which can capture the following properties:

1. Unknown number of latent nodes. This is essential for applications of finding hidden causes, such as scene modeling where the number of possible human poses in a scene is unknown and changes across different scenes.

2. Unknown number of the ***types*** *of potential functions.* Potential function measures the relationship between nodes, and therefore, having variety in them can help us model complex relations. For example, in the task of image segmentation, different types of context can be modeled as different edges in a CRF [50]. In this work, we use them to capture different object affordances.

3. Mixture CRFs. The complexity of real-world data may not always be explained by a single CRF. Therefore having a mixture of CRFs, with each one modeling one particular conditional independency in the data, can increase the expressive power of the model.

4. Ability to place informative priors on the structure of CRFs. This can help producing more plausible CRFs as well as reducing the computational complexity.

We achieve this by imposing Bayesian nonparametric priors—Dirichlet processes (DPs)—to the latent variables, potential functions and graph structures. In this chapter, we first describe the classic DPs and a non-parametric mixture model built upon DPs in Sec. 4.1. We then present our ILCRFs in Sec. 4.2 and a Gibbs-sampling based learning and inference algorithm along with the model.

Lastly we discuss related works in Sec. 4.3.

4.1 Background: DPMM

Dirichlet process is a stochastic process to generate distributions that is used to model clustering effects in the data. It has been widely applied to modeling *unknown* number of components in mixture models (such as modeling the unknown number of object parts in part-based object detection models [127]), which are often called *infinite* mixture models. (Formal definition can be found in [133].)

Definition 2. A DP mixture model, $DP(\alpha, B)$, defines the following generative process (also called the stick-breaking process), with a concentration parameter α and a base distribution B:

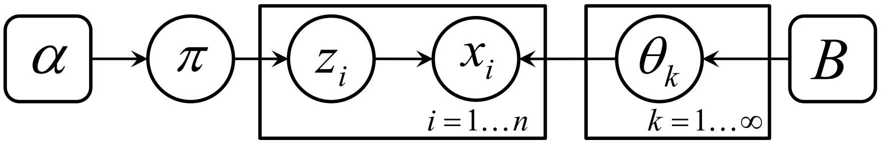
1. Generate infinite number of mixture components, parameterized by $\Theta = \{\theta_1, \ldots, \theta_\infty\}$, and their mixture weights $\boldsymbol{\pi}$:

$$\theta_k \sim B,\ b_k \sim Beta(1, \alpha),\ \ \pi_k = b_k \prod_{i=1}^{k-1}(1 - b_i). \tag{4.1}$$

2. Assign the z_i^{th} component to each data point x_i and draw from it:

$$z_i \sim \boldsymbol{\pi}, \qquad x_i \sim F(\theta_{z_i}). \tag{4.2}$$

The process can be represented in the following plate notation: ■



4.2 ILCRF

ILCRF uses DPs to admit an arbitrary number of latent variables and potential functions to obtain a mixture of latent CRFs. In brief, it generates latent variables and potential functions from two DPs respectively, and each data point builds a link, associated with one potential function, to one latent variable. Different samples thus form different CRFs.

Definition 3. An ILCRF($\mathcal{X}, \mathcal{Y}, E_Y, \alpha_h, B_h, \alpha_\psi, B_\psi$) is a mixture of CRFs, where the edges in $\mathcal{Y}$ are defined in graph E_Y and latent variables $\mathcal{H}$ as well as the edges between $\mathcal{H}$ and $\mathcal{Y}$ are generated through the following process:

1. Generate infinite number of latent nodes $\mathcal{H} = \{h_1, h_2, \ldots, h_\infty\}$ and a distribution $\boldsymbol{\pi}_h$ from a DP process $DP(\alpha_h, B_h)$ following Eq. (4.1); Assign one edge to each label y_i that links to h_{z_i}, where $z_i \sim \boldsymbol{\pi}_h$ following Eq. (4.2).
2. Generate infinite number of potential functions ('types' of edges) $\Psi = \{\psi_1, \ldots, \psi_\infty\}$ and a distribution $\boldsymbol{\pi}_\psi$ from a DP process $DP(\alpha_\psi, B_\psi)$ following Eq. (4.1); Assign one potential function ψ_{ω_i} to each edge (y_i, h_{z_i}), where $\omega_i \sim \boldsymbol{\pi}_\psi$ following Eq. (4.2). ■

We now illustrate the process using Fig. 4.1. Consider first sampled CRF ('CRF-1' in the figure) with four visible nodes y_i ($i = 1 \ldots 4$). In the first step, y_1 is connected to h_1, y_2 to h_3, y_3 to h_7 and y_4 to h_1 again. This is because z_i's ($i = 1 \ldots 4$) are sampled as $(1, 3, 7, 1)$ from $DP(\alpha_h, B_h)$. Since only h_1, h_3 and h_7 are active, we draw their values from $DP(\alpha_h, B_h)$. Thus, we get a CRF with three latent nodes $\{h_1, h_3, h_7\}$. In the second step, the potential function of edge (y_1, h_1) is assigned to ψ_1, (y_2, h_3) to ψ_2, (y_3, h_7) to ψ_5 and (y_4, h_1) to ψ_1. This is because ω_i's are sampled as

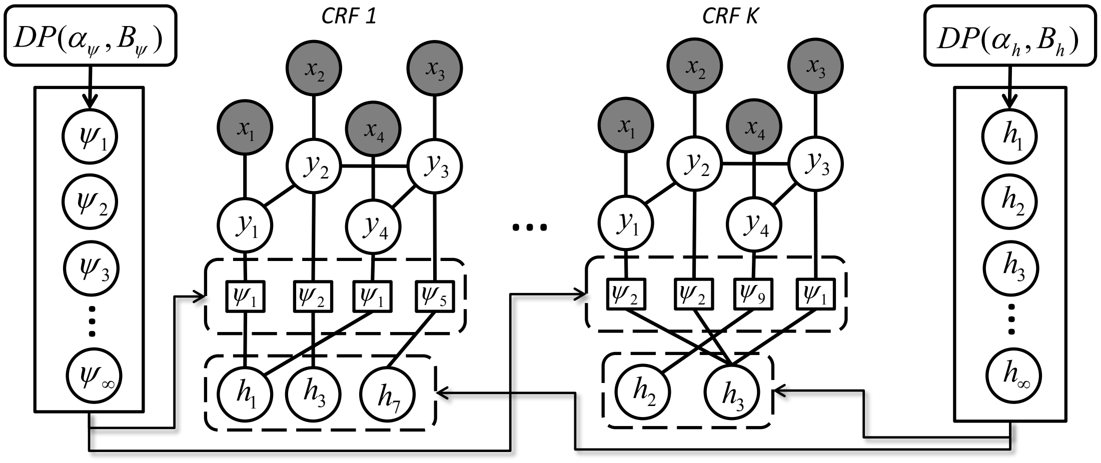


Figure 4.1: Graphical representations of our infinite latent CRF (ILCRF).

$(1, 2, 5, 1)$ from $DP(\alpha_\psi, B_\psi)$. Since, only (ψ_1, ψ_2, ψ_5) are active, we have three edge types in this CRF. We draw their parameters from $DP(\alpha_\psi, B_\psi)$. Repeating this procedure may generate different latent CRFs such as *'CRF-K'* which has two different latent nodes and three different edge types. In the end, their mixture forms the ILCRF. Note that the structure of labels (edges between y_i's) is defined by E_Y and is shared across all the sampled CRFs.

From the probabilistic perspective, ILCRF defines a distribution over different CRFs with latent variables, where each CRF is specified by $\mathcal{G} = \{\mathcal{H}, \mathcal{Z}, \Psi, \Omega\}$ and its likelihood is governed by prior distributions B_h and B_ψ (their specific forms are given in Sec. 4.2.2). Specifically, the first generative step above defines the probability of latent nodes and edges:

$$P(\mathcal{H}, \mathcal{Z}|\alpha_h, B_h) = \int P(\mathcal{H}, \boldsymbol{\pi}_h|\alpha_h, B_h) \prod_{i=1}^{N} \boldsymbol{\pi}_h(z_i) d\boldsymbol{\pi}_h \tag{4.3}$$

Similarly, the second step defines the probability of potentials for all edges between $\mathcal{Y}$ and $\mathcal{H}$:

$$P(\Psi, \Omega|\alpha_\psi, B_\psi) = \int P(\Psi, \boldsymbol{\pi}_\psi|\alpha_\psi, B_\psi) \prod_{i=1}^{N} \boldsymbol{\pi}_\psi(\omega_i) d\boldsymbol{\pi}_\psi \tag{4.4}$$

Since $\mathcal{G}$ is latent, we marginalize over all its possible values to compute the overall likelihood of an ILCRF:

$$\begin{aligned} P(\mathcal{Y}|\mathcal{X}) &= \int P(\mathcal{Y}, \mathcal{G} = \{\mathcal{H}, \mathcal{Z}, \Psi, \Omega\}|\mathcal{X})\; d\mathcal{G} \qquad (4.5) \\ &= \int \underbrace{P(\mathcal{H}, \mathcal{Z}|\alpha_h, B_h)}_{\text{DP prior for } \mathcal{H} \text{ (4.3)}} \underbrace{P(\Psi, \Omega|\alpha_\psi, B_\psi)}_{\text{DP prior for } \Psi \text{ (4.4)}} \\ &\quad \times \underbrace{P(\mathcal{Y}|\mathcal{X}, \mathcal{G} = \{\mathcal{H}, \mathcal{Z}, \Psi, \Omega\})}_{\text{conditional prob. of the CRF (3.6)}} d\mathcal{G} \end{aligned}$$

Exact computation of this likelihood is prohibitive in practice. We therefore present learning and inference methods based on Gibbs sampling in the following.

4.2.1 Gibbs Sampling for Learning and Inference

Gibbs sampling states that, if we sample latent CRFs, including the edge/structure of G, the value of latent nodes $\mathcal{H}$ and the edge types Ψ, from their posterior distributions, then the samples approach the joint distribution $P(\mathcal{Y}, G_\ell, \mathcal{H}, \Psi|\mathcal{X})$. And this can be further used to estimate $P(\mathcal{Y}|\mathcal{X})$ in (4.5) and to infer the most likely values of $\mathcal{Y}$.

We present the posterior distributions below, modified from the Chinese restaurant process [96, 133] for classic DP mixture models.

- Sample the graph structure, i.e., one edge for each y_i to one latent node:[1]

$$z_i = z \propto \begin{cases} \frac{n^h_{-i,z}}{N+m-1+\alpha_h} \psi_{\omega_i}(x_i, y_i, h_z) & n^h_{-i,z} \geq 0, \\ \frac{\alpha_h/m}{N+m-1+\alpha_h} \psi_{\omega_i}(x_i, y_i, h_z) & \text{otherwise} \end{cases} \tag{4.6}$$

- Sample values for each latent node in the graph:

$$h_k = h \propto B_h(h) \times \prod_{i:z_i=k} \psi_{\omega_i}(x_i, y_i, h) \tag{4.7}$$

- Assign the type of potential functions to each edge:[2]

$$\omega_i = \omega \propto \begin{cases} \frac{n^\psi_{-i,\omega}}{N+m-1+\alpha_\psi} \psi_\omega(x_i, y_i, h_{z_i}) & n^\psi_{-i,\omega} \geq 0, \\ \frac{\alpha_\psi/m}{N+m-1+\alpha_\psi} \psi_\omega(x_i, y_i, h_{z_i}) & \text{otherwise} \end{cases} \tag{4.8}$$

- Sample the parameters of each selected potential function:

$$\psi_k = \psi \propto B_\psi(\psi) \times \prod_{i:\omega_i=k} \psi(x_i, y_i, h_{z_i}) \tag{4.9}$$

- Sample labels:

$$\begin{aligned} y_i = y \quad \propto \quad & \psi_{\omega_i}(x_i, y, h_{z_i}) \times \psi^o(x_i, y) \\ & \times \prod_{(y_i, y_j)} \psi^{oo}(x_i, x_j, y, y_j) \end{aligned} \tag{4.10}$$

Note that when we sample the graph structure in Eq. (4.6) and (4.8), we assume that the partition function accross the different graph structures is constant. Another commonly used approximation is via pseudo-likelihood [79]:

[1] The posterior distribution of a variable is proportional to its prior and to its likelihood. In the case of z_i, it means that the probability of linking an edge from y_i to h_z is determined by: 1) the likelihood of this edge, given by $\psi_{\omega_i}(x_i, y_i, h_z)$; 2) the number of other subjects choosing the same latent node, i.e., $n^h_{-i,z}$ where $n^h_{-i,z} = I\{z_j = z, j \neq i\}$. In addition, the chance of selecting a new latent node is given by α_h/m out of m latent nodes sampled from B_h. (See [96] for more details).

[2] Similar to (4.6), the probability of choosing ψ_ω is proportional to the number of other edges choosing the same function ($n^\psi_{-i,\omega}$) and the likelihood of this edge using this function.

Table 4.1: Summary of Gibbs sampling in ILCRF for two applications.

Task	Phase	Gibbs sampling (Sect. 4.2.1)				
		z (4.6)	$\mathbf{h}$ (4.7)	$\boldsymbol{\omega}$ (4.8)	$\boldsymbol{\psi}$ (4.9)	$\mathcal{Y}$ (4.10)
Scene	train	✓	✓		✓	
Labeling	test	✓	✓	✓		✓
Scene	train	✓	✓		✓	
Arrangement	test	✓	✓			✓

We approximate the true likelihood $P(\mathcal{Y}|\mathcal{X},\mathcal{G})$ by $\prod_i P(y_i|\mathcal{Y}_{-i},\mathcal{X},\mathcal{G})$ where $P(y_i|\mathcal{Y}_{-i},\mathcal{X},\mathcal{G}) = \frac{\psi_{\omega_i}(x_i,y_i,h_z)\psi^o(x_i,y_i)\prod_{(y_i,y_j)}\psi^{oo}(x_i,x_j,y_i,y_j)}{\sum_{y_i=y}\psi_{\omega_i}(x_i,y,h_{z_i})\psi^o(x_i,y)\prod_{(y_i,y_j)}\psi^{oo}(x_i,x_j,y,y_j)}$ (Eq. (3.6)). Now we can sample z_i based on this pseudo-likelihood, i.e., $z_i = z \propto \frac{n^h_{-i,z}}{N+m-1+\alpha_h}P(\mathcal{Y}|\mathcal{X},\mathcal{G}) \propto \frac{n^h_{-i,z}}{N+m-1+\alpha_h}\prod_i P(y_i|\mathcal{Y}_{-i},\mathcal{X},\mathcal{G})$. Note that for all other $j \neq i$, $P(y_j|\mathcal{Y}_{-i},\mathcal{X},\mathcal{G})$ is constant w.r.t. z_i, and so is $\psi^o(x_i,y_i)$ and $\psi^{oo}(x_i,x_j,y_i,y_j)$. Hence, $z_i = z \propto \frac{n^h_{-i,z}}{N+m-1+\alpha_h}P(y_i|\mathcal{Y}_{-i},\mathcal{X},\mathcal{G}) \propto \frac{n^h_{-i,z}}{N+m-1+\alpha_h}\frac{\psi_{\omega_i}(x_i,y_i,h_z)}{\sum_{y_i=y}\psi_{\omega_i}(x_i,y,h_{z_i})\psi^o(x_i,y)\prod_{(y_i,y_j)}\psi^{oo}(x_i,x_j,y,y_j)}$. However in our experiments, we observe little performance gain by doing this and hence ignore the denominator in our implementations.

As for learning the E_Y, when labels are given in the training data, E_Y is independent with latent variables $\mathcal{H}$ (if the partition function is ignored), and therefore can be learned separately. For instance, E_Y used in our labeling task is learned separately using max-margin learning [1].

4.2.2 Learning Object Affordances

The primary part of learning an ILCRF model is to learn object affordances. As we defined in Sec. 3.2, the affordance ψ is parameterized by $\Theta = \{\mu_d,\sigma_d,\mu_r,\kappa_r,\mu_o,\kappa_o,\mu_v,\sigma_v\}$ for each object class. Therefore, sampling ψ in Eq. (4.9)

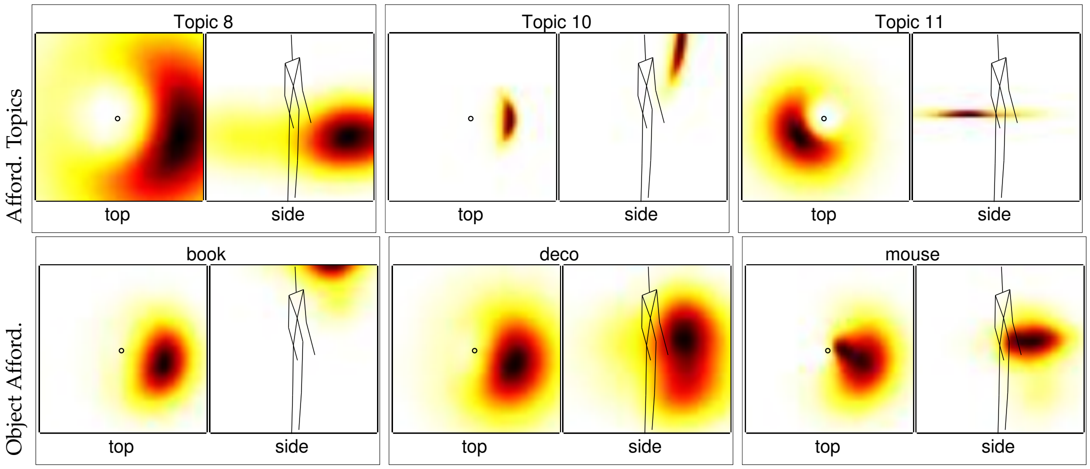


Figure 4.2: Examples of learned affordance *topics* and object affordances as a mixture of those topics (see Sec. 4.2.2).

is done through sampling each parameter in Θ. In practice, the posterior sampling of Θ may be difficult when not using conjugate priors. To handle this, we use the maximum a posteriori (MAP) estimate instead of sampling. For example, the parameters for distance preference, μ_d and σ_d in Θ_k is updated by

$$\mu_d, \sigma_d = \arg\max_{\mu,\sigma} B_\psi(\psi) \prod_{i:\omega_i=k} \psi_{\text{dist}}(x_i, y_i, h_{z_i}; \mu, \sigma),$$

and similar for the other six parameters. In this work, we use non-informative prior for B_ψ, which is a zero-mean Gaussian with large variance for each of these four terms. We illustrate the learning process in Figure 4.3 shows an example of how the object affordances are refined progressively along with the sampled human poses.

Often, each object type is associated with one object affordance. Thus in our two tasks, we assume they are equivalent, i.e. $\omega_i = y_i$. Since object labels y_i are given during the training, we only perform Gibbs sampling on z_i, h_k and ψ_k, as summarized in Table 4.1.

However, because of the DP prior on affordances, our ILCRF can actually learn the number of affordances needed from the data, which we refer as 'affordance topics' (for its analogy to topic modeling for text data [133]). Each affordance topic can be shared across multiple object types while the affordance of each object type is now represented as a mixture of multiple topics. To demonstrate, we performed a learning experiment where ω_i is also sampled during the training.[3] We are able to learn 11 affordance topics for a total of 19 object categories in our scene arrangement dataset (see Sec. 5.2). Figure 4.2 shows some examples of learned topics and object affordance as a mixture of those topics. This ability is particularly useful when handling a large number of object types, as only a relatively small number of topics are learned yet they are able to represent the variety of object affordances.

4.2.3 ILCRF for Scene Arrangement

So far, we have presented ILCRF in the context of scene labeling task. Now we describe how to apply ILCRF to scene arrangement. While the two tasks have been studied with different approaches and algorithms in previous work, we show that they can be addressed in a unified model, ILCRF with the same definition on human poses and object affordances.

The arrangement task requires finding proper locations and orientations for placing new objects in a given scene. The scene is represented as an RGB-D point cloud and each object is represented by its appearance features and object class, included in x_i. Each y_i now denotes the placement (location and orienta-

[3] To make sure that objects from the same category have the same affordance, instead of sampling ω_i for each object instance i in Eq. (4.8), we sample ω_y for each object type y, i.e., $\omega_y = \omega \propto \frac{n^{\psi}_{-y,\omega}}{N+m-1+\alpha_\psi} \prod_{i:y_i=y} \psi_\omega(x_i, y_i, h_{z_i})$. Then the affordance of each object type y is given by $\frac{1}{S} \sum_s \psi_{\omega_y^{(s)}}(\cdot)$.

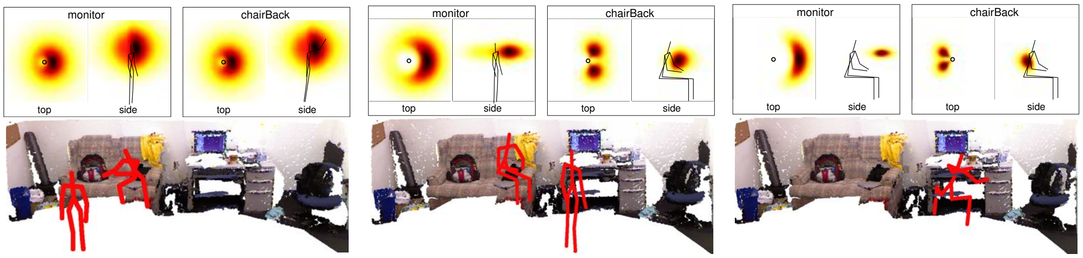
Algorithm 1: Labeling a new scene.

```
Data: $x_1, \ldots, x_N$: segment locations and appearance features.
    $\Psi$: learned object affordances.
Result: $y_1, \ldots, y_N$: labels for each segment.
Step 1: Initialization
$B_h \leftarrow$ a uniform distribution over all possible human configurations in the scene;
$\mathcal{H} \leftarrow$ randomly sample from $B_h$;
$z_1, \ldots, z_N \leftarrow$ random integers between 1 and $N$;
$\omega_1, \ldots, \omega_N \leftarrow$ same as $z_i$;
Step 2: Gibbs sampling
for each iteration $s$ do
    Sample $z_i$ using Eq. (4.6), $\forall i = 1, \ldots, N$;
    Sample $h_k$ using Eq. (4.7), $\forall k \in \{k | \exists z_i = k\}$;
    Sample $\omega_i^{(s)}$ using Eq. (4.8), $\forall i = 1, \ldots, N$;
end
Step 3: Labeling
For each segment $i$, use the histogram of $\omega_i^{(s)}$ as additional affordance features (along with object self and pairwise features). Then label all segments using the max-margin classifier in [1]
```

tion) of an object.[4]

During training, we learn object affordances same as in the labeling task as described in the last section. We also learn the object-object structure, E_Y, based on object co-occurence, as computed from the training data. In this task, ψ^{oo} is

[4] Since the object's placement is given either by x as in the labeling task or by y as in the arrangement task, our object affordance is defined as a function of both x and y as in Eq. (3.1).



(a) Iteration #1 (b) Iteration #10 (c) Iteration #100

Figure 4.3: **Learning object affordances**. This example shows the learned object affordances (top row, shown as heatmaps) and top sampled human configurations (bottom) through iterations. In Iteration#1, the affordance is only based on the prior B_ψ which is same for all objects. Thus, the sampled human poses also randomly appear in the scene. In later iterations, the affordances diverge to different but reasonable functions and so do the sampled humans based on these affordances.

defined as a multi-variate Gaussian distribution of the location and orientation difference between the two objects.

During testing, given the type of the object to be placed and learned object affordances, we sample human-object edges, human poses and placements. In the end, the predicted placement is the one sampled most as that represents the highest probability. The inference algorithms for both tasks are summarized in Alg. 1 and Alg. 2.

4.3 Related Work

Variants of Conditional Random Fields (CRFs) ([80]) have emerged as a popular way to model hidden states and have been successfully applied to many vision problems.

There are many models that enrich the structure of labels in CRFs. For ex-

Algorithm 2: Arranging a new scene.

```
Data: $x_1, \dots, x_N$: object types and appearance features.
      $\Psi$: learned object affordances.
Result: $y_1, \dots, y_N$: locations and orientations.
Step 1: Initialization
$B_h \leftarrow$ a uniform distribution over all possible human configurations in the scene;
$\mathcal{H} \leftarrow$ randomly sample from $B_h$;
$z_1, \dots, z_N \leftarrow$ random integers between 1 and $N$;
$\omega_1, \dots, \omega_N \leftarrow$ given by $x_i$;
$y_1, \dots, y_N \leftarrow$ randomly placed in the scene;
Step 2: Gibbs sampling
for each iteration $s$ do
    Sample $z_i$ using Eq. (4.6), $\forall i = 1, \dots, N$;
    Sample $h_k$ using Eq. (4.7), $\forall k \in \{k | \exists z_i = k\}$;
    Sample $y_i^{(s)}$ using Eq. (4.10), $\forall i = 1, \dots, N$;
end
Step 3: Placing
for $i = 1, \dots, N$ do
    $y_i \leftarrow \arg\max_y \sum_s \mathbf{1}\{y_i^{(s)} \in \text{neighborhood}(y)\}$;
end
```

ample, latent CRFs [103] assume that the overall label Y depends on a sequence of hidden states $(s_1, s_2, \dots, s_k)$ (see Fig. 2.1-bottom). This can be applied to object recognition (an object label is determined by its part labels) [123] and gesture recognition [140, 141]. Further, factorial (or dynamic) CRFs [130] substitute every label with a Markov network structure to allow structured labeling, especially for sequential data (such as labeling object and action simultaneously in video sequences [70, 74]). However, the labels and hidden states are discrete and take only finite number of values. In contemporary work, Bousmalis et al. [10] present a model that shares a name similar to ours, but is quite different. They estimate the correct number of values a latent node can take using Dirichlet processes in a way similar to augmenting hidden Markov models (HMM) to infinite HMM [4]. However, the number of hidden nodes is fixed in their model. In our model, we estimate the number of latent nodes, and even allow the labels to be continuous.

Some works impose a non-parametric Bayesian prior to the network's structure so that it can potentially generate as many nodes as needed. For example, Indian Buffet process [35] assumes the latent nodes and links are generated through Beta processes and the infinite factorial HMM [138] incorporates it to HMM to allow any number of latent variables for each observation. However, they are limited to binary Markov chains and do not consider different types of potential functions either. Thus these models are complementary to ours. Jancsary et al. [50] considers Gaussian CRFs on fixed number of nodes but unknown number of potential functions and proposes a non-parametric method to learn the number as well as parameters of each potential function. Unlike this work, our model can handle unknown number of *nodes* as well as *types of edges*.

Cast in the light of mixture models, mixtures of graphical models have been proposed to overcome the limited representational power that a single graphs often suffers. For example, Anandkumar et al. [3] propose a novel method to estimate a mixture of a finite number of discrete graphs from data. Other works consider a Dirichlet process mixture model over graphs so that the number of different graphical models is determined by the data [105, 46]. However, they are limited to Gaussian graphical models and do not consider latent variables.

CHAPTER 5

APPLICATIONS

In this chapter, we instantiate our ILCRF model in two applications: object detection and object arrangement. Given a room, the first task requires to identify existing objects, and the second task asks to place more designated objects in proper locations and orientations.

In our applications, the scenes (including objects/furnitures) are perceived as point-clouds (Fig. 5.8), either generated from 3D models in synthetic datasets or obtained using Microsoft Kinect camera in real datasets.

5.1 Scene Labeling Results

In this experiment, the goal is to label each segment in a given room with correct class, such as table, chair-back, keyboard, etc.

Dataset. We used the Cornell RGB-D indoor dataset [73, 1] for our experiments. This data consists full-scene RGB-D point clouds of 52 offices and homes obtained from 550 RGB-D views. The point-clouds are over-segmented, and the goal is to label these segments with object labels and attribute labels. Each segment can have multiple attribute labels but has only one object label. The attribute labels are: {*wall, floor, flat-horizontal-surfaces, furniture, fabric, heavy, seating-areas, small-objects, table-top-objects, electronics*} and the object labels are: {*wall, floor, tableTop, tableDrawer, tableLeg, chairBackRest, chairBase, chairBack, monitor, printerFront, printerSide, keyboard, cpuTop, cpuFront, cpuSide, book, paper, sofaBase, sofaArm, sofaBackRest, bed, bedSide, quilt, pillow, shelfRack, laptop*}.

Table 5.1: **Object and Attribute Labeling Results.** The table shows average micro precision/recall, and average macro precision and recall for 52 scenes. Computed with 4-fold cross-validation.

Algorithm	Object Labeling			Attribute Labeling			
	micro	macro		micro		macro	
	P/R	prec	recall	prec	recall	prec	recall
Chance	5.88	5.88	5.88	12.50	12.50	12.50	12.50
Affordances	31.38	16.33	15.99	50.93	34.06	42.02	28.02
Appearance	67.24	53.31	50.48	81.81	60.85	73.30	52.36
Afford. + Appear.	68.63	55.69	52.86	83.04	63.95	78.85	56.00
Object context [1]	78.72	68.67	63.72	85.52	70.98	80.04	63.07
ILCRF	**78.86**	**71.14**	**65.07**	**85.91**	**73.51**	**82.76**	**69.22**

Baselines. We perform 4-fold cross-validation where we train the model on data from three folds and tested on the fourth fold of unseen data. Table 5.1 presents the results for object labeling and attribute labeling. In order to study the effects of different algorithms, we compare with the following algorithms:

(a) Affordances (Human Context). This is our affordance and human configurations information being used in prediction, without any object context.

(b) Appearance. We run versions with both local image and shape features [1].

(c) Afford. + Appear. It combines the affordance and appearance features.

(d) Object context. We use the learning algorithm presented in [1] that uses Markov Random Field with log-linear node and pairwise edge potentials.

(e) Our ILCRF. Here we combine the human context (from affordances and human configurations) with object-object context. In detail, we append the node features of each segment with the affordance topic proportions derived from

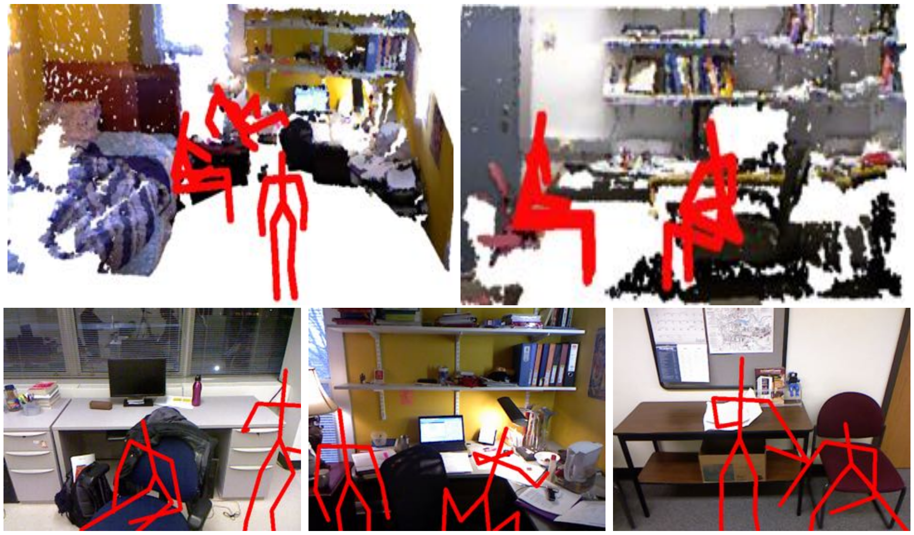


Figure 5.1: **Top sampled human poses in different scenes.** The first two are from stitched point-cloud from multiple RGB-D views, and the last three scenes are shown in RGB-D single views.

the learned object-affordance topics and learn the semantic labeling model as described in [1].

Evaluation metrics. We report precision and recall using both micro and macro aggregation. Since we predict only one label for each segment in case of predicting object labels, our micro precision and recall is the same as the percentage of correctly classified segments (shown as 'P/R' in Table 5.1). The macro precision and recall are the average of precision and recall of all classes respectively.

Results. Table 5.1 shows that our algorithm performs better than the state-of-the-art in both object as well as attribute labeling experiment. Our approach is able to predict the correct labels for majority of the classes as can be seen from the strong diagonal in the confusion matrices. We discuss our results in the light of the following questions.

Are the sampled human poses meaningful? Being able to hallucinate sensible human poses is critical for learning object affordances. To verify that our algorithm can sample meaningful human poses, we plot a few top sampled poses in the scenes, shown in Fig. 5.1. In the first home scene, some sampled human poses are sitting on the edge of the bed while others standing close to the desk (so that they have easy access to objects on the table or the shelf-rack). In the next office scene (Fig. 5.1-b), there is one L-shaped desk and two chairs on each side. It can be seen that our sampled human poses are not only on these chairs but also with correct orientation. Also, as can be seen in Fig. 4.3-c, our algorithm successfully identifies the workspaces in the office scene. Note that these poses naturally explain why the monitors, keyboards and CPUs are arranged in this particular way. It is these correctly sampled human poses that give us the possibility to learn correct object affordances.

Are the discovered affordances meaningful? During training, we are given scenes with the objects and their labels, but not humans. Our goal is to learn object affordance for each class. Fig. 5.2 shows the affordances from the top-view and side-view respectively for typical object classes. Here the X-Y dimensions of the box are 5m×5m, and the height axis's range is 3m. The person is in the center of the box. From the side views, we can see that for objects such as wall and cpuTop, the distributions are more spread out compared to objects such as floor, chairBase and keyboard. This is because that that chairBase is often associated with a sitting pose at similar heights, while CPUs can either be on the table or on the floor. While this demonstrates that our method can learn meaningful affordances, we also observe certain biases in our affordances. For example, the wall is more to the front as compared to the back, and monitor is biased to the side. We attribute to the limited data and imperfect generation of valid human skeletons. Note that while the affordance topics are unimodal, the affordance for each objects is a mixture of these topics and thus could be multi-modal and

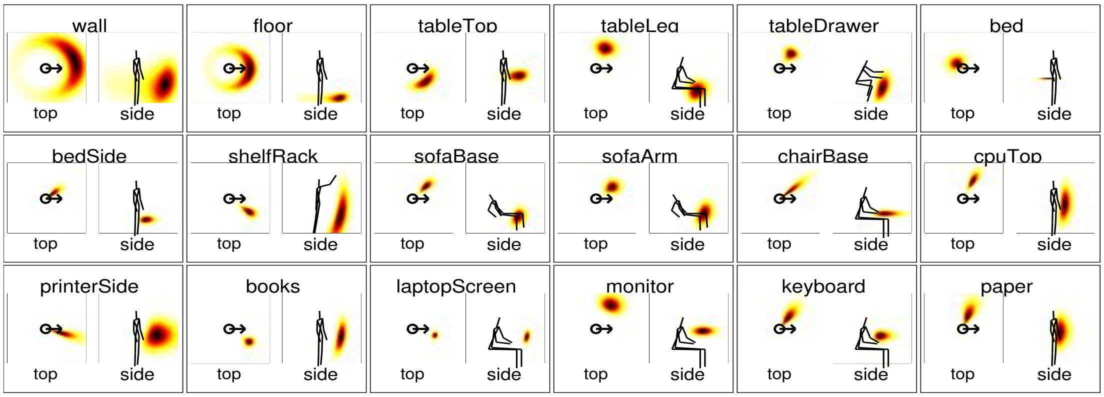


Figure 5.2: **Examples of learned object-affordance topics.** An affordance is represented by the probability distribution of an object in a 5×5×3 space given a human pose. We show the projected top views and side views for different object classes.

more expressive.

Can we obtain object-object relations from object affordances? Since objects are related to humans, it turns out that we can infer object-object spatial relations (and object co-occurences) from the human-object relations. For example, if we convolve keyboard-human and human-monitor relations, we obtain the spatial relations between keyboard and monitor. More formally, we compute the conditional distribution of one object's location x_i (with type y_i) given another object's location x_j (with type y_j) as,

$$\begin{aligned} P(x_i|x_j) &= \int P(x_i|h)P(h|x_j)dh \\ &\propto \int \psi_{ho}(x_i, y_i, h)\psi_{ho}(x_j, y_j, h)B_h(h)dh \end{aligned}$$

Some examples are shown in Fig. 5.3. We can find that many object-object relationships are recovered reasonably from our learned affordances. For example, given a keyboard, a monitor is likely to be found in front of and above it while tableTop at the same height as it (sometimes above it as the keyboard is often in a keyboard-tray in offices). In home scenes, given a bed, we can find

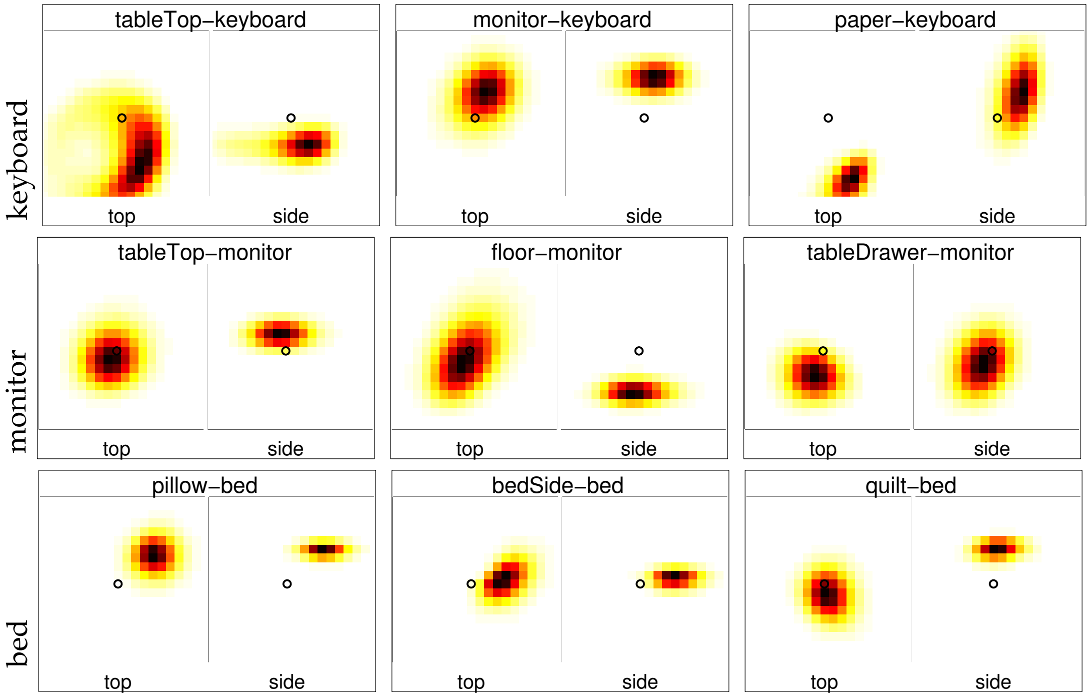
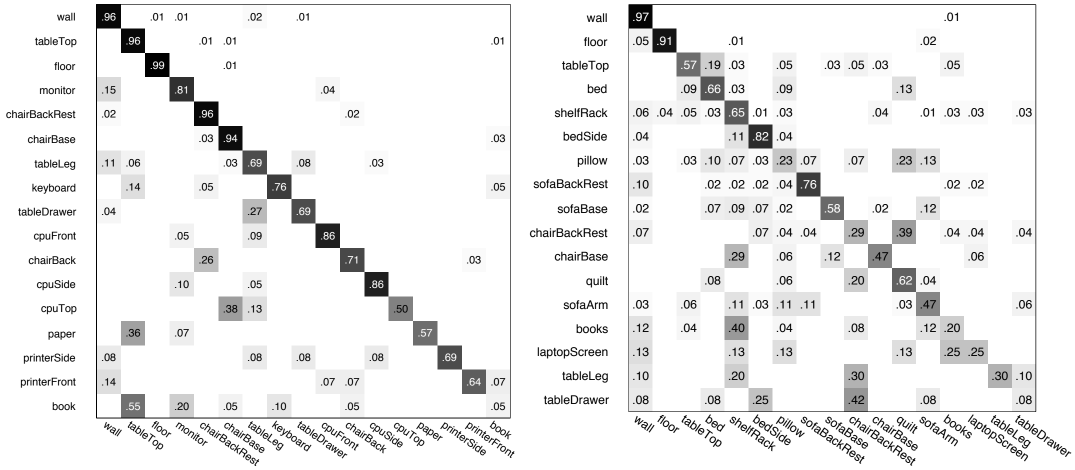


Figure 5.3: **Object-object context obtained from our learned human context.** Each pair of the top- and side-view of a heatmap with the title of 'obj1-obj2' shows the distribution of obj1 given obj2 at the center facing right. For example, in the first row the keyboard is in the center of the image and the heat-maps show the probability of finding other related objects such as table top, monitor, etc.

a pillow on the head of the bed, quilt right above the bed and bedSide slightly below it. This supports our hypothesis that object-object relations are only an artifact of the hidden context of human-object relations. It also demonstrates that we can efficiently model $O(n^2)$ object-object relations for n objects using only $O(n)$ human-object parameters.

Does human context helps in scene labeling? Table. 5.1 shows that the affordance topic proportions (human context) as extra features boosts the labeling performance. First, when combining human context with the image- and shape-features, we see a consistent improvement in labeling performance in all evaluation metrics, regardless of the object-object context. Second, when we add



	wall	tableTop	floor	monitor	chairBackRest	chairBase	tableLeg	keyboard	tableDrawer	cpuFront	chairBack	cpuSide	cpuTop	paper	printerSide	printerFront	book
wall	.96		.01	.01			.02		.01								
tableTop		.96			.01	.01											.01
floor			.99			.01											
monitor	.15			.81						.04							
chairBackRest	.02				.96						.02						
chairBase					.03	.94											.03
tableLeg	.11	.06				.03	.69		.08			.03					
keyboard		.14			.05			.76									.05
tableDrawer	.04						.27		.69								
cpuFront				.05			.09			.86							
chairBack					.26						.71					.03	
cpuSide				.10			.05					.86					
cpuTop						.38	.13						.50				
paper		.36		.07										.57			
printerSide	.08						.08		.08			.08			.69		
printerFront	.14									.07	.07					.64	.07
book		.55		.20		.05		.10			.05						.05

	wall	floor	tableTop	bed	shelfRack	bedSide	pillow	sofaBackRest	sofaBase	chairBackRest	chairBase	quilt	sofaArm	books	laptopScreen	tableLeg	tableDrawer
wall	.97													.01			
floor	.05	.91			.01								.02				
tableTop			.57	.19	.03		.05		.03	.05	.03			.05			
bed			.09	.66	.03		.09					.13					
shelfRack	.06	.04	.05	.03	.65	.01	.03				.04		.01	.03	.03		.03
bedSide	.04				.11	.82	.04										
pillow	.03		.03	.10	.07	.03	.23	.07		.07		.23	.13				
sofaBackRest	.10			.02	.02	.02	.04	.76						.02	.02		
sofaBase	.02			.07	.09	.07	.02		.58		.02		.12				
chairBackRest	.07					.07	.04	.04		.29		.39		.04	.04		.04
chairBase					.29		.06		.12		.47				.06		
quilt				.08			.06			.20		.62	.04				
sofaArm	.03		.06		.11	.03	.11	.11				.03	.47				.06
books	.12		.04		.40		.04			.08			.12	.20			
laptopScreen	.13				.13		.13					.13		.25	.25		
tableLeg	.10				.20					.30						.30	.10
tableDrawer	.08			.08		.25				.42			.08				.08

Figure 5.4: **Confusion matrices** for office dataset (left) and home dataset (right) using our ILCRF model.

object-object context, the performance is further boosted in the case of office scenes and improves marco precision for home scenes. This indicates that there is some orthogonality in the human-object context and object-object context. In fact, adding object-object context to human-object context was particularly helpful for small objects such as keyboards and books that are not always used by humans together, but still have a spatial correlation between them.

We also show the confusion matrices in Fig. 5.4. We found that while our algorithm can distinguish most of the objects, it sometimes confuses objects with similar affordance. For example, it confuses pillow with quilt and confuses book and paper with tableTop. Similarly, it confuses cpuTop with chairBase because the CPU-top (placed on the ground) could also afford sitting human poses!

5.2 Scene Arrangement Results

In this experiment, the goal is to find proper locations and orientations for placing one or multiple objects in a given room.

Dataset. We test on a synthetic dataset and a real dataset. We downloaded 20 different rooms from Google 3D Warehouse, including six living rooms, seven kitchens and seven offices. All these scenes are commonly seen in the real world and have different layouts and sizes. We also collected 47 different objects from 19 categories for arranging: { *book, clean tool, laptop, monitor, keyboard, mouse, pen, decoration, dishware, pan, cushion, TV, desk light, floor light, utensil, food, shoe, remote control, and phone*}. Every room is assigned to three to five subjects (not associated with the project) to manually label the arrangements of 10 to 30 objects. In total, we have 67 different labeled arrangements for 20 rooms.

We also test on *real scenes* from [55] using the learned model from the synthetic dataset. The real dataset consists of five empty offices and apartments, each of which is asked to arrange 4, 18, 18, 21 and 18 number of objects respectively.

Experimental setup. For the synthetic dataset, we conduct 5-fold cross validation on 20 rooms such that the four test rooms are new to the algorithms. We consider two different testing scenarios, where the test room is either: *partially-filled* and the task is to arrange one new type of objects (may have multiple instances); or *empty* (with only furnitures) and the task is to arrange multiple types of objects.

Baselines. We compare all the following methods:

Table 5.2: **Scene arrangement results** on partially-filled scenes and empty scenes in synthetic dataset, evaluated by the location and height difference to the labeled arrangements.

Algorithms	partially-filled scenes		empty scenes	
	location (m)	height (m)	location (m)	height (m)
Chance	2.35±0.23	0.41±0.04	2.31±0.23	0.42±0.05
Obj. [54]	1.71±0.23	0.13±0.02	2.33±0.17	0.44±0.04
CRF	1.69±0.05	0.12±0.01	2.17±0.07	0.39±0.01
ILCRF-H [54]	1.48±0.18	0.11±0.01	1.65±0.20	0.12±0.01
Human+obj [54]	1.44±0.18	**0.09** ±0.01	1.63±0.19	0.11±0.01
ILCRF-Aff.	1.59±0.06	0.14±0.01	1.60±0.06	0.15±0.01
ILCRF-NSH	1.64±0.05	0.15±0.01	1.77±0.06	0.16±0.01
FLCRF	1.55±0.06	0.12±0.01	1.63±0.06	0.14±0.01
ILCRF	**1.33±0.19**	**0.09±0.01**	**1.52±0.06**	**0.10±0.01**

Table 5.3: **Scene arrangement results** on 5 real empty scenes (3 offices and 2 apartments). **Co**: % of semantically correct placements, **Sc**: average score (0-5).

	office1		office2		office3		apt1		apt2		AVG	
	Co	Sc	Co	Sc	Co	Sc	Co	Sc	Co	Sc	Co	Sc
Obj.	100	4.5	100	3.0	45.0	1.0	20.0	1.8	75.0	3.3	68.0	2.7
ILCRF-H	100	**5.0**	100	4.3	91.0	4.0	74.0	3.5	88.0	4.3	90.0	4.2
Human+obj	100	4.8	100	4.5	92.0	4.5	89.0	4.1	81.0	3.5	92.0	4.3
ILCRF	**100**	**5.0**	**100**	**4.6**	**94.0**	**4.6**	**90.0**	**4.1**	**90.0**	**4.4**	**94.8**	**4.5**

1) Chance. Objects are placed randomly in the room.

2) Obj. We use heuristic object-object spatial relations to infer placements in sequence (not jointly). [1]

[1] We model the relative location/orientation between any pair of object types as Gaussian

3) CRF, a ILCRF with only object-object edges (i.e. (y_i, y_j)), without latent human nodes.

4) ILCRF-H, a ILCRF with only human-object edges (i.e., (y_i, h_{z_i})), without considering object relations.

5) Human+obj, a heuristic way combining object context and human context. It linearly combines the inferred distributions of arrangements $\mathcal{Y}$ from Obj. and from ILCRF-H, and select the maximum. Our ILCRF, on the other hand, incorporate the two relationships during the inference, not after.

6) ILCRF-Aff, a ILCRF with only one type of edge, i.e., one shared affordance across all object classes.

7) ILCRF-NSH, a ILCRF with with non-sharing latent human nodes. Each object is assigned with its own human node, i.e. $z_i = i$ for each y_i, similar to *hidden CRFs* in Fig. 2.1. While this model can still affect the object arrangements through possible human poses (e.g., monitor will be placed near any sitting area), it cannot capture phenomena of objects sharing the same human pose, such as a monitor and a keyboard being placed together. ILCRF achieves this ability of sharing latent nodes through the clustering effect (on z_i's) inherited from DPs.

8) FLCRF, a ILCRF with fixed/finite number of latent nodes (same number of human poses across all scenes). It requires a good estimate on the number of human poses, and the optimal number may vary for rooms of different types or sizes.

9) ILCRF, our full model.

distributions with parameters learned from training data. For placing a new object, a reference object (already placed in the room) is selected with the smallest variance and then sample the new object's location/orientation from the Gaussian distributions.

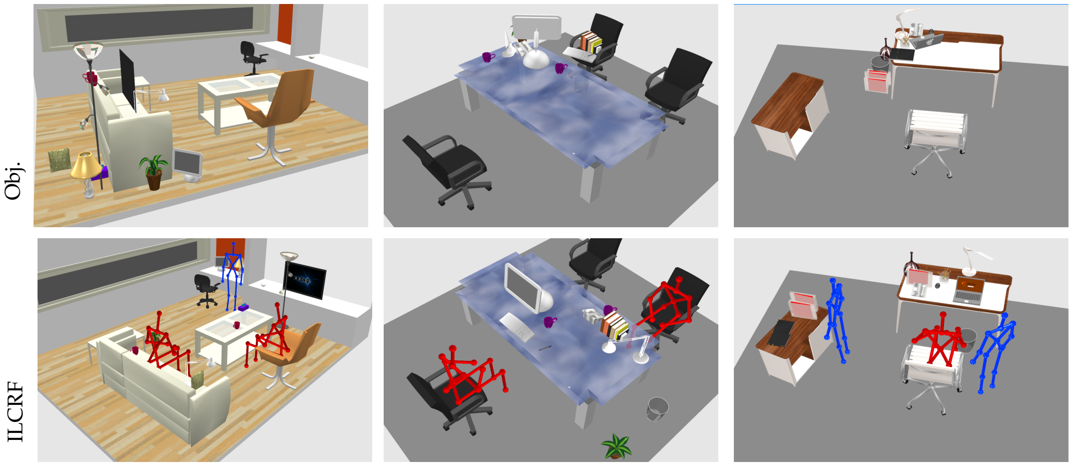


Figure 5.5: Results of arranging empty rooms by using object-object relationships only (top) and by ILCRFs (bottom).

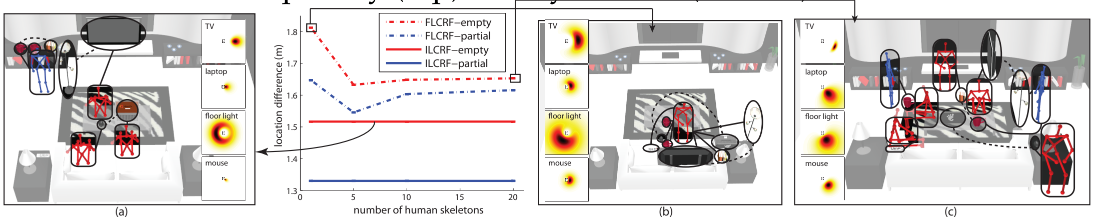


Figure 5.6: Results of FLCRF with different number of human poses versus ILCRF. We also show examplar sampled CRFs and learned object affordances (in top-view heatmaps) by different methods.

Evaluation metrics. For synthetic datasets, the predicted arrangements are evaluated by two metrics, same as in [54]: location difference and height difference (in meters) to the labeled arrangements (averaged over different object types across all test rooms). The results are shown in Table 5.2.

Results of arranging empty real scenes are evaluated by human subjects: Each arrangement is measured by the percentage of predicted locations that are semantically correct and a score of the overall arrangement between 0 and 5, labeled by two human subjects that are not associated with this project. Results are presented in Table 5.3.

Results. Results in Table 5.2 demonstrate, same as the previous experiment, that modeling human context does improve the performance: On average, the location and height difference are reduced from 1.69m (2.17m) and .12m (.39m) when modeling object context only using CRF, to 1.33m (1.52m) and .09m (.10m) when modeling both human and object context using ILCRF, in arranging partially-filled (empty) scenes. Even methods that use non-sharing skeletons (ILCRF-NSH) and finite skeletons (FLCRF) achieve better results than CRF. We also visually compare some predicted arrangements for empty rooms (Fig. 5.5), where using object relations only often leads to over-crowded arrangements (especially in empty rooms) or inconvenient/inaccessible locations due to the lack of human context. In the following, we study how well the latent human context is modeled by ILCRF.

Why do we need handle unknown number of human poses? The advantage of using DP mixture models in ILCRF is being able to determine the number of human poses from the data instead of guessing manually. We investigate this in in Fig. 5.6. We compare ILCRF with the FLCRF where the number of human poses varyies from 1 to 20.

While having five poses in FLCRF gives the best result, it is still outperformed by ILCRF. This is because scenes of different sizes and functions prefer different number of skeletons. If we force all scenes using only one human pose, the learned object affordances will have large variances because all objects in the scene attempting to relate to one human, e.g., in Fig. 5.6-(b). If we force all scenes using a large number of human poses, say 20 per scene, the model will overfit in each scene and leading to meaningless affordances, e.g., Fig. 5.6-(c). Therefore, having the correct number of latent human nodes in CRFs is crucial for learning good object affordances as well as for inferring reasonable arrangements across

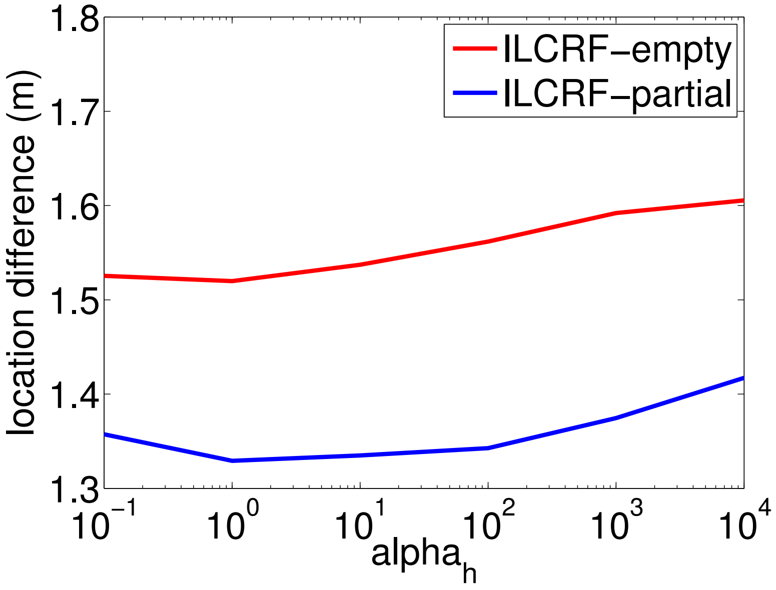


Figure 5.7: The average performance of ILCRF with different hyper-parameter α_h.

diverse scenes (Fig. 5.6-a).

How sensitive is ILCRF to the number of human poses? The parameter α_h in ILCRF controls the probability of selecting a new human pose and thus can be viewed as a counterpart of K (the fixed number of human poses) in FLCRF. However, unlike FLCRF, ILCRF is much less sensitive to this parameter, as shown in Fig. 5.7 where its performance does not vary much for α_h from 0.1 to 10^4. Therefore, ILCRF does not rely on either informative prior knowledge or a careful hand-picked value of α_h to achieve high performance.

5.3 Robotic Experiment.

Robotic simulation experiment. In order to study how the desired placements are affected by the robot constraints, we tested arranging these synthetic scenes

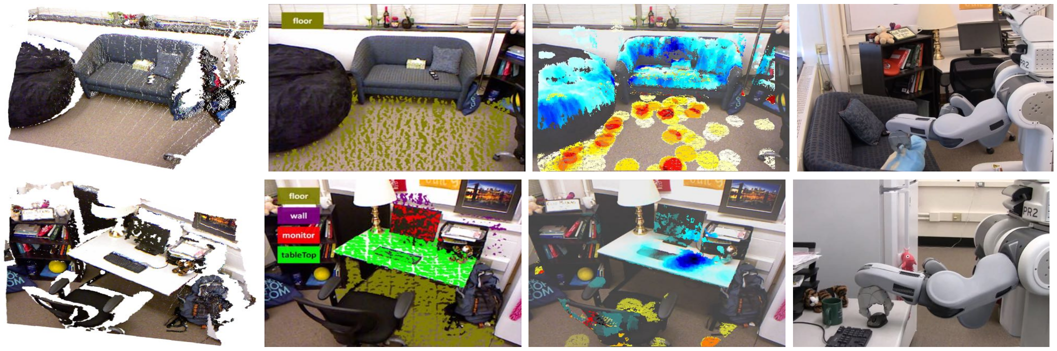


Figure 5.8: **Robotic experiment** (from left to right): (a) A given scene is perceived as a RGB-D point-cloud; (b) The robot uses ILCRF to detect objects; (c) The robot uses ILCRF to infer possible human poses (shown in red heatmaps) and possible placements (shown in blue heatmaps) for placing a cushion (top) and a mouse (bottom) in the scene; (d) The robot successfully places objects in the predicted locations.

using Kodiak (PR2) in simulation. Please refer to [57] for more details on results.

Arrange Real Scenes. We apply the ILCRF to our Kodiak PR2 robot to perform the scene arrangement in practice. We test our system on a small set of objects (a cushion, mouse and mug) in a given scene (Fig. 5.8). The system works as follows: (a) The robot perceives the environment as point clouds; (b) It hallucinate human poses and detect objects using ILCRF; (c) When asked to place a new object, it hallucinate human poses as well as sample the object's locations. The most sampled location will be the final prediction; (d) The robot executes the arrangement by placing the object at the predicted location. To see PR2 arranging the scene in action (along with code and data), please visit: `http://pr.cs.cornell.edu/hallucinatinghumans`

CHAPTER 6

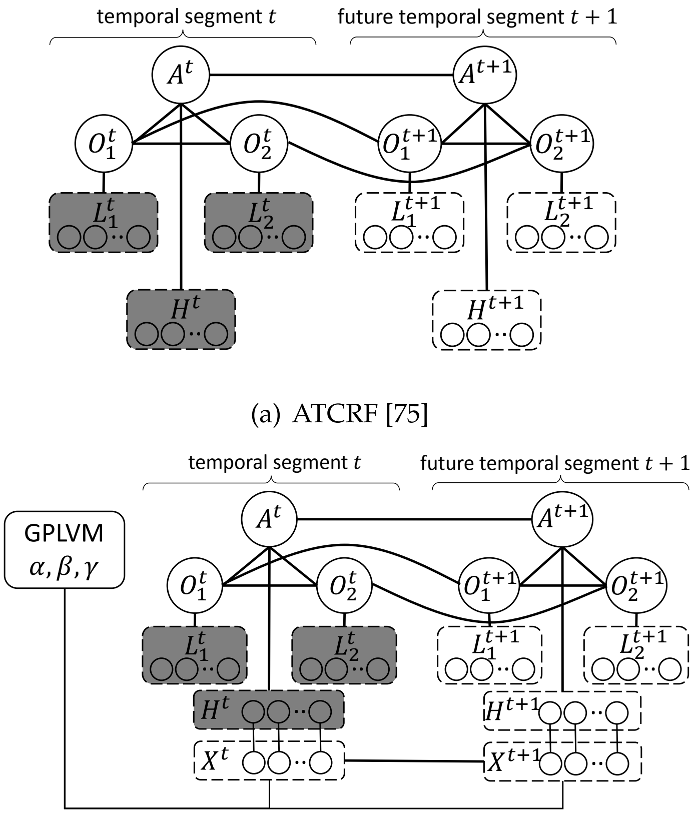
MODELING HUMAN DYNAMICS

So far, we have shown that static human poses can be hallucinated in a given scene. In this chapter, we further show that dynamic human motions can also be effectively modeled and thus anticipated using our proposed latent CRFs.

For robots, the ability to model human configurations and temporal dynamics is crucial for the task of anticipating future human activities, yet requires conflicting properties: On one hand, we need a detailed high-dimensional description of human configurations to reason about the physical plausibility of the prediction; on the other hand, we need a compact representation to be able to parsimoniously model the relations between the human and the environment.

We therefore propose a new model, GP-LCRF, which admits both the high-dimensional and low-dimensional representation of humans. It assumes that the high-dimensional representation is generated from a latent variable corresponding to its low-dimensional representation using a Gaussian process. The generative process not only defines the mapping function between the high- and low-dimensional spaces, but also models a distribution of humans embedded as a potential function in GP-LCRF along with other potentials to jointly model the rich context among humans, objects and the activity.

In the following, we first give an overview of our problem and motivation in Sec. 6.1. We then introduce our model GP-LCRF and its learning and inference



(a) ATCRF [75]

(b) Our GP-LCRF

Figure 6.1: Graphical representations of the original ATCRF [75] and our GP-LCRF. Our model adds latent low-dimensional nodes $\mathcal{X}$ to the model, which are related to the original high-dimensional human configuration nodes $\mathcal{H}$ through Gaussian Process latent variable model with parameters α,β, γ. Shaded nodes indicate observations. In both models, temporal segment t is given for anticipation with observed human poses $\mathcal{H}^t$ and object locations $\mathcal{L}^t$, and the goal is to infer the next segment $t + 1$ where nothing is observed.

algorithm in Sec. 6.2. Finally we present our experimental results in Sec. 6.3.

6.1 Overview

We define the anticipation task as follows: Given an RGB-D video of a human interacting with the surrounding environment, our goal is to predict what will happen to the environment in a time span in terms of the next sub-activity label, object affordance labels and object trajectories. Modeling future human configurations, in the context of the activity and the environment, is a key ingredient for a good anticipation.

Human configuration has two sides of nature: It is *high-dimensional* in terms of the degree of freedom a human body possesses. One would need the location and orientation of each joint of a human skeleton to fully describe a static human pose, and the velocities and accelerations to describe a sequence of human motions. The high-dimensional representation is a guarantee for generating realistic human poses/motions. We need it to perform (self-)collision detection, inverse kinematics and path planning.

However, most dynamic human behaviors are intrinsically *low-dimensional*, as our arms and legs operate in a coordinated way and they are far from independent with each other. Many daily behaviors such as walking and jumping have been represented in low-dimensional space [36, 112]. The low-dimensional representation is a requisite for probabilistic models of human motions. The distribution of human poses can be used to synthesize or predict new poses.

Our main idea is to keep both the high- and low-dimensional representation of human dynamics in anticipating human activities. Our learning model thus has two parts:

Learning low-dimensional human dynamic distributions. For each human pose, indexed by i, we use h_i to denote its high-dimensional and x_i for its corresponding low-dimensional representation. The correspondence is specified by a mapping function, i.e. $h_i = f(x_i)$. Additionally, we are also interested in associating the mapping with a probabilistic model, so that we can generate new human dynamics (x_i, h_i) from the learned distribution. Hence, the objective of this part is to learn the parameters of f as well as a likelihood function $L(x_i, h_i)$ from the training data $\{h_i\}$.

Modeling the spatial and temporal context of human activities. We use a graphical model, following [75] to capture the three important aspects in a human activity—sub-activities $\mathcal{A}$, objects $\mathcal{O}$ and humans $\mathcal{H}$. Given a video segment t,[1] each entity is represented by a node in the graph modeling its prior distribution and the edges in the graph model their relations, as shown in Fig. 6.1. The whole video is a repetition of such a graph. Edges between consecutive segments are used to model temporal dynamics. In particular, for each human pose in segment t, in addition to the original human node h_i^t, we add a low-dimensional latent node x_i^t. The edges between h_i^t and $\mathcal{O}^t$ or $\mathcal{A}^t$ are used to capture human-object and human-activity relations, while the edges between x_i^{t-1} and x_i^t are for modeling the human dynamics. This graphical model thus defines a joint distribution $P(\mathcal{A}, \mathcal{O}, \mathcal{H}, \mathcal{X})$ as a product of parameterized edge potentials. We learn those parameters from labeled data and then sample future segments from this distribution for anticipation.

By combining these two parts, our proposed GP-LCRF possesses many advantages: First, we can now use the context of high-dimensional data that is difficult to model for a traditional CRF. Second, as the low-dimensional representation is modeled as latent nodes and the mapping is learned in an unsupervised way, our model does not require any extra label/data to learn. Third, being able to learn the distribution of the low-dimensional latent node makes our

[1] Frames of a video are grouped into temporal segments, and each segment spans a set of contiguous frames, during which the sub-activity and object affordance labels do not change.

GP-LCRF a generative model that suits the anticipation problem. Before presenting our GP-LCRF, we first briefly review the background of the two parts in the following.

6.2 GP-LCRF for Human Activity Anticipation

We propose a model, GP-LCRF, that learns a probabilistic mapping between the high- and low-dimensional representation of human dynamics based on Gaussian processes. Then it embeds the compactly represented humans as latent nodes in a CRF to capture a variety of context between the human, objects and activities.

Our GP-LCRF introduces a layer of latent nodes in a CRF: each node h_i is now linked to a latent node x_i and their relation is defined by a GPLVM with parameters (α, β, γ). Because latent nodes have much lower dimensions, we can model the edges between latent nodes (e.g., (x_i^t, x_i^{t+1})) instead of attempting to capture it with high-dimensional nodes directly. (The high-dimensionality of the human nodes makes the edge distribution ill-conditioned.) Figure 6.1 shows the corresponding graphical model.

GP-LCRF differs from other latent CRFs in two aspects:

Prior. We adopt GPLVM to impose a Gaussian process prior on the mapping and a ℓ_2-norm prior on the latent nodes. This prior regulates the mapping so that the high-dimensional human configurations h_i that are close in the original space would remain close in the latent space x_i. This property of local distance preservation is very desirable in many applications, especially for time series

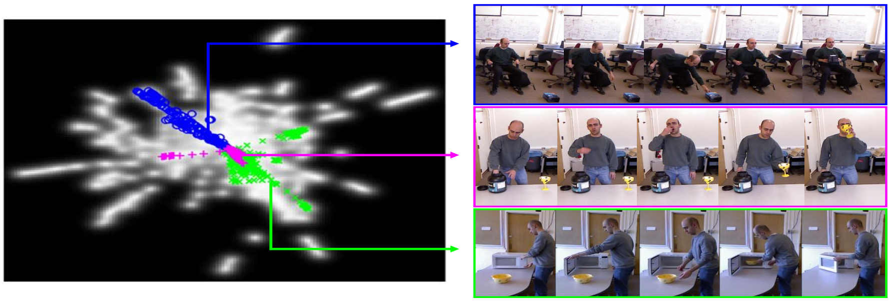


Figure 6.2: An example of the learned mapping from the high-dimensional human configurations to a 2-dimensional space. The intensity of each pixel x visualizes the its probability $-\frac{D}{2}\ln\sigma^2(x) - \frac{1}{2}\|x\|^2$. We also plot the projected 2D points for different activities in different colors. We can see that human configurations from the same activity are mapped to a continuous area in the 2D space while the ones from different activities are separated.

analysis.

Non-parametric. In many latent CRFs, the values of latent nodes are discrete and finite [103]. Some other works consider a non-parametric Bayesian prior over the latent values but they do not handle dimensionality reduction. In our GP-LCRF, the latent space is completely determined by the training data, making it more adaptive to various applications.

6.2.1 Background: Dimensionality Reduction with a Gaussian Processes

Consider a general setting for regression problems: Our goal is to learn a mapping $h = f(x)$ for a set of N training pairs (x_i, h_i). However, from a Bayesian point of view, instead of mapping to one point, a Gaussian process (GP) "maps" x to a distribution of h. Let μ be the mean of the training data $\mu = \sum h_i/N$, and let

$H_k = [h_{1,k} - \mu_k, \dots, h_{N,k} - \mu_k]^T$ be the feature vectors of the k^{th} dimension. In a GP model, H_k can be viewed as one sample from a multivariate Gaussian distribution:

$$P(H_k|\{x_i\}) = \frac{1}{(2\pi)^{N/2}|K|^{1/2}} \exp(-\frac{1}{2} H_k^T K^{-1} H_k) \tag{6.1}$$

K is the covariance matrix of all inputs x_i. We can use many non-linear kernel functions, such as the popular "RBF kernel", to admit non-linear mappings:

$$K_{i,j} = k(x_i, x_j) = \alpha \exp(-\frac{\gamma}{2} ||x_i - x_j||^2) + \delta_{x_i,x_j} \beta^{-1}$$

where δ_{x_i,x_j} is the Kronecker delta. Using this kernel means that the two points, x_i and x_j, that are correlated in the latent space, will also be highly correlated after the mapping. The parameter α imposes a prior on how much the two points are correlated, γ is the inverse width of the similarity function, and β reflects how noisy the prediction is in general.

In a more general setup, only $h_1, \dots h_N$ are given and the goal is to determine the mapping function f as well as the corresponding x_i. This can be solved using Gaussian process latent variable models (GPLVM) proposed in [82]. GPLVM maximizes the likelihood of the training data, based on Eq. (6.1), to learn the parameters of the kernel function (γ, α, β) and the latent variables $x_1, \dots, x_N$.

Since GPLVM provides a probabilistic model of nonlinear mappings and generalizes well for small datasets, it has been extended to model human motions in many works. For example, it is integrated with a dynamical model to capture dynamical patterns in human motions [139] so that it can provide a strong prior for tracking human activities [137, 147, 30]. In this work, we also adopt GPLVM as a dimensionality reduction approach, however, our goal is to incorporate this with Latent CRFs to model high-dimensional human motions and *rich context* in the environment at the same time.

6.2.2 Likelihood of GP-LCRF

As shown in Fig. 6.1-(b), a GP-LCRF is a repetition of small graphs (one per each temporal segment). A segment t contains one sub-activity label node $\mathcal{A}^t$, object affordance label nodes $\mathcal{O}^t = \{O_i^t\}$, object location nodes $\mathcal{L}^t = \{L_i^t\}$, high-dimensional human configurations $\mathcal{H}^t = \{h_i^t\}$ and low-dimensional human representations $\mathcal{X}^t = \{x_i^t\}$.

Following the independence assumptions imposed by the edges in the graph, the likelihood of one temporal segment $P(\mathcal{A}^t, \mathcal{O}^t, \mathcal{X}^t | \mathcal{L}^t, \mathcal{H}^t)$ is,

$$\begin{aligned} P_t &\propto \psi(\mathcal{A}^t, \mathcal{H}^t) \prod_i \psi(\mathcal{A}^t, o_i^t) \prod_i \psi(o_i^t, L_i^t) \\ & \prod_{(i,j)} \psi(o_i^t, o_j^t) \prod_i \phi(x_i^t, h_i^t) \end{aligned} \tag{6.2}$$

where the first four terms capture human-activity relations, object-activity relations, object affordances and object-object relations respectively. These potentials are parameterized as log-linear functions of feature vectors [75]. We define the last term, potential of the mapping between x_i and h_i as the likelihood derived from GPLVM:

$$\phi(x_i, h_i) = \exp L(x_i, h_i) \tag{6.3}$$

$$L(x, h) = -\frac{||h - f(x)||^2}{2\sigma^2(x)} - \frac{D}{2} \ln \sigma^2(x) - \frac{1}{2}||x||^2 \tag{6.4}$$

where

$$\begin{aligned} f(x) &= \mu + H^T K^{-1} \mathbf{k}(x) \\ \sigma^2(x) &= k(x, x) - \mathbf{k}(x)^T K^{-1} \mathbf{k}(x) \\ \mathbf{k}(x) &= [k(x, x_1), \ldots, k(x, x_N)]^T \end{aligned}$$

The three terms in $L(x, h)$ measure the discrepancy between the given h and the prediction $f(x)$, the uncertainty of the prediction, and the prior of the latent value x.

We now consider the temporal relations between the two consecutive temporal segments $t-1$ and t:

$$P_{t-1,t} \propto \psi(\mathcal{A}^t, \mathcal{A}^{t-1}) \prod\nolimits_i \psi(o_i^t, o_i^{t-1}) \phi(x_i^t, x_i^{t-1}) \tag{6.5}$$

where the first two terms capture the temporal transitions of sub-activity labels and object affordance labels. They are also parameterized as log-linear functions of features [75]. We define the last term, the temporal transitions of latent nodes, as Gaussian distributions:

$$\phi(x_i^t, x_i^{t-1}) \propto \mathcal{N}(||x_i^t - x_i^{t-1}||^2; 0, 1) \tag{6.6}$$

Hence, the overall likelihood of a GP-LCRF is

$$L_{\text{GP-LCRF}} \propto \prod\nolimits_{t=1}^{T} P_t \times \prod\nolimits_{t=2}^{T} P_{t-1,t} \tag{6.7}$$

Using this function, we learn the parameters by maximize the training data's likelihood and to predict the future activities and human dynamics by sampling from this distribution.

6.2.3 Learning

During training, given all observations ($\mathcal{H}$ and $\mathcal{L}$) and labels ($\mathcal{A}$ and $\mathcal{O}$), our goal is to learn the parameters in every potentials *and latent nodes $\mathcal{X}$* by maximizing the likelihood in Eq. (6.7), which can be written into two parts:

$$\begin{aligned}L_{\text{GP-LCRF}} &= \prod_t \Bigg(\psi(\mathcal{A}^t, \mathcal{H}^t) \prod_i \psi(\mathcal{A}^t, o_i^t)\psi(o_i^t, L_i^t) \\ &\qquad \prod_{(i,j)} \psi(o_i^t, o_j^t)\psi(\mathcal{A}^t, \mathcal{A}^{t-1}) \prod_i \psi(o_i^t, o_i^{t-1}) \Bigg) \\ &\times \prod_t \Bigg(\prod_i \phi(x_i^t, h_i^t)\phi(x_i^t, x_i^{t-1}) \Bigg)\end{aligned}$$

The first pair of parentheses contains the CRF terms, with parameters denoted by Θ_{CRF}. (They are similar to the terms in ATCRF.) The second pair of parentheses contains all terms related to latent nodes in GP-LCRF with parameters including K, α, γ, β, denoted by Θ_{latent}. Note that Θ_{CRF} and Θ_{latent} are two *disjoint* sets.

Therefore, learning can be decomposed into two independent problems: 1) learning Θ_{CRF} by using the cutting-plane method in the structural learning for SVM [65], same as [75]; 2) learning Θ_{latent} by minimizing the negative log-likelihood, given by:

$$\begin{aligned}& -\ln P(\{x_i\}, \alpha, \gamma, \beta | \{h_i\}) \\ =& -\ln P(\{h_i\}|\{x_i\}, \alpha, \gamma, \beta)P(\{x_i\})P(\alpha, \gamma, \beta) \\ =& \frac{D}{2}\ln|K| + \frac{1}{2}\sum_{k=1}^{D} H_k^T K^{-1} H_k + \frac{1}{2}\sum_{i=1}^{N} ||x_i||^2 + \ln \alpha\beta\gamma\end{aligned}$$

where the priors on the unknowns are: $P(x) = \mathcal{N}(0, I)$ and $P(\alpha, \beta, \gamma) \propto \alpha^{-1}\beta^{-1}\gamma^{-1}$. We use numerical optimization method L-BFGS [98] to minimize it.

6.2.4 Inference for Anticipation

Given the observed segment t, we predict the next future segment $t + 1$ in the following way: We first sample possible object trajectories, represented in locations $\mathcal{L}^{t+1}$. Then we sample human configurations $\mathcal{H}^{t+1}$ and $\mathcal{X}^{t+1}$. We now use the sampled $\mathcal{L}^{t+1}$ and $\mathcal{H}^{t+1}$ as observations and infer the most likely sub-activity labels $\mathcal{A}^{t+1}$and object affordance labels $\mathcal{O}^{t+1}$ by maximizing the conditional likelihood in Eq. (6.7). All the samples together form a distribution over the future possibilities and we use the one with maximum a posterior (MAP) as our final anticipation.

We now present how to sample $\mathcal{H}^{t+1}$ and $\mathcal{X}^{t+1}$ in particular. (Sampling other terms is similar as in [75].) Given object locations, we generate a human motion of either moving or reaching an object. In both cases, the hand trajectory is given and the problem is formulated as: Given a target hand location ℓ^*, compute the most likely human configurations where both x and h are unknown. A good pose should reach to the target as well as being reasonable which can be measured by the likelihood from GPLVM, $L(x, h)$ in Eq. (6.4). Hence, we define the objective function as:

$$\arg\min_{x,h} -L(x, h) + \lambda\|\ell^* - \ell(h)\|^2 \tag{6.8}$$

where λ is the penalty of the new pose deviating from the target. In our implementation, we start with a simple IK solution h_0, and use the inverse mapping function $g(h) = x$ (given by GPLVM with back constraints [83]) to compute its corresponding x_0. In this way, the first term in Eq. (6.4) is always zero and can be neglected. So the new objective becomes a function of h only:

$$\arg\min_{h} \frac{D}{2} \ln \sigma^2(g(h)) + \frac{1}{2}\|g(h)\|^2 + \lambda\|\ell^* - \ell(h)\|^2 \tag{6.9}$$

We then use L-BFGS to optimize it.

Table 6.1: Anticipation Results, computed over 3 seconds in the future averaged by 4-fold cross validation. The first six columns are in percentage and a higher value is better. The last column is in centimeters and a lower value is better.

Algorithms	Anticipated sub-activities			Anticipated object affordances			Anticipated traj.
	micro-P/R@1	macro-F1@1	Pre@3	micro-P/R@1	macro-F1@1	Pre@3	MHD@1 (cm)
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	48.1±0.9
ATCRF-KGS [75]	47.7±1.6	37.9±2.6	69.2±2.1	66.1±1.9	36.7±2.3	71.3±1.7	31.0±1.0
ATCRF [76]	49.6±1.4	40.6±1.6	74.4±1.6	67.2±1.1	41.4±1.5	73.2±1.0	30.2±1.0
HighDim-LCRF	47.0±1.8	37.2±2.8	68.5±2.1	65.8±1.8	37.3±2.4	70.6±1.6	29.3±0.9
PPCA-LCRF	50.0±1.5	40.7±1.4	74.2±1.2	67.8±1.7	41.7±1.3	73.4±1.0	28.7±0.9
Our GP-LCRF	**52.1±1.2**	**43.2±1.5**	**76.1±1.5**	**68.1±1.0**	**44.2±1.2**	**74.9±1.1**	**26.7±0.9**

6.3 Experiments

Data. We test our model on the Cornell Activity Dataset-120 (CAD-120), same as used in [75, 76]. It contains 120 3D videos of four different subjects performing 10 high-level activities, where each high-level activity was performed three times with different objects. It contains a total of 61,585 total 3D video frames. The dataset is labeled with both sub-activity and object affordance labels. The sub-activity labels are: {*reaching, moving, pouring, eating, drinking, opening, placing, closing, scrubbing, null*} and the affordance labels are: {*reachable, movable, pourable, pour-to, containable, drinkable, openable, placeable, closable, scrubbable, scrubber, stationary*}.

Baselines. We compare against the following baselines:

1) *Chance*. Labels are chosen at random.

2) *ATCRF-KGS* [75]. ATCRF with fixed temporal structure.

3) *ATCRF* [76]. ATCRF with sampled temporal structures.

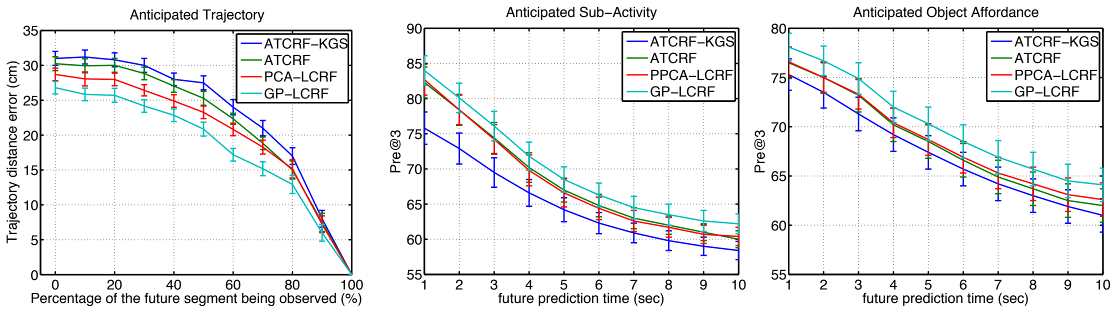


Figure 6.3: Plots showing (from left to right): a) how the trajectory distance error changes with the observed percentage in the segment to anticipate increases from 0% to 100%; b) The Pre@3 of the anticipated sub-activity labels as a function of the length of future prediction time in seconds; c) The Pre@3 of the anticipated object affordance labels as a function of the length of future prediction time in seconds.

4) *HighDim-LCRF*. In this method, we do not compress the human configuration into a low-dimensional representation but directly model human dynamics in the high-dimensional space. We replace $\phi(x_i^t, h_i^t)$ with a Gaussian based on the distance between h_i^t to its nearest neighbor h^* in the training data. For an anticipated frame, we use inverse kinematics to generate a new pose that is closest to the target trajectory (without considering its GPLVM likelihood). We also change $\phi(x_i^{t-1}, x_i^t)$ to $\phi(h_i^{t-1}, h_i^t) \sim \mathcal{N}(\|h_i^{t-1} - h_i^t\|^2; 0, 1)$.

5) *PPCA-LCRF*. We use probabilistic principal component analysis (PPCA) instead of GPLVM for dimensionality reduction of human configurations. PPCA only learns a linear mapping and do not impose any prior on the latent space and the mapping. We verify through experiments that it does not model low-dimensional human dynamics well and thus is outperformed by our GP-LCRF model.

Evaluation. We 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 and evaluated by the following metrics (same are used in [75, 76]):

1) *Labeling Metrics (on top#1 prediction).* For anticipated sub-activity and affordance labels, we compute the overall micro accuracy (P/R) and macro F1 score. Micro precision/recall is equal to the percentage of correctly classified labels. Macro precision and recall are averaged over all classes.

2) *Pre@3.* In practice a robot should plan 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 label matches one of the top three predictions, then it counts towards positive.

3) *Trajectory Metric (on top#1 prediction).* For anticipated human trajectories, we compute the modified Hausdorff distance (MHD) to the true trajectories. MHD finds the best local point correspondence of the two trajectories over a small temporal window to compute distance between those points. The distance is normalized by the length of the trajectory.

Table 6.1 shows the frame-level metrics for anticipating subactivity and object affordance labels for 3 seconds in the future on the CAD-120 dataset. We can see that our proposed GP-LCRF outperforms all the baseline algorithms and achieves a consistent increase across all metrics. Especially as our GP-LCRF aims to model human configurations better, we can see that the anticipated human trajectory error is reduced from 30.2 cm to 26.7 cm which is a 11.6% improvement and has a p-value of 0.0107 indicating the difference is statistically very significant. We now inspect the results in detail from the following aspects:

The importance of dimensionality reduction. Table 6.1 shows that when not using any dimensionality reduction, HighDim-LCRF performs even worse than

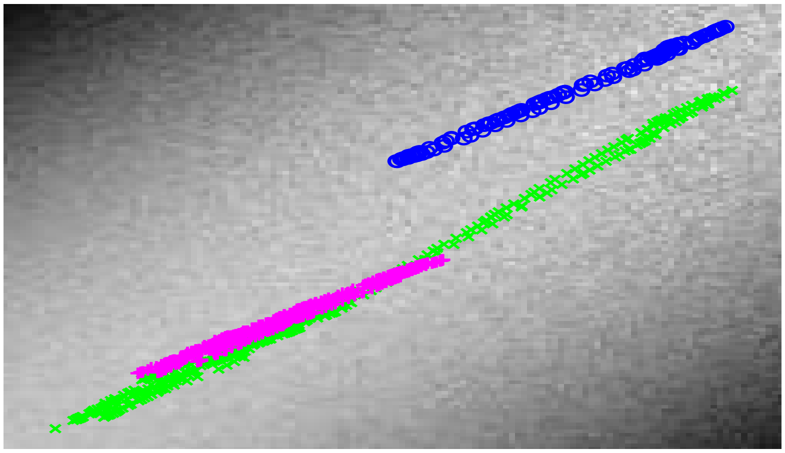


Figure 6.4: The learned mapping using PPCA. The colored points corresponding to the activities in Fig. 6.2.

ATCRF even though it tries to model the human temporal dynamics. This is because that in the high-dimensional space, $\phi(h^{t-1}, h^t)$ can be noisy and overfitted, thus modeling it actually hurts the performance.

On the other hand, with dimensionality reduction, PPCA-LCRF outperforms HighDim-LCRF, however it only achieves comparable results as ATCRF. This shows that the quality of the dimensionality reduction is quite important. Figure 6.4 illustrates a learned mapping of human configurations. Although both mapped to a 2D space, compared to GPLCRF in Fig. 6.2, PPCA learns a flat mapping and does not distinguish different motions well enough. For instance, the motions in the activity of 'taking medicine' (in magenta) and 'microwaving food' (in green) are very different, however they are mapped to an overlapped area using PPCA in Fig. 6.4. As a result, the effect of the dimensionality reduction in PPCA-LCRF is not as significant as our GP-LCRF.

Sensitivity of the results to the degree of dimensionality reduction. We investigate the performance of GP-LCRF with different dimensions of the latent

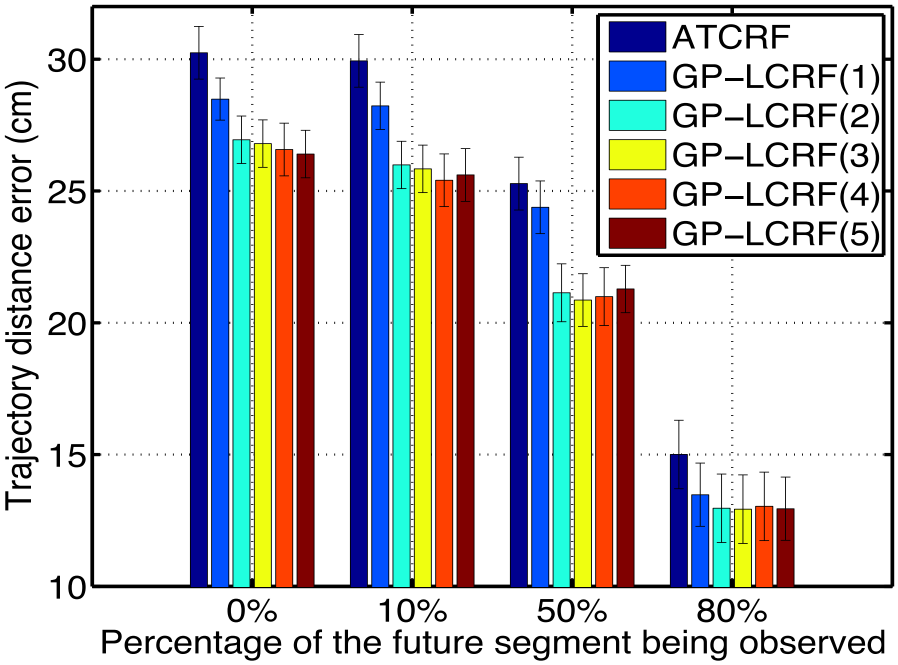


Figure 6.5: The trajectory distance error of GP-LCRF with different dimensions of latent space (from 1D to 5D, shown in the parentheses). We evaluate the performance under different conditions where the percentage of the future segment observed is 0%, 10%, 50% and 80%, i.e., the task is to anticipate is the rest of 100%, 90%, 50% and 20% of that segment respectively.

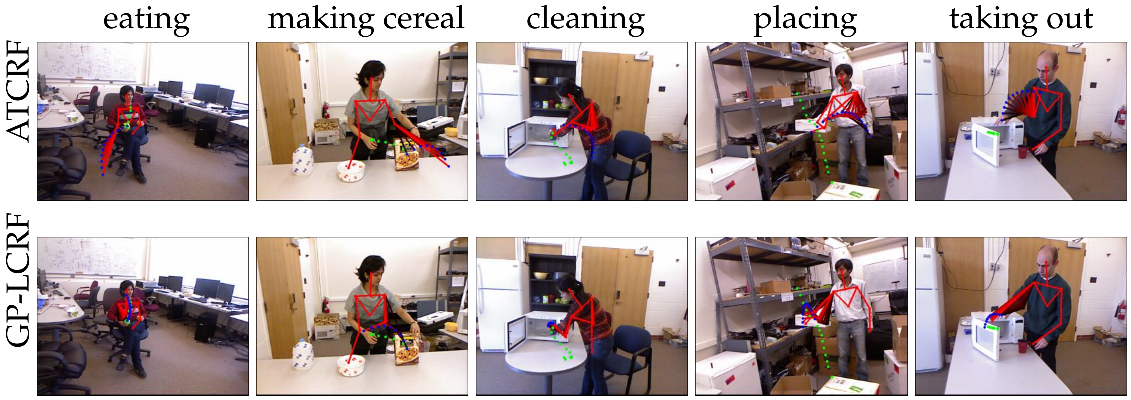


Figure 6.6: Top-ranked trajectories predicted by ATCRF (top) and our GP-LCRF (bottom) for different activities. In each image, the ground-truth trajectory is shown in green dots, predicted trajectory in blue, and the anticipated human skeletons in red in the order of from dark to bright.

space, from 1-D to 5-D in Fig. 6.5, in terms of the trajectory distance error. We can see that under various learning conditions (where the anticipated segment is observed in different percentages), GP-LCRF with latent dimensions of two to five all give similar performance. Dimensions of one has an obvious performance drop but is still better than ATCRF. However, with the observation's percentage increase to 80%, the gap diminishes as the anticipation problem becomes easier. Evaluations with the labeling metrics share similar trends. Hence, this shows that our GP-LCRF is very robust to the choices of the latent dimensions.

The impact of the observation time. The first plot in Fig. 6.3 shows how the trajectory distance error, averaged over all the moving sub-activities in the dataset, changes with the increase of the observed part (in percentage) in the segment to be anticipated. While all approaches achieve better predictions with increased observation time, our GP-LCRF consistently performs better than the others, especially in the range of 20% to 60%. Because this part, unlike the beginning where the evidence of human motions is too weak to be useful and unlike the near end where the evidence human-object interactions weighs more than humans alone, is where the momentum of human motions can be captured from the observation by our model (through the velocity and acceleration features) and be fully utilized for anticipation.

Results with change in the future anticipation time. The last two plots in Fig. 6.3 show the changes of Pre@3 with the anticipation time lengthened. The longer the anticipation time, the harder the task gets and thus the performances of all approaches decrease. However, the improvement of our GP-LCRF against ATCRF grows from 1.7% to 2.2% for sub-activity anticipation and from 1.6% to 2.1% for object affordance anticipation. This demonstrates the potential of modeling human kinematics well in long-term anticipations.

How does modeling human dynamics improve anticipated trajectories? In addition to the quantitative results in Fig. 6.3-(a), we also sample some qualitative results showing the top-ranked predicted trajectories in Fig. 6.6 using ATCRF (top) and using our GP-LCRF (bottom). In each image, we illustrate the predicted hand trajectories in blue dots, the ground-truth trajectories in green dots and human skeletons in red. We performed an ablative analysis and we now discuss some failures in the original ATCRF but are avoided by our GP-LCRF, arranged in three major categories:

1) Unrealistic skeletons leading to impossible trajectories: In the first two cases/columns, the trajectories sampled by ATCRF are both not reachable (without making any effort such as bending over or leaning forward). As ATCRF does not consider any human kinematics and it simply changes the hand location to match the trajectory, the forearms in these two cases are stretched out. The features computed from these false human skeletons are erroneous and thus wrong trajectories are picked out. Our GP-LCRF, however, generates kinematically-plausible skeletons (because of availability of high-dimensional configurations in the model) so that the out-of-reach trajectories will have high penalty in the likelihood $L(x, h)$ and out-ranked by those reachable ones.

2) Unnatural poses leading to unlikely trajectories: In other cases, such as the third column in Fig. 6.6 where the subject picked up a rag on the table along the green dots to clean the microwave, both trajectories are physically possible but the top one requires raising the right hand to cross the left hand making a very unnatural pose. Because GP-LCRF learns the distribution of human poses from the training data, it assigns a low probability to uncommon poses such as the top one and prefers the bottom poses and the trajectory instead.

3) Not modeling motions leading to discontinuous trajectories: How human body moved in the past gives such a strong cue that sometimes we can have decent anticipated trajectories purely based on the continuity and smoothness of human motions. For instance, the two subjects are lifting the box (4th column) and reaching towards the microwave door (last column). While our GP-LCRF chooses trajectories matching the moving directions best, ATCRF which does not model human temporal relations (i.e., no edges between $\mathcal{H}^{t-1}$ and $\mathcal{H}^{t}$) produces trajectories with sudden changes in the direction.

Runtime. On a 16-core 2.7GHz CPU, our code takes 11.2 seconds to anticipate 10 seconds in the future, thus achieving *near real-time* (1.12*X)* performance.

CHAPTER 7

LEARNING TO PLACE

7.1 Overview

In order to autonomously perform common daily tasks such as setting up a dinner table, arranging a living room or organizing a closet, a personal robot should be able to figure out where and how to place objects. However, this is particularly challenging because there can potentially be a wide range of objects and placing environments. Some of them may not have been seen by the robot before. For example, to tidy a disorganized house, a robot needs to decide *where* the best place for an object is (e.g., books should be placed on a shelf or a table and plates are better inserted in a dish-rack), and *how* to place the objects in an area (e.g. clothes can be hung in a closet and wine glasses can be held upside down on a stemware holder). In addition, limited space, such as in a cabinet, raises another problem of how to *stack* various objects together for efficient storage. Determining such a placing strategy, albeit rather natural or even trivial to (most) people, is quite a challenge for a robot.

In this chapter, we consider multiple objects and placing areas represented by possibly incomplete and noisy point-clouds. Our goal is to find proper placing strategies to place the objects into the areas. A placing *strategy* of an object is described by a preferred placing area for the object and a 3D location and orientation to place it in that area. As an example, Fig. 7.1 shows one possible strategy to place six different types of objects onto a bookshelf. In practice, the following criteria should be considered.

Stability: Objects should be placed stably in order to avoid falling. For example, some objects can be stably placed upright on flat surfaces,[1] plates can be placed stably in different orientations in a dish-rack, and a pen can be placed horizontally on a table but vertically in a pen-holder.

Semantic preference: A placement should follow common human preferences in placing. For instance, shoes should be placed on the ground but not on a dinner plate, even though both areas have geometrically similar flat surfaces. Therefore, a robot should be able to distinguish the areas semantically, and make a decision based on common human practice.

Stacking: A placement should consider possible stacking of objects such as piling up dinner plates. However, this raises more challenges for a robot because it has to decide which objects can be stacked together semantically. For example, it is a bad idea to stack a dinner plate on top of a cell phone rather than another dinner plate. In addition, the robot has to decide the order of stacking in a dynamically changing environment, since previously placed objects can change the structure of placing areas for objects placed later.

In addition to the difficulties introduced by these criteria, perceiving the 3D geometry of objects and their placing environments is nontrivial as well. In this thesis, we use a depth camera mounted on a robot to perceive the 3D geometries as point-clouds. In practice, the perceived point-clouds can be noisy and incomplete (see Fig. 7.1), requiring the robot to be able to infer placements with only partial and noisy geometric information.

In this chapter, we address these challenges using a learning-based approach. We encode human preferences about placements as well as the geo-

[1] Even knowing the “upright” orientation for an arbitrary object is a non-trivial task [29].

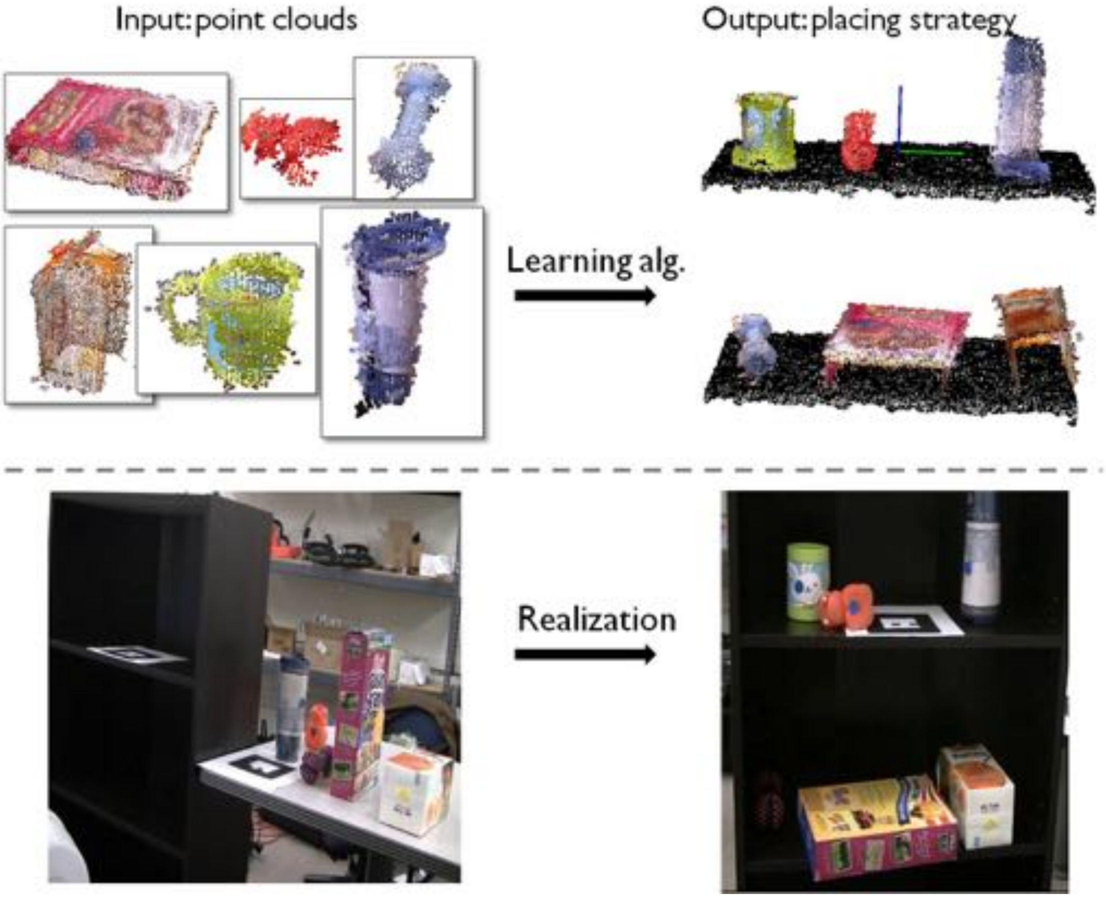


Figure 7.1: An example task of placing items on a bookshelf. Given the point-clouds of the bookshelf and six objects to be placed (shown in top-left part of the figure), our learning algorithm finds out the best placing strategy, specified by the location and orientation of every object (shown in top-right). Following this inferred strategy, the robot places each object accordingly. The bottom part of the figure shows the scene before and after placing. Note that in some cases, the placing environment can be quite complex (e.g, see Fig. 7.9).

metric relationship between objects and their placing environments by designing appropriate features. We then propose a graphical model that has two substructures to capture the stability and the semantic preferences respectively. The model also incorporates stacking and constraints that keeps the placing strategy physically feasible. We use max-margin learning for estimating the parameters in our graphical model. The learned model is then used to score the potential placing strategies. Given a placing task, although inferring the best strategy (with the highest score) is provably NP-complete, we express the inference as an integer linear programming (ILP) problem which is then solved efficiently using an linear programming (LP) relaxation.

To extensively test our approach, we constructed a large placing database composed of 98 household objects from 16 different categories and 40 placing areas from various scenes. Experiments ranged from placing a single object in challenging situations to complete-scene placing where we placed up to 50 objects in real offices and apartments. In the end-to-end test of placing multiple objects in different scenes, our algorithm significantly improves the performance—on metrics of stability, semantic correctness and overall impressions on human subjects—as compared to the best baseline. Quantitatively, we achieve an average accuracy of 83% for stability and 82% for choosing a correct placing area for single-object placements. Finally, we tested our algorithm on two different robots on several placing tasks. We then applied our algorithm to several practical placing scenarios, such as loading multiple items in dish-racks, loading a fridge, placing objects on a bookshelf and cleaning a disorganized room. We have also made the code and data available online at: `http://pr.cs.cornell.edu/placingobjects`

The rest of this chapter is organized as follows: We start with a review of the related work in Section 7.2. We then formulate the placing problem in a machine learning framework in Section 7.3. We present our learning algorithm for single-object placements in Section 7.4 and the corresponding experiments in Section 7.5. Finally, we give the algorithm and the experiments for multiple-object placements in Section 7.6 and Section 7.7 respectively.

7.2 Related Work

While there has been significant previous work on grasping objects [e.g., 9, 12, 95, 115, 116, 122, 6, 104, 45, 11, 84, 56, 20, 106, 61], there is little work on object placement, and it is restricted to placing objects on flat horizontal surfaces. For example, [124] recently developed a learning algorithm to detect clutter-free 'flat' areas where an object can be placed. Unfortunately, there are many non-flat placing areas where this method would not apply. Even if placing only on flat surfaces, one needs to decide the upright or the current orientation of a given object, which is a challenging task. For example, [29] proposed several geometric features to learn the upright orientation from an object's 3D model and [120] predicted the orientation of an object given its 2D image. Recently, [33] used a database of models to estimate the pose of objects with partial point-clouds. Our work is different and complementary to these studies: we generalize placing environment from flat surfaces to more complex ones, and desired configurations are extended from upright to all other possible orientations that can make the best use of the placing area. Furthermore, we consider: 1) placing objects in scenes comprising a wide-variety of placing areas, such as dish-racks, stemware holders, cabinets and hanging rods in closets; 2) placing multiple objects; 3) placing objects in semantically preferred locations (i.e., how to choose a proper placing context for an object).

For robotic placing, one component is to plan and control the arm to place the objects without knocking them down. [23] considered placing objects on a flat shelf, but their focus was to use passive compliance and force control to gently place the object on the shelf. Planning and rule-based approaches have been used to move objects around. For example, [93] proposed a task-level (in contrast with motion-level) planning system and tested it on picking and placing objects on a table. [128] used rule-based planning in order to push objects on a table surface. However, most of these approaches assume known full 3D models of the objects, consider only flat surfaces, and do not model semantic preferences in placements.

Placing multiple objects also requires planning and high-level reasoning about the order of execution. [6] coupled planning and grasping in cluttered scenes. They utilized an 'environment clearance score' in determining which object to grasp first. [136] integrated control, planning, grasping and reasoning in the 'blocks-world' application in which table-top objects were rearranged into several stacks by a robot. How to arrange objects efficiently is also related to the classic bin packing problem [13] which can be approached as integer linear programming [28], constraint satisfaction problem [100] or tabu search [92]. These studies focus on generic planning problems and are complementary to ours.

Contextual cues [e.g., 135] have proven helpful in many vision applications. For example, using estimated 3D geometric properties from images can be useful for object detection [119, 117, 42, 89, 40, 19]. In [144] and [2], contextual information was employed to estimate semantic labels in 3D point-clouds of indoor environments. [17] used object context for object retrieval. [26] and [27] designed a context-based search engine using geometric cues and spatial relationships to find the proper object for a given scene. Unlike our work, their goal was only to retrieve the object but not to place it afterwards. While these works address different problems, our work that captures the semantic preferences in placing is motivated by them.

Object categorization is also related to our work as objects from same category often share similar placing patterns. Categorization in 2D images is a well-studied computer vision problem. Early work [e.g., 143, 86, 25, 7] tried to solve shape and orientation variability, limited by a single viewpoint. Motivated by this limitation, multi-view images were considered for categorizing 3D generic object by connecting 2D features [134, 113]. When 3D models were available, some work categorized objects based on 3D features instead, e.g., using synthetic 3D data to extract pose and class discriminant features [90], and using features such as spin images [66] and point feature histograms [111] for 3D recognition. Instead of images or 3D models, [81] proposed a learning algorithm for categorization using both RGB and multi-view point-cloud. These works are complementary to ours in that we do not explicitly categorize the object before placing, but knowing the object category could potentially help in placing.

Most learning algorithms require good features as input, and often these features are hand-designed for the particular tasks [29, 122]. There have also been some previous works on high-dimensional 3D features [44, 111, 66, 90, 122] but they do not directly apply to our problem. There is also a large body of work on automatic feature selection [see 22, 91, 51], which could potentially improve the performance of our algorithm.

In the area of robotic manipulation, a wide range of problems have been studied so far, such as folding towels and clothes [94], opening doors [47, 71], inferring 3D articulated objects [126, 67] and so on. However, they address different manipulation tasks and do not apply to the placing problem we consider in this thesis. Our work is the first one to consider object placements in complex scenes.

7.3 Problem Formulation

In this section, we formulate the problem of how to predict good placements in a placing task.

Specifically, our goal is to place a set of objects in a scene that can contain several potential placing areas. Both the objects and the placing areas are perceived as point-clouds that can be noisy and incomplete. A placement of an object is specified by 1) a 3D location describing at which 3D position in the scene the object is placed, and 2) a 3D rotation describing the orientation of the object when it is placed. In the following, we first consider the problem of single-object placements. We will then extend it to multiple-object placements by adding semantic features and modifying the algorithm to handle multiple objects. The learning algorithm for single-object placements is only a special case of the algorithm for multiple-object placements.

7.3.1 Single-Object Placements

Here, our goal is to place an object stably in a designated placing area in the scene. We consider stability and preferences in orientation, but would not consider the semantic preferences about which placing area to choose. Specifically, the goal is to infer the 3D location ℓ (in a placing area E) and 3D orientation c in which to place the object O.

We illustrate the problem formulation in Fig. 7.2. We are given an object O and a placing area E in the form of point-clouds. Given the input, we first sample a set of possible placements, compute relevant features for each, and then use the learned model to compute a score for each candidate. The highest-score placement is then selected to be executed by the robot. We describe our features and the learning algorithm in Section 7.4.

7.3.2 Multiple-Object Placements

In addition to finding a proper location and orientation to place the object, we often need to decide which placing area in the scene is semantically suitable for placing the object in. When multiple objects are involved, we also need to consider stacking while placing them.

As an example, consider organizing a kitchen (see Fig. 7.7). In such a case, our goal is to place a set of given objects into several placing areas. One can place objects in two ways: directly on an existing area in the environment (e.g., saucepans on the bottom drawer), or stacking one on top of another (e.g., bowls piled up in the cabinet).

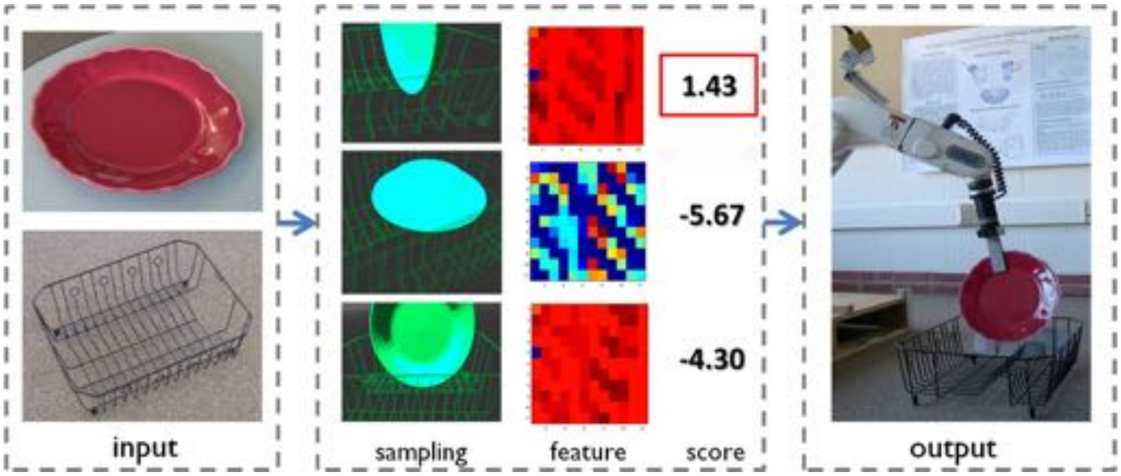
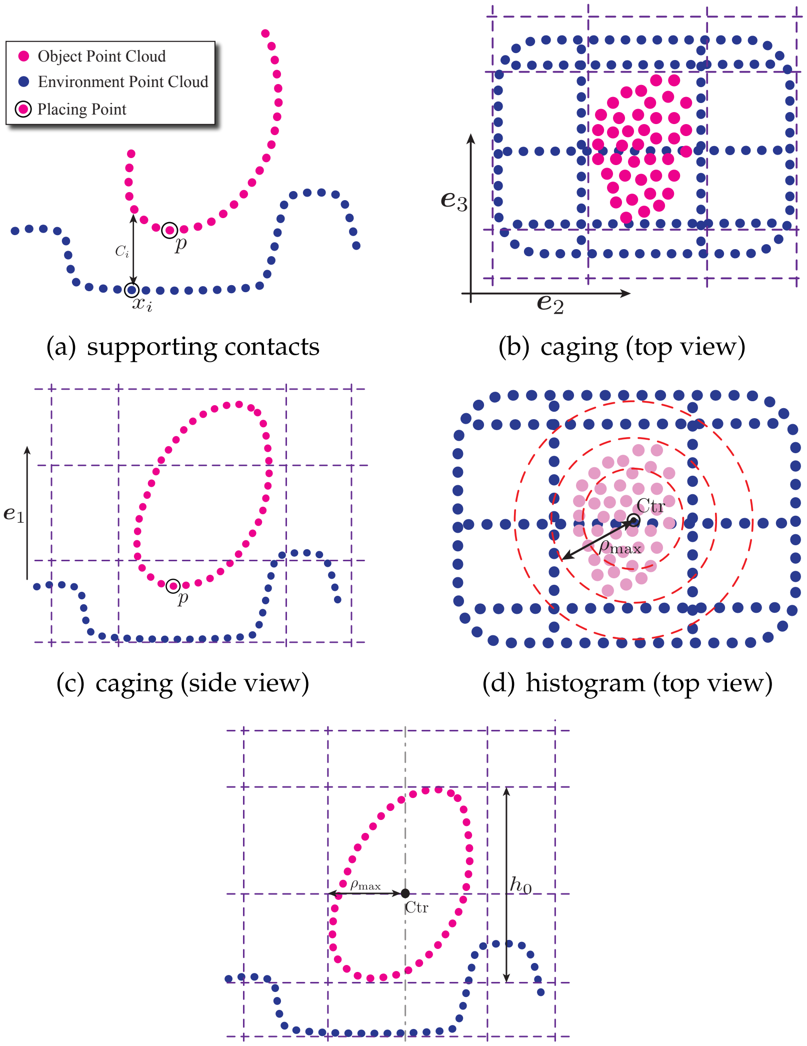


Figure 7.2: Our formulation of single-object placements as a learning problem. Steps from left to right: 1) we are given an object to be placed and a placing area, in the form of point-clouds; 2) we first sample possible placements, extract features for each sample, and then use our learned model to compute a score for each sample. The higher the score is, the more likely it is to be a good placement; 3) the robot plans a path to the predicted placement and follows it to realize the placing.

Formally, the input of this problem is n objects $O = \{O_1, \ldots, O_n\}$ and m placing areas (also called environments) $\mathcal{E} = \{E_1, \ldots, E_m\}$, all of which are represented by point-clouds. The output will be a placing strategy depicting the final layout of the scene after placing. It is specified by the pose of every object, which includes the configuration (i.e., 3D orientation) c_i, the placing area (or another object when stacking) selected, and the relative 3D location ℓ_i w.r.t. this area. We propose a graphical model to represent the placing strategy, where we associate every possible strategy with a potential function. Finding the best placing strategy is then equivalent to maximizing the potential function. Details are in Section 7.6.



(a) supporting contacts

(b) caging (top view)

(c) caging (side view)

(d) histogram (top view)

(e) histogram (side view)

Figure 7.3: Illustration of features in our learning algorithm for single-object placements. These features are designed to capture the stability and preferred orientations in a good placement.

7.4 Algorithm for Single-Object Placements

In order to identify good placements, we first need to design features that indicate good placements across various objects and placing areas. We then use a max-margin learning algorithm to learn a function that maps a placement, represented by its features, to a placing score. In testing, we first randomly sample some placements, then use the function to find the highest-score candidate as our best placement.

7.4.1 Features

The features used in our learning algorithm are designed to capture the following two properties:

- **Supports and Stability.** The object should stay still after placing. Ideally, it should also be able to withstand small perturbations.
- **Preferred Orientation.** A good placement should have semantically preferred orientation as well. For example, plates should be inserted into a dish-rack vertically and glasses should be held upside down on a stemware holder.

An important property of the stability features is invariance under translation and rotation (about the gravity, i.e., Z-axis). This is because as long as the relative configuration of the object and the placing area remains same, the features should not change. Most of our features will follow this property.

We group the features into three categories. In the following description, we use O' to denote the point-cloud of the object O after being placed, and use B to denote the point-cloud of a placing area B. Let $\boldsymbol{p}_o$ be the 3D coordinate of a point $o \in \mathsf{O}'$ from the object, and $\boldsymbol{x}_t$ be the coordinate of a point $t \in \mathsf{B}$ from the placing area.

Supporting Contacts: Intuitively, an object is placed stably when it is supported by a wide spread of contacts. Hence, we compute features that reflect the distribution of the supporting contacts. In particular, we choose the top 5% points in the placing area closest to the object (measured by the vertical distance, c_i shown in Fig. 7.3(a)) at the placing point. Suppose the k points are $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$. We quantify the set of these points by 8 features:

1. Falling distance $\min_{i=1\ldots k} c_i$.
2. Variance in XY-plane and Z-axis respectively, $\frac{1}{k}\sum_{i=1}^{k}(\boldsymbol{x}'_i - \bar{\boldsymbol{x}}')^2$, where $\boldsymbol{x}'_i$ is the projection of $\boldsymbol{x}_i$ and $\bar{\boldsymbol{x}}' = \frac{1}{k}\sum_{i=1}^{k}\boldsymbol{x}'_i$.
3. Eigenvalues and ratios. We compute the three Eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) of the covariance matrix of these k points. Then we use them along with the ratios λ_2/λ_1 and λ_3/λ_2 as the features.

Another common physics-based criterion is the center of mass (COM) of the placed object should be inside of (or close to) the region enclosed by contacts. So we calculate the distance from the centroid of O' to the nearest boundary of the 2D convex hull formed by contacts projected to XY-plane, $\mathcal{H}_{con}$. We also compute the projected convex hull of the whole object, $\mathcal{H}_{obj}$. The area ratio of these two polygons $S_{\mathcal{H}_{con}}/S_{\mathcal{H}_{obj}}$ is included as another feature.

Two more features representing the percentage of the object points below or above the placing area are used to capture the relative location.

Caging: There are some placements where the object would not be strictly immovable but is well confined within the placing area. A pen being placed upright in a pen-holder is one example. While this kind of placement has only a few supports from the pen-holder and may move under perturbations, it is still considered a good one. We call this effect 'gravity caging.'[2]

We capture this by partitioning the point-cloud of the environment and computing a battery of features for each zone. In detail, we divide the space around the object into 3 × 3 × 3 zones. The whole divided space is the axis-aligned bounding box of the object scaled by 1.6, and the dimensions of the center zone are 1.05 times those of the bounding box (Fig. 7.3(b) and 7.3(c)). The point-cloud of the placing area is partitioned into these zones labelled by $\Psi_{ijk},\ i, j, k \in \{1, 2, 3\}$, where i indexes the vertical direction $\boldsymbol{e}_1$, and j and k index the other two orthogonal directions, $\boldsymbol{e}_2$ and $\boldsymbol{e}_3$, on horizontal plane.

From the top view, there are 9 regions (Fig. 7.3(b)), each of which covers three zones in the vertical direction. The maximum height of points in each region is computed, leading to 9 features. We also compute the horizontal distance to the object in three vertical levels from four directions ($\pm\boldsymbol{e}_2, \pm\boldsymbol{e}_3$) (Fig. 7.3(c)). In particular, for each $i = 1, 2, 3$, we compute

$$\begin{aligned}
d_{i1} &= \min_{\substack{\boldsymbol{x}_t \in \Psi_{i11} \cup \Psi_{i12} \cup \Psi_{i13} \\ \boldsymbol{p}_o \in \mathcal{O}'}} \boldsymbol{e}_2^T(\boldsymbol{p}_o - \boldsymbol{x}_t) \\
d_{i2} &= \min_{\substack{\boldsymbol{x}_t \in \Psi_{i31} \cup \Psi_{i32} \cup \Psi_{i33} \\ \boldsymbol{p}_o \in \mathcal{O}'}} -\boldsymbol{e}_2^T(\boldsymbol{p}_o - \boldsymbol{x}_t) \\
d_{i3} &= \min_{\substack{\boldsymbol{x}_t \in \Psi_{i11} \cup \Psi_{i21} \cup \Psi_{i31} \\ \boldsymbol{p}_o \in \mathcal{O}'}} \boldsymbol{e}_3^T(\boldsymbol{p}_o - \boldsymbol{x}_t) \\
d_{i4} &= \min_{\substack{\boldsymbol{x}_t \in \Psi_{i13} \cup \Psi_{i23} \cup \Psi_{i33} \\ \boldsymbol{p}_o \in \mathcal{O}'}} -\boldsymbol{e}_3^T(\boldsymbol{p}_o - \boldsymbol{x}_t)
\end{aligned} \tag{7.1}$$

and produce 12 additional features.

The degree of gravity-caging also depends on the relative height of the object and the caging placing area. Therefore, we compute the histogram of the height

[2] This idea is motivated by previous works on force closure [97, 101] and caging grasps [18].

of the points surrounding the object. In detail, we first define a cylinder centered at the lowest contact point with a radius that can just cover O'. Then points of B in this cylinder are divided into $n_r \times n_\theta$ parts based on 2D polar coordinates. Here, n_r is the number of radial divisions and n_θ is the number of divisions in azimuth. The vector of the maximum height in each cell (normalized by the height of the object), $H = (h_{1,1}, \ldots, h_{1,n_\theta}, \ldots, h_{n_r,n_\theta})$, is used as additional caging features. To make H rotation-invariant, the cylinder is always rotated to align polar axis with the highest point, so that the maximum value in H is always one of $h_{i,1}, i = 1...n_r$. We set $n_r = 4$ and $n_\theta = 4$ for single-object placement experiments.

Histogram Features: Generally, a placement depends on the geometric shapes of both the object and the placing area. We compute a representation of the geometry as follows. We partition the point-cloud of O' and of B radially and in Z-axis, centered at the centroid of O'. Suppose the height of the object is h_O and its radius is ρ_{max}. The 2D histogram with $n_z \times n_\rho$ number of bins covers the cylinder with the radius of $\rho_{max} \cdot n_\rho/(n_\rho - 2)$ and the height of $h_O \cdot n_z/(n_z - 2)$, illustrated in Fig. 7.3(d) and 7.3(e). In this way, the histogram (number of points in each cell) can capture the global shape of the object as well as the local shape of the environment around the placing point. We also compute the ratio of the two histograms as another set of features, i.e., the number of points from O' over the number of points from B in each cell. The maximum ratio is fixed to 10 in practice. For single-object placement experiments, we set $n_z = 4$ and $n_\rho = 8$ and hence have 96 histogram features.

In total, we generate 145 features for the single-object placement experiments: 12 features for supporting contacts, 37 features for caging, and 96 for the histogram features.

7.4.2 Max-Margin Learning

Under the setting of a supervised learning problem, we are given a dataset of labeled good and bad placements (see Section 7.5.1), represented by their features. Our goal is to learn a function of features that can determine whether a placement is good or not. As support vector machines (SVMs) [14] have strong theoretical guarantees in the performance and have been applied to many classification problems, we build our learning algorithm based on SVM.

Let $\phi_i \in \mathbb{R}^p$ be the features of i^{th} instance in the dataset, and let $y_i \in \{-1, 1\}$ represent the label, where 1 is a good placement and -1 is not. For n examples in the dataset, the soft-margin SVM learning problem [64] is formulated as:

$$\begin{aligned} &\min \frac{1}{2} \|\theta\|_2^2 + C \sum_{i=1}^{n} \xi_i \\ &\text{s.t. } y_i(\theta^T \phi_i - b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \ \forall 1 \leq i \leq n \end{aligned} \tag{7.2}$$

where $\theta \in \mathbb{R}^p$ are the parameters of the model, and ξ are the slack variables. This method finds a separating hyperplane that maximizes the margin between the positive and the negative examples. Note that our formulation here is a special case of the graphical model for the multiple-object placements described in Section 7.6. We also use max-margin learning to estimate the parameters in both cases.

7.4.3 Shared-sparsity Max-margin Learning

If we look at the objects and their placements in the environment, we notice that there is an intrinsic difference between different placing settings. For example, it seems unrealistic to assume placing dishes into a rack and hanging martini

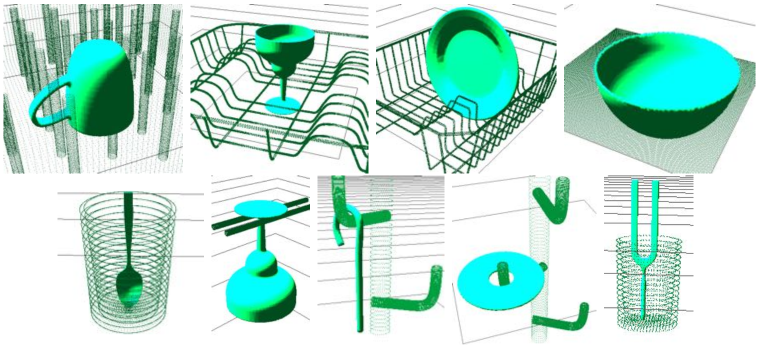


Figure 7.4: Some snapshots from our rigid-body simulator showing different objects placed in different placing areas. Placing areas from left: rack1, rack2, rack3, flat surface, pen holder, stemware holder, hook, hook and pen holder. Objects from left: mug, martini glass, plate, bowl, spoon, martini glass, candy cane, disc and tuning fork. Rigid-body simulation is only used in labeling the training data (Section 7.5.1) and in first half of the robotic experiments when 3D object models are used (Section 7.5.6).

glasses upside down on a stemware holder share exactly the same hypothesis, although they might agree on a subset of attributes. While some attributes may be shared across different objects and placing areas, there are some attributes that are specific to the particular setting. In such a scenario, it is not sufficient to have either one single model or several completely independent models for each placing setting that tend to suffer from over-fitting. Therefore, we propose to use a shared sparsity structure in our learning.

Say, we have M objects and N placing areas, thus making a total of $r = MN$ placing 'tasks' of particular object-area pairs. Each task can have its own model but intuitively these should share some parameters underneath. To quantify this constraint, we use ideas from recent works [49, 87] that attempt to capture the structure in the parameters of the models. [49] used a shared sparsity structure for multiple linear regressions. We apply their model to the classic soft-margin SVM.

In detail, for r tasks, let $\Phi_i \in \Re^{p \times n_i}$ and Y_i denote training data and its corresponding label, where p is the size of the feature vector and n_i is the number of data points in task i. We associate every task with a weight vector θ_i. We decompose θ_i in two parts $\theta_i = S_i + B_i$: the self-owned features S_i and the shared features B_i. All self-owned features, S_i, should have only a few non-zero values so that they can reflect individual differences to some extent but would not become dominant in the final model. Shared features, B_i, need not have identical values, but should share similar sparsity structure across tasks. In other words, for each feature, they should all either be active or non-active. Let $\|S\|_{1,1} = \sum_{i,j} |S_i^j|$ and $\|B\|_{1,\infty} = \sum_{j=1}^{p} \max_i |B_i^j|$. Our new goal function is now:

$$\begin{aligned}
\min_{\theta_i, b_i, i=1,\ldots,r} \quad & \textstyle\sum_{i=1}^{r} \left(\frac{1}{2} \|\theta_i\|_2^2 + C \sum_{j=1}^{n_i} \xi_{i,j} \right) + \\
& \qquad \lambda_S \|S\|_{1,1} + \lambda_B \|B\|_{1,\infty} \\
\text{subject to} \quad & Y_i^j (\theta_i^T \Phi_i^j + b_i) \geq 1 - \xi_{i,j}, \quad \xi_{i,j} \geq 0 \\
& \qquad \forall 1 \leq i \leq r, 1 \leq j \leq n_i \\
& \qquad \theta_i = S_i + B_i, \quad \forall 1 \leq i \leq r
\end{aligned} \tag{7.3}$$

When testing in a new scenario, different models vote to determine the best placement.

While this modification results in superior performance with new objects in new placing areas, it requires one model per object-area pair and therefore it does not scale to a large number of objects and placing areas.

7.5 Experiments on Placing Single Objects

We perform experiments on placing a single object in a designated placing area. The dataset includes 8 different objects and 7 placing areas. In these exper-

Table 7.1: Average performance of our algorithm using different features on the SESO scenario.

	chance	contact	caging	histogram	all
R_0	29.4	1.4	1.9	1.3	**1.0**
P@5	0.10	0.87	0.77	0.86	**0.95**
AUC	0.54	0.89	0.83	0.86	**0.95**

iments, our main purpose is to analyze the performance of our learning approach in finding stable placements with preferred orientations. Section 7.7 describes our full experiments with placing multiple objects in complete 3D scenes.

7.5.1 Data

Our dataset contains 7 placing areas (3 racks, a flat surface, pen holder, stemware holder and hook) and 8 objects (mug, martini glass, plate, bowl, spoon, candy cane, disc and tuning fork). We generated one training and one test dataset for each object-environment pair. Each training/test dataset contains 1800 random placements with different locations and orientations. After eliminating placements that have collisions, we have 37655 placements in total.

These placements were labeled by rigid-body simulation (Fig. 7.4) and then used for our supervised learning algorithm. Simulation enabled us to generate massive amounts of labeled data. However, the simulation itself had no knowledge of placing preferences. When creating the ground-truth training data, we manually labeled all the stable (as verified by the simulation) but non-preferred placements as negative examples.

Table 7.2: **Learning experiment statistics:** The performance of different learning algorithms in different scenarios is shown. The top three rows are the results for baselines, where no training data is used. The fourth row is trained and tested for the SESO case. The last three rows are trained using joint, independent and shared sparsity SVMs respectively for the NENO case.

Listed object-wise, averaged over the placing areas.

		plate			mug			martini			bowl			candy cane			disc			spoon			tuning fork			**Average**		
		flat, 3 racks			flat, 3 racks			flat, 3 racks, stemware holder			flat, 3 racks			flat, hook, pen holder			flat, hook, pen holder			flat, pen holder			flat, pen holder					
		R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC
baseline	chance	4.0	0.20	0.49	5.3	0.10	0.49	6.8	0.12	0.49	6.5	0.15	0.50	102.7	0.00	0.46	32.7	0.00	0.46	101.0	0.20	0.52	44.0	0.00	0.53	29.4	0.10	0.49
	flat-up	4.3	0.45	0.38	11.8	0.50	0.78	16.0	0.32	0.69	6.0	0.40	0.79	44.0	0.33	0.51	20.0	0.40	0.81	35.0	0.50	0.30	35.5	0.40	0.66	18.6	0.41	0.63
	lowest	27.5	0.25	0.73	3.8	0.35	0.80	39.0	0.32	0.83	7.0	0.30	0.76	51.7	0.33	0.83	122.7	0.00	0.86	2.5	0.50	0.83	5.0	0.50	0.76	32.8	0.30	0.80
	SESO	1.3	0.90	0.90	1.0	0.85	0.92	1.0	1.00	0.95	1.0	1.00	0.92	1.0	0.93	1.00	1.0	0.93	0.97	1.0	1.00	1.00	1.0	1.00	1.00	1.0	0.95	0.95
NENO	joint	8.3	0.50	0.78	2.5	0.65	0.88	5.2	0.48	0.81	2.8	0.55	0.87	16.7	0.33	0.76	20.0	0.33	0.81	23.0	0.20	0.66	2.0	0.50	0.85	8.9	0.47	0.81
	indep.	2.0	0.70	0.86	1.3	0.80	0.89	1.2	0.86	0.91	3.0	0.55	0.82	9.3	0.60	0.87	11.7	0.53	0.88	23.5	0.40	0.82	2.5	0.40	0.71	5.4	0.64	0.86
	shared	1.8	0.70	0.84	1.8	0.80	0.85	1.6	0.76	0.90	2.0	0.75	0.91	2.7	0.67	0.88	1.3	0.73	0.97	7.0	0.40	0.92	1.0	0.40	0.84	2.1	0.69	0.89

Listed placing area-wise, averaged over the objects.

		rack1			rack2			rack3			flat			pen holder			hook			stemware holder			**Average**		
		plate, mug, martini, bowl			plate, mug, martini, bowl			plate, mug, martini, bowl			all objects			candy cane, disc, spoon, tuningfork			candy cane, disc			martini					
		R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC
baseline	chance	3.8	0.15	0.53	5.0	0.25	0.49	4.8	0.15	0.48	6.6	0.08	0.48	128.0	0.00	0.50	78.0	0.00	0.46	18.0	0.00	0.42	29.4	0.10	0.49
	flat-up	2.8	0.50	0.67	18.3	0.05	0.47	4.8	0.20	0.60	1.0	0.98	0.91	61.3	0.05	0.45	42.0	0.00	0.45	65.0	0.00	0.58	18.6	0.41	0.63
	lowest	1.3	0.75	0.87	22.0	0.10	0.70	22.0	0.15	0.80	4.3	0.23	0.90	60.3	0.60	0.85	136.5	0.00	0.71	157.0	0.00	0.56	32.8	0.30	0.80
	SESO	1.0	1.00	0.91	1.3	0.75	0.83	1.0	1.00	0.93	1.0	0.95	1.00	1.0	1.00	0.98	1.0	1.00	1.00	1.0	1.00	1.00	1.0	0.95	0.95
NENO	joint	1.8	0.60	0.92	2.5	0.70	0.84	8.8	0.35	0.70	2.4	0.63	0.86	20.5	0.25	0.76	34.5	0.00	0.69	18.0	0.00	0.75	8.9	0.47	0.81
	indep.	1.3	0.70	0.85	2.0	0.55	0.88	2.5	0.70	0.86	1.9	0.75	0.89	12.3	0.60	0.79	29.0	0.00	0.79	1.0	1.00	0.94	5.4	0.64	0.86
	shared	1.8	0.75	0.88	2.0	0.70	0.86	2.3	0.70	0.84	1.3	0.75	0.92	4.0	0.60	0.88	3.5	0.40	0.92	1.0	0.80	0.95	2.1	0.69	0.89

7.5.2 Learning Scenarios

In real-world placing, the robot may or may not encounter new placing areas and new objects. Therefore, we trained our algorithm for two different scenarios: 1) Same Environment Same Object (**SESO**), where training data only contains the object and the placing environment to be tested. 2) New Environment New Object (**NENO**). In this case, the training data includes all other objects and environments except the one for test. We also considered two additional learning scenarios, SENO and NESO, in [63]. More results can be found there.

7.5.3 Baseline Methods

We compare our algorithm with the following three heuristic methods:

- *Chance.* The location and orientation is randomly sampled within the bounding box of the area and guaranteed to be 'collision-free.'
- *Flat-surface-upright rule.* Several methods exist for detecting 'flat' surfaces [124], and we consider a placing method based on finding flat surfaces. In this method, objects would be placed with *pre-defined* upright orientation on the surface. When no flat surface can be found, a random placement would be picked. Note that this heuristic actually uses *more* information than our method.
- *Finding lowest placing point.* For many placing areas, such as dish-racks or containers, a lower placing point often gives more stability. Therefore, this heuristic rule chooses the placing point with the lowest height.

7.5.4 Evaluation Metrics

We evaluate our algorithm's performance on the following metrics:

- R_0: Rank of the first valid placement. ($R_0 = 1$ ideally)
- P@5: In top 5 candidates, the fraction of valid placements.

- AUC: Area under ROC Curve [38], a classification metric computed from all the candidates.
- $P_{\text{stability}}$: Success-rate (in %) of placing the object stably with the robotic arm.
- $P_{\text{preference}}$: Success-rate (in %) of placing the object stably in preferred configuration with the robotic arm.

7.5.5 Learning Experiments

We first verified that having different types of features is helpful in performance, as shown in Table 7.1. While all three types of features outperform chance, combining them together gives the best results under all evaluation metrics.

Next, Table 7.2 shows the comparison of three heuristic methods and three variations of SVM learning algorithms: 1) joint SVM, where one single model is learned from all the placing tasks in the training dataset; 2) independent SVM, which treats each task as a independent learning problem and learns a separate model per task; 3) shared sparsity SVM (Section 7.4.3), which also learns one model per task but with parameter sharing. Both independent and shared sparsity SVM use voting to rank placements for the test case.

Table 7.2 shows that all the learning methods (last four rows) outperform heuristic rules under all evaluation metrics. Not surprisingly, the chance method performs poorly (with Prec@5=0.1 and AUC=0.49) because there are very few preferred placements in the large sampling space of possible placements. The two heuristic methods perform well in some obvious cases such as using flat-surface-upright method for table or lowest-point method for rack1.

However, their performance varies significantly in non-trivial cases, including the stemware holder and the hook. This demonstrates that it is hard to script a placing rule that works universally.

We get close to perfect results for the SESO case—i.e., the learning algorithm can very reliably predict object placements if a known object was being placed in a previously seen location. The learning scenario NENO is extremely challenging—here, for each task (of an object-area pair), the algorithm is trained without either the object or the placing area in the training set. In this case, R_0 increases from 1.0 to 8.9 with joint SVM, and to 5.4 using independent SVM with voting. However, shared sparsity SVM (the last row in the table) helps to reduce the average R_0 down to 2.1. While shared sparsity SVM outperforms other algorithms, the result also indicates that independent SVM with voting is better than joint SVM. This could be due to the large variety in the placing situations in the training set. Thus imposing one model for all tasks decreases the performance. We also observed that in cases where the placing strategy is very different from the ones trained on, the shared sparsity SVM does not perform well. For example, R_0 is 7.0 for spoon-in-pen-holder and is 5.0 for disk-on-hook. This issue could potentially be addressed by expanding the training dataset. For comparison, the average AUC (area under ROC curve) of shared sparsity SVM is 0.89, which compares to 0.94 in Table 7.5 for corresponding classes (dish-racks, stemware holder and pen holder).

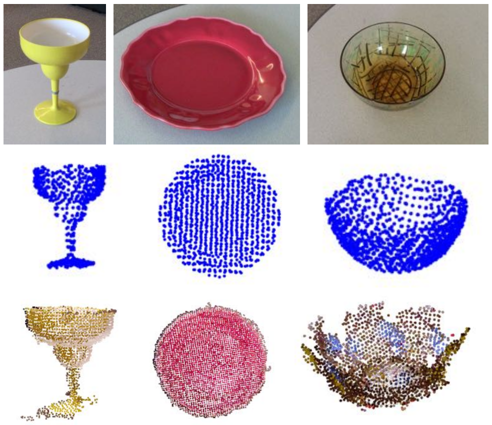


Figure 7.5: Three objects used in our robotic experiments. The top row shows the real objects. Center row shows the perfect point-clouds extracted from object models. Bottom row shows the raw point-clouds perceived from the Kinect sensor, used in the robotic experiments.

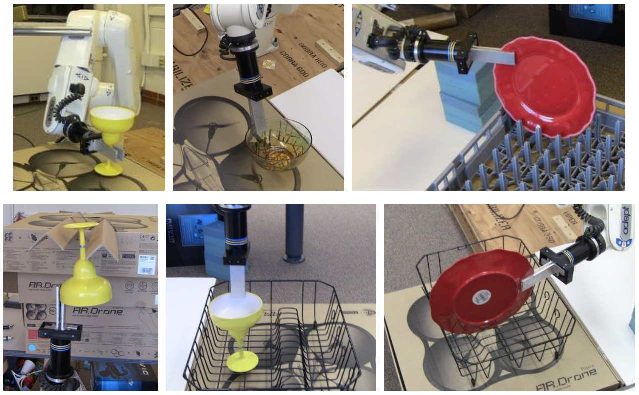


Figure 7.6: Robotic arm placing different objects in several placing areas: a martini glass on a flat surface, a bowl on a flat surface, a plate in rack1, a martini glass on a stemware holder, a martini glass in rack3 and a plate in rack3.

7.5.6 Robotic Experiments

We conducted single-object placing experiments on our PANDA robot with the Kinect sensor, using the same dataset (Section 7.5.1) for training. We tested 10 different placing tasks with 10 trials for each.

Table 7.3: Robotic experiments. The algorithm is trained using shared sparsity SVM under the two learning scenarios: SESO and NENO. 10 trials each are performed for each object-placing area pair. P_s stands for $P_{\text{stability}}$ and P_p stands for $P_{\text{preference}}$. In the experiments with object models, R_0 stands for the rank of first predicted placements passed the stability test. In the experiments without object models, we do not perform stability test and thus R_0 is not applicable. In summary, robotic experiments show a success rate of 98% when the object has been seen before and its 3D model is available, and show a success-rate of 82% (72% when also considering semantically correct orientations) when the object has not been seen by the robot before in any form.

			plate			martini				bowl			**Average**
			rack1	rack3	flat	rack1	rack3	flat	stem.	rack1	rack3	flat	
w/ obj models	SESO	R_0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	**1.0**
		P_s(%)	100	100	100	100	100	100	80	100	100	100	**98**
		P_p(%)	100	100	100	100	100	100	80	100	100	100	**98**
	NENO	R_0	1.8	2.2	1.0	2.4	1.4	1.0	1.2	3.4	3.0	1.8	1.9
		P_s(%)	100	100	100	80	100	80	100	100	100	100	96
		P_p(%)	100	100	100	60	100	80	100	80	100	100	92
w/o obj models	SESO	R_0	-	-	-	-	-	-	-	-	-	-	-
		P_s(%)	100	80	100	80	100	100	80	100	100	100	94
		P_p(%)	100	80	100	80	80	100	80	100	100	100	92
	NENO	R_0	-	-	-	-	-	-	-	-	-	-	-
		P_s(%)	80	60	100	80	80	100	70	70	100	80	**82**
		P_p(%)	80	60	100	60	70	80	60	50	80	80	**72**

In each trial, the robot had to pick up the object and place it in the designated area. The input to our algorithm in these experiments was raw point-clouds of the object and the placing area (see Fig. 7.5). Given a placement predicted by our learning algorithm and a feasible grasp, the robot used path planning to move the object to the destination and released it. A placement was considered successful if it was stable (the object remained still for more than a minute) and in its preferred orientation (within ±15° of the ground-truth orientation after placing).

Table 7.3 shows the results for three objects being placed by the robotic arm in four placing scenarios (see the bottom six rows). We obtain a 94% success rate in placing the objects stably in SESO case, and 92% if we disqualify those stable placements that were not preferred ones. In the NENO case, we achieve 82% performance for stable placing, and 72% performance for preferred placing. Figure 7.6 shows several screenshots of our robot placing the objects. There were some cases where the martini glass and the bowl were placed horizontally in rack1. In these cases, even though the placements were stable, they were not counted as preferred. Even small displacement errors while inserting the martini glass in the narrow slot of the stemware holder often resulted in a failure. In general, several failures for the bowl and the martini glass were due to incomplete capture of the point-cloud which resulted in the object hitting the placing area (e.g., the spikes in the dish-rack).

In order to analyze the source of the errors in robotic placing, we did another experiment in which we factored away the errors caused by the incomplete point-clouds. In detail, we recovered the full 3D geometry by registering the raw point-cloud against a few parameterized objects in a database using the Iterative Closest Point (ICP) algorithm [108, 31]. Furthermore, since we had access to a full solid 3D model, we verified the stability of the placement using the rigid-body simulation before executing it on the robot. If it failed the stability test, we would try the placement with the next highest score until it would pass the test. Even though this simulation was computationally expensive,[3] we only needed to compute this for a few top scored placements.

In this setting with a known library of object models, we obtain a 98% success rate in placing the objects in SESO case. The robot failed only in one experi-

[3] A single stability test takes 3.8 second on a 2.93GHz dual-core processor on average.

ment, when the martini glass could not fit into the narrow stemware holder due to a small displacement occurred in grasping. In the NENO case, we achieve 96% performance in stable placements and 92% performance in preferred placements. This experiment indicates that with better sensing of the point-clouds, our learning algorithm can give better performance.

7.6 Algorithm for Placing Multiple Objects

We will now describe our approach for placing multiple objects in a scene, where we also need to decide which placing area in the scene is semantically suitable for placing every object in.

We first state our assumptions in this setting. We consider two scenarios: 1) the object is placed directly on the placing area; 2) the object is stacked on another object. While an unlimited number of objects can stack into one pile in series (e.g., the plates and bowls in the cabinet in Fig. 7.7), we do not allow more than one object to be placed on one single object and we do not allow one object to be placed on more than one object. We refer this assumption as **'chain stacking'**. For instance, in Fig. 7.7, it is not allowed to place two strawberries in the small bowl, nor is it allowed to place a box on top of the blue and the orange sauce bowls at the same time. Note that this constraint does not limit placing multiple objects on a placing area directly, e.g., the dish-rack is given as a placing area and therefore we can place multiple plates in it.

A physically feasible placing strategy needs to satisfy two constraints: 1) **Full-coverage**: every given object must be placed somewhere, either directly on a placing area or on top of another object. 2) **Non-overlap**: two objects cannot

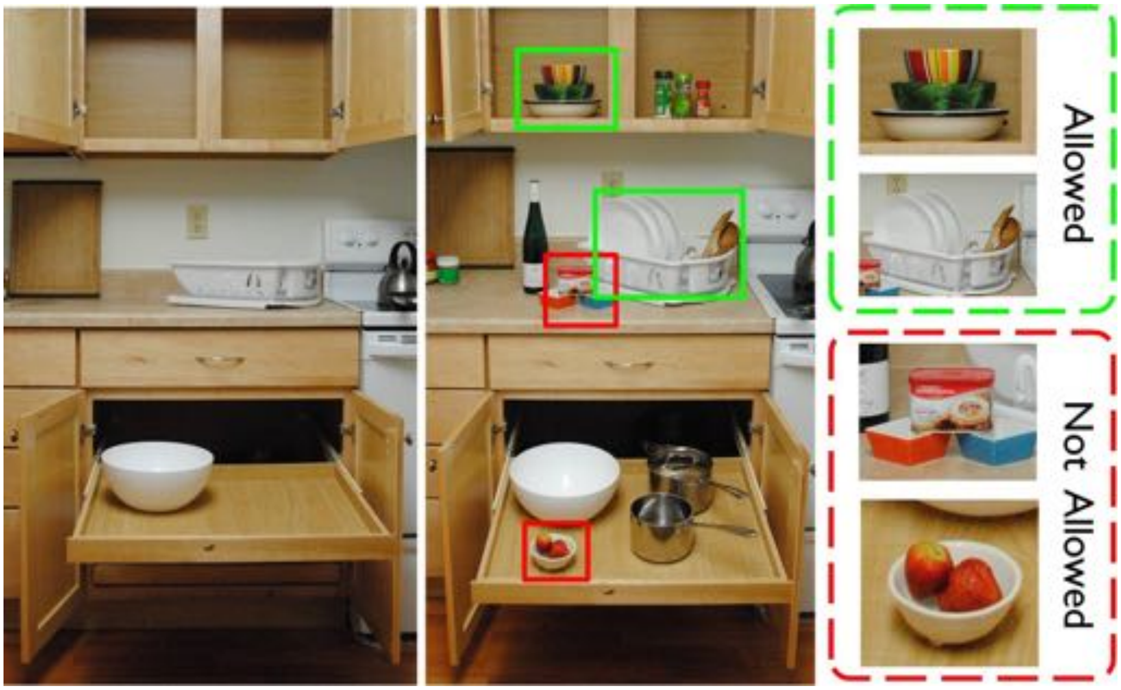


Figure 7.7: Given an initial kitchen scene (left), a possible placing strategy for multiple objects could be as shown in the middle image: loading the dish-rack with spatulas and plates, or stacking them up in the cabinet, storing saucepans on the bottom drawer, etc. In this paper, we only allow chain stacking (see text in Section 7.6), which allows most but not all the possible placing situations (right column).

be placed at same location, although being in the same placing area is allowed. For example, multiple plates can be loaded into one dish-rack, however, they cannot occupy the same slot at the same location. 3) **Acyclic**: placing strategy should be acyclic, i.e., if object A is on top of B (either directly or indirectly), then B cannot be above A.

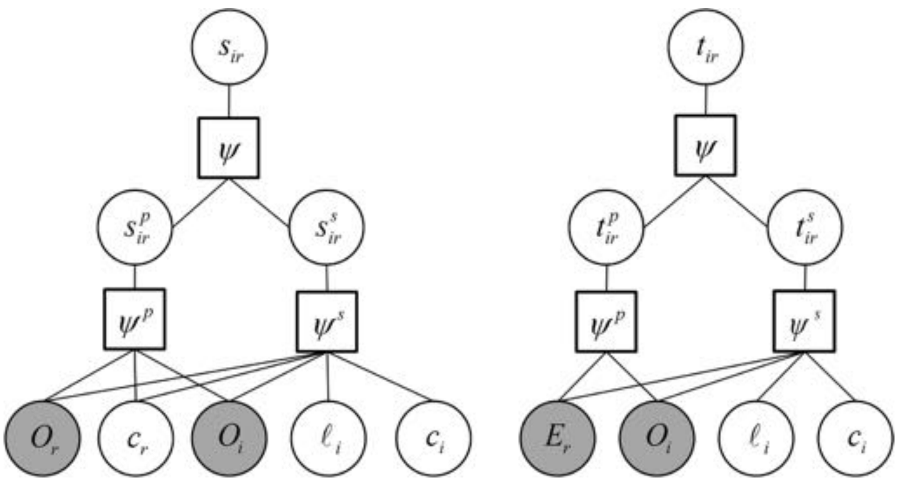


Figure 7.8: Graphical models for two types of single placements: stacking on another object (left) and directly placing on an environment (right). The shaded nodes are observed point-clouds.

7.6.1 Graphical Model

We now introduce our representation of a placing strategy as a graphical model, followed by our design of features, max-margin learning algorithm, and inference as a linear programming problem.

As we mentioned in Section 7.3.2, given n objects $\mathcal{O} = \{O_1, \ldots, O_n\}$ and m placing areas (or environments) $\mathcal{E} = \{E_1, \ldots, E_m\}$ our goal is to find a placing strategy, that can be specified by a tuple (S, T, C, L):

- $S = \{s_{ir} \in \{0,1\} | 1 \leq i \leq n, 1 \leq r \leq n\}$: whether O_i is stacking on top of O_r.
- $T = \{t_{ir} \in \{0,1\} | 1 \leq i \leq n, 1 \leq r \leq m\}$: whether O_i is put directly on E_r.
- $C = \{c_i \in \mathrm{SO}(3) | 1 \leq i \leq n\}$: the configuration (i.e., 3D orientation) of O_i.
- $L = \{\ell_i \in \Re^3 | 1 \leq i \leq n\}$: the 3D location of O_i w.r.t. its base.

We now design a potential function over all placing strategies given a scene, i.e., $\Psi(S, T, L, C, \mathcal{O}, \mathcal{E})$. This function reflects the placing quality and we associate higher potential to better placing strategies. The best strategy is the one with the maximum potential. Therefore, our goal is,

$$(S^*, T^*, L^*, C^*) = \arg\max_{S,T,L,C} \Psi(S, T, L, C, \mathcal{O}, \mathcal{E}) \tag{7.4}$$

where the solution (S, T, L, C) should follow certain constraints that we will discuss in Section 7.6.4.

We use an undirected graphical model (Markov networks [72]) for defining the potential. Fig. 7.8 shows two simplified graphical models for a single placement. The entire graphical model for multiple objects and placing areas is an assembly of these basic structures (one for each s_{ir} and t_{ir}) with shared O, E, ℓ and c nodes. We now explain this graphical model and show how to factorize $\Psi(S, T, L, C, \mathcal{O}, \mathcal{E})$ into small pieces so that learning and inference is tractable.

Our graphical model indicates the following independence:

- s_{ir} and t_{ir} are conditionally independent given $L, C, \mathcal{O}, \mathcal{E}$.
- $s_{ir}(t_{ir})$ only depends on objects (environments) involved in this single placing task of placing O_i on O_r (E_r). This implies that, when placing an object i on object j, it does not matter where object j is placed.

Consequently, we can factorize the overall potential as,

$$\begin{aligned}\Psi(S, T, L, C, \mathcal{O}, \mathcal{E}) \;\; &= \;\; \prod_{i,r} \Psi(s_{ir}, O_i, \ell_i, c_i, O_r, c_r) \\ &\quad \times \prod_{i,r} \Psi(t_{ir}, O_i, \ell_i, c_i, E_r)\end{aligned} \tag{7.5}$$

We further factorize the potential of each placement into three terms to encode the stability and semantic preference in placing, as shown in Fig. 7.8. For each placement, we introduce two binary variables indicating its stability (with superscript s) and semantic preference (with superscript p). Considering they are latent, we have,

$$\begin{aligned}&\Psi(s_{ir},O_i,\ell_i,c_i,O_r,c_r)=\\&\textstyle\sum_{s_{ir}^s,s_{ir}^p\in\{0,1\}}\psi(s_{ir},s_{ir}^s,s_{ir}^p)\psi^s(s_{ir}^s,O_i,\ell_i,c_i,O_r,c_r)\psi^p(s_{ir}^p,O_i,O_r,c_r)\\&\Psi(t_{ir},O_i,\ell_i,c_i,E_r)=\\&\textstyle\sum_{t_{ir}^s,t_{ir}^p\in\{0,1\}}\psi(t_{ir},t_{ir}^s,t_{ir}^p)\psi^s(t_{ir}^s,O_i,\ell_i,c_i,E_r)\psi^p(s_{ir}^p,O_i,E_r)\end{aligned}\tag{7.6}$$

The intuition is that a placement is determined by two factors: stability and semantic preference. This is quantified by the potential function ψ. ψ^s captures stability which is only dependent on local geometric information, i.e., exact poses and locations. On the other hand, semantic preference concerns how well this object fits the environment. So the function ψ^p is determined by the object and the base regardless of the details of the placement. For example, placing a plate in a dish-rack has a high semantic preference over the ground, but different placements (vertical vs. horizontal) would only change its stability. Note that in stacking, semantic preference (s_{ir}^p) is also dependent on the configuration of the base (c_r), since the base's configuration would affect the context. For instance, we can place different objects on a book lying horizontally, but it is hard to place any object on a book standing vertically.

Based on the fact that a good placement should be stable as well as semantically correct, we set $\psi(s_{ir}, s_{ir}^s, s_{ir}^p) = 1$ if $s_{ir} = (s_{ir}^s \wedge s_{ir}^p)$ otherwise 0. We do the same for $\psi(t_{ir}, t_{ir}^s, t_{ir}^p)$. As for the potential functions ψ^s and ψ^p, they are based on a collection of features that indicate good stability and semantic preference. Their parameters are learned using the max-margin algorithm described in Section 7.6.3.

Table 7.4: Features for multiple objects placements and their dimensions ('Dim').

Feature descriptions	Dim	Feature descriptions	Dim
Stability	**178**	**Semantic preference**	**801**
Supporting contacts	12	Zernike	37×4
Caging features	4	BOW	100×4
Histograms	162	Color histogram	46×4
		Curvature histogram	12×4
		Overall shape	5×4
		Relative height	1

7.6.2 Features

In the multiple-object placements, we introduce additional semantic features for choosing a good placing area for an object. Our stability features ϕ_s are similar to those described in Section 7.4.1.[4] Semantic features ϕ_p depend only on O and B, where O denotes the point-cloud of object O to be placed and B denote the point-cloud of the base (either an environment or another object). This is because the semantic features should be invariant to different placements of the object within the same placing area. We describe them in the following:

- **3D Zernike Descriptors:** Often the semantic information of an object is encoded in its shape. Because of their rotation- and translation-invariant property, we apply 3D Zernike descriptors [99] to O and B for capturing their geometric information. This gives us 37 values for a point-cloud.
- **Bag-of-words (BOW) Features:** Fast Point Feature Histograms (FPFH) [109] are persistent under different views and point densities and pro-

[4] In multiple-object placement experiments, we only use the relative height caging features with $n_r = 1$ and $n_\theta = 4$, and use 81-bin (9×9) grid without the ratios for the histogram features.

duce different signatures for points on different geometric surfaces such as planes, cylinders, spheres, etc. We compute a vocabulary of FPFH signatures by extracting 100 cluster centers from FPFH of every point in our training set. Given a test point-cloud, we compute the FPFH signature for each point and associate it with its nearest cluster center. The histogram over these cluster centers makes our BOW features.

- **Color Histograms:** The features described above capture only the geometric cues, but not the visual information such as color. Color information can give clues to the texture of an object and can help identify some semantic information. We compute a 2D histogram of hue and saturation (6×6) and a 1D 10-bin histogram of intensity, thus giving a total of 46 features.
- **Curvature Histograms:** We estimated the curvature of every point using Point Cloud Library [110], and then compute its 12-bin histogram.
- **Overall Shape:** Given a point-cloud, we compute three Eigenvalues of its covariance matrix ($\lambda_1 \geq \lambda_2 \geq \lambda_3$). We then compute their ratios (λ_2/λ_1, λ_3/λ_2), thus giving us a total of 5 features.

We extract the aforementioned semantic features for both O and B separately. For each, we also add their pairwise product and pairwise minimum, thus giving us 4 values for each feature. The last feature is the height to the ground (only for placing areas), thus giving a total of 801 features. We summarize the semantic features in Table 7.4.

7.6.3 Max-margin Learning

Following [132], we choose the log-linear model for potential functions ψ^s and ψ^p,[5]

$$\log \psi^s(s_{ir}^s) \propto \theta_s^\top \phi_s(O_i, \ell_i, c_i, O_r, c_r) \tag{7.7}$$

where θ_s is the weight vector of the features and $\phi_s(\cdot)$ is the feature vector for stability. Let θ_p and $\phi_p(\cdot)$ denote the weight and features for semantic preference respectively. The discriminant function for a placing strategy is given by (from Eq. (7.5))

$$f(S,T) = \sum_{ir} \log \Psi(s_{ir}) + \sum_{ir} \log \Psi(t_{ir}) \tag{7.8}$$

In the learning phase, we use supervised learning to estimate θ_s and θ_p. The training data contains placements with ground-truth stability and semantic preference labels (we describe it later in Section 7.7.1). By Eq. (7.6),

$$f(S,T) = \sum_{ir} \theta_s^T \phi_s(s_{ir}^s) + \theta_p^T \phi_p(s_{ir}^p) + \sum_{ir} \theta_s^T \phi_s(t_{ir}^s) + \theta_p^T \phi_p(t_{ir}^p) \tag{7.9}$$

A desirable θ_s and θ_p should be able to maximize the potential of a good placing strategy (S,T), i.e., for any different placing strategy (S',T'), $f(S,T) > f(S',T')$. Furthermore, we want to maximize this difference to increase the confidence in the learned model. Therefore, our objective is,

$$\arg\max_{\theta_s,\theta_p} \gamma \quad \text{s.t.} \quad {\scriptstyle f(S,T)-f(S',T')\geq\gamma, \quad \forall (S',T')\neq(S,T)} \tag{7.10}$$

which, after introducing slack variables to allow errors in the training data, is

[5] For conciseness, we use $\psi(s_{ir}^s)$ to denote the full term $\psi^s(s_{ir}^s, O_i, \ell_i, c_i, O_r, c_r)$ explicitly in the rest of the paper unless otherwise clarified. We do the same for ψ^p.

equivalent to[6]

$$\arg\min_{\theta_s,\theta_p} \tfrac{1}{2}(||\theta_s||^2+||\theta_p||^2)+C\sum(\xi^s_{s,ir}+\xi^p_{s,ir}+\xi^s_{t,ir}+\xi^p_{t,ir}) \tag{7.11}$$

$$\text{s.t. } s^s_{ir}(\theta_s^\top\phi_s(\cdot))\geq 1-\xi^s_{s,ir}, \quad s^p_{ir}(\theta_p^\top\phi_p(\cdot))\geq 1-\xi^p_{s,ir}, \tag{7.12}$$

$$t^s_{ir}(\theta_s^\top\phi_s(\cdot))\geq 1-\xi^s_{t,ir}, \quad t^p_{ir}(\theta_p^\top\phi_p(\cdot))\geq 1-\xi^p_{t,ir}, \forall i,r \tag{7.13}$$

We use max-margin learning [64] to learn θ_s and θ_p respectively.

Note that the learning method in Section 7.4.2 for single-object placements is actually a special case of the above. Specifically, for one object and one placing area, the stacking and the semantic preference problems are trivially solved, and the subscripts i and r are not needed. Therefore, this equation reduces to Eq. (7.2) with θ_s, $\phi_s(\cdot)$ and t^s corresponding to θ, ϕ_i and y_i in Eq. (7.2).

7.6.4 Inference

Once we have learned the parameters in the graphical model, given the objects and placing environment, we need to find the placing strategy that maximizes the likelihood in Eq. (7.4) while satisfying the aforementioned constraints as well:

$$\sum_{i=1}^{n} s_{ir} \leq 1, \quad \forall r; \qquad \text{(chain stacking)} \tag{7.14}$$

$$\sum_{r=1}^{n} s_{ir} + \sum_{r=1}^{m} t_{ir} = 1, \quad \forall i; \qquad \text{(full-coverage)} \tag{7.15}$$

$$\ell_i \neq \ell_j, \quad \forall t_{ir} = t_{jr} = 1, \tag{7.16}$$

$$t_{ir} + t_{jr} \leq 1, \quad \forall O_i \text{ overlaps } O_j. \qquad \text{(non-overlap)} \tag{7.17}$$

Eq. (7.14) states that for every object r, there can be at most one object on its

[6] Here, without loss of the generality, we map the domain of S and T from {0,1} to {-1,1}.

top. Eq. (7.15) states that every object i should be placed exactly at one place. Eq. (7.16) states that two objects cannot occupy the same location. Eq. (7.17) states that even if the objects are at different locations, they still cannot have any overlap with each other. In another word, if O_i placed at ℓ_i conflicts with O_j placed at ℓ_j, then only one of them can be placed. We do not need constrain s_{ir} since 'chain stacking' already eliminates this overlap issue.

Enforcing the acyclic property could be hard due to the exponential number of possible cycles in placing strategies. Expressing all the cycles as constraints is infeasible. Therefore, we assume a topological order on stacking: s_{ir} can be 1 only if O_r with configuration c_r does not have smaller projected 2D area on XY-plane than O_i with configuration c_i. This assumption is reasonable as people usually stack small objects on big ones in practice. This ensures that the optimal placing strategy is acyclic.

As mentioned before, the search space of placing strategies is too large for applying any exhaustive search to. Several prior works have successfully applied ILP to solve inference in Conditional Random Fields or Markov networks, e.g., 107, 131, 146, 32. Motivated by their approach, we formulate the inference (along with all constraints) as an ILP problem and then solve it by LP relaxation, which works well in practice.

We use random samples to discretize the continuous space of the location ℓ_i and configuration c_i.[7] We abuse the notation for S, T and use the same symbols for representing sampled variables. We use $s_{ijrtk} \in \{0, 1\}$ to represent object O_i with the jth sampled configuration is placed on object O_r with the tth configuration at the kth location. Similarly, we use t_{ijrk} to represent placing object O_i in

[7] In our experiments, we randomly sample ℓ_i in about every 10cm × 10cm area. An object is rotated every 45 degree along every dimension, thus generating 24 different configurations.

Table 7.5: Stability results for three baselines and our algorithm using different features (contact, caging, histograms and all combined) with two different kernels (linear and quadratic polynomial).

	dish-racks			stemware-holder			closet			pen-holder			**Average**		
	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC	R_0	P@5	AUC
chance	1.5	0.70	0.50	2.0	0.48	0.50	2.0	0.47	0.50	2.1	0.44	0.50	1.9	0.52	0.50
vert	1.3	0.79	0.74	2.0	0.70	0.77	2.5	0.63	0.71	1.0	1.00	1.00	1.7	0.78	0.80
hori	2.1	0.22	0.48	4.3	0.35	0.54	6.0	0.03	0.36	7.2	0.00	0.38	4.9	0.15	0.44
lin-contact	1.9	0.81	0.80	2.0	0.60	0.71	1.8	0.70	0.79	1.0	0.81	0.91	1.7	0.73	0.80
lin-caging	3.5	0.60	0.74	1.3	0.94	0.95	1.2	0.93	0.77	2.5	0.70	0.85	2.1	0.79	0.83
lin-hist	1.4	0.93	0.93	2.3	0.70	0.85	1.0	1.00	0.99	1.0	1.00	1.00	1.4	0.91	0.94
lin-all	1.2	0.91	0.91	2.0	0.80	0.86	1.0	1.00	1.00	1.0	1.00	1.00	1.3	0.93	0.94
poly-contact	1.2	0.95	0.88	1.0	0.80	0.85	1.2	0.93	0.93	1.2	0.91	0.88	1.1	0.90	0.89
poly-caging	2.1	0.81	0.87	2.0	0.60	0.79	1.0	0.97	0.94	2.2	0.70	0.88	1.8	0.77	0.87
poly-hist	1.2	0.94	0.91	3.0	0.30	0.67	1.0	1.00	1.00	1.0	1.00	1.00	1.6	0.81	0.90
poly-all	1.1	0.95	0.94	1.8	0.60	0.92	1.0	1.00	1.00	1.0	1.00	1.00	1.2	0.89	0.96

configuration j on top of E_r at location k. Now our problem becomes:

$$\begin{aligned}
&\arg\max_{S,T} \textstyle\sum_{ijrk}\big(t_{ijrk}\log\Psi(t_{ijrk}{=}1){+}(1{-}t_{ijrk})\log\Psi(t_{ijrk}{=}0)\big)\\
&+\textstyle\sum_{ijrtk}\big(s_{ijrtk}\log\Psi(s_{ijrtk}{=}1){+}(1{-}s_{ijrtk})\log\Psi(s_{ijrtk}{=}0)\big)\\
&\text{s.t.}\ \textstyle\sum_{ijk} s_{ijrtk}\le\sum_{pqk} s_{rtpqk}+\sum_{pk} t_{rtpk},\ \forall r,t;\\
&\quad\textstyle\sum_{jrtk} s_{ijrtk}+\sum_{jrk} t_{ijrk}=1,\ \forall i;\\
&\quad\textstyle\sum_{ij} t_{ijrk}\le 1,\ \forall r,k;\\
&\quad t_{ijrk}+t_{i'j'rk'}\le 1,\ \forall t_{ijrk}\ \text{overlaps}\ t_{i'j'rk'}.
\end{aligned} \tag{7.18}$$

While this ILP is provably NP-complete in the general case, for some specific cases it can be solved in polynomial time. For example, if all objects have to stack in one pile, then it reduces to a dynamic programming problem. Or if no stacking is allowed, then it becomes a maximum matching problem. For other general cases, an LP relaxation usually works well in practice. We use an open-sourced Mixed Integer Linear Programming solver [8] in our implementation.

7.7 Experiments on Placing Multiple Objects

In these experiments, the input is the raw point-clouds of the object(s) and the scene, and our output is a placing strategy composed of the placing location and orientation for each object. Following this strategy, we can construct the point-cloud of the scene after placing (e.g., Fig. 7.1 top-right) and then use path planning to guide the robot to realize it.

We extensively tested our approach in different settings to analyze different components of our learning approach. In particular, we considered single-object placing, placing in a semantically appropriate area, multiple-object single-environment placing, and the end-to-end test of placing in offices and houses. We also tested our approach in robotic experiments, where our robots accomplished several placing tasks, such as loading a bookshelf and a fridge.

7.7.1 Data

We consider 98 objects from 16 categories (such as books, bottles, clothes and toys shown Table 7.6) and 40 different placing areas in total (e.g., dish-racks, hanging rod, stemware holder, shelves, etc). The robot observes objects using its Kinect sensor, and combines point-clouds from 5 views. However, the combined point-cloud is still not complete due to reflections and self-occlusions, and also because the object can be observed only from the top (e.g., see Fig. 7.1).

Table 7.6: Semantic results for three baselines and our algorithm using different features (BOW, color, curvature, Eigenvalues, Zernike and all combined) with two different kernels (linear and quadratic polynomial) on AUC metric.

	big bottles	books	boxes	clothes	comp. accssry	hats& gloves	dishware	food	misc. house.	martini glasses	mugs	shoes	small bottles	stationery	toys	utensils	AVG
chance	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50	.50
table	.70	.61	.59	.90	.69	.78	.61	1.0	.65	.32	.45	.60	.64	.57	.66	.50	.66
size	.68	.67	.67	.80	.81	.80	.66	1.0	.83	.46	.52	.90	.70	.60	.82	.51	.72
lin-BOW	.90	1.0	.98	.08	.93	.96	.86	1.0	.91	.68	.89	.70	.96	.86	.94	.79	.85
lin-color	.75	1.0	1.0	.15	.96	.93	.59	1.0	.80	.64	.72	.45	.98	.95	.86	.61	.79
lin-curv.	.85	.90	.89	.78	.92	.85	.62	1.0	.74	.61	.67	.60	.86	.82	.83	.48	.79
lin-Eigen	.67	.93	.93	.40	.93	.95	.53	1.0	.87	.43	.47	.90	.89	.89	.96	.53	.79
lin-Zern.	.86	1.0	.91	.20	.86	.89	.73	.80	.83	.79	.74	.65	.80	.82	.84	.59	.77
lin-all	.93	1.0	1.0	.60	.93	.96	.84	1.0	.88	.71	.87	.50	.99	.89	.89	.82	.85
poly-BOW	.86	1.0	.89	.47	.90	.97	.90	1.0	.89	.82	.87	.75	.96	.85	.87	.82	.87
poly-all	.91	1.0	1.0	.62	.90	.99	.93	1.0	.89	.86	.89	.75	1.0	.89	.88	.84	.90

For the environment, only a single-view point-cloud is used. We pre-processed the data and segmented the object from its background to reduce the noise in the point-clouds. For most of the placing areas,[8] our algorithm took real point-clouds. For *all the objects*, our algorithm took only real point-clouds as input and we did not use any assumed 3D model.

[8] For some racks and holders that were too thin to be seen by our sensor, we generated synthetic point-clouds from tri-meshes.

7.7.2 Placing Single New Objects with Raw Point-Cloud as Input

The purpose of this experiment is, similar to Section 7.5, to test our algorithm in placing a *new* object in a scene, but instead with raw point-clouds composing the training data. We test our algorithm on the following four challenging scenes:

1. dish-racks, tested with three dish-racks and six plates;
2. stemware holder, tested with one holder and four martini glass/handled cups;
3. hanging clothes, tested with one wooden rod and six articles of clothing on hangers;
4. cylinder holders, tested with two pen-holder/cup and three stick-shaped objects (pens, spoons, etc.).

Since there is only a single placing environment in every scene, only stability features play a role in this test. We manually labeled 620 placements in total, where the negative examples were chosen randomly. Then we used leave-one-out training/testing so that the testing object is always *new* to the algorithm.

We compare our algorithm with three heuristic methods:

- *Chance*. Valid placements are randomly chosen from the samples.
- *Vertical placing*. Placements are chosen randomly from samples where the object is vertical (the height is greater than the width). This is relevant for cases such as placing plates in dish-racks.

- *Horizontal placing*. Placements are chosen where the object is horizontal (opposed to 'vertical'). This applies to cases when the placing area is somewhat flat, e.g., placing a book on a table.

For the placing scenarios considered here, the heuristic based on finding flat surface does not apply at all. Since we do not know the upright orientation for all the objects and it is not well-defined in some cases (e.g. an article of clothing on a hanger), we do not use the flat-surface-upright rule in these experiments.

Results are shown in Table 7.5. We use the same three evaluation metrics as in Section 7.5.4: R_0, P@5 and AUC. The top three rows of the table shows our three baselines. Very few of these placing areas are 'flat', therefore the horizontal heuristic fares poorly. The vertical heuristic performs perfectly in placing in pen-holder (all vertical orientations would succeed here), but its performance in other cases is close to chance. All the learning algorithms perform better than all the baseline heuristics in the AUC metric.

For our learning algorithm, we compared the effect of different stability features as well as using linear and polynomial kernels. The results show that combining all the features together give the best performance in all metrics. The best result is achieved by polynomial kernel with all the features, giving an average AUC of 0.96.

7.7.3 Selecting Semantically Preferred Placing Areas

In this experiment, we test if our algorithm successfully learns the preference relationship between objects and their placing areas. Given a single object and multiple placing areas, we test whether our algorithm correctly picks out the most suitable placing area for placing that object. As shown in Table 7.6, we tested 98 objects from 16 categories on 11 different placing areas: the ground, 3 dish-racks, a utensil organizer, stemware-holder, table, hanging rod, pen-holder and sink. We exhaustively labeled every pair of object and area, and used leave-one-out method for training and test. Again, we build three baselines where the best area is chosen 1) by chance; 2) if it is a table (many of objects in our dataset can be placed on table); 3) by its size (whether the area can cover the object or not).

Our algorithm gives good performance in most object categories except clothes and shoes. We believe this is because clothes and shoes have a lot of variation making it harder for our algorithm to learn. Another observation is that the heuristic 'table' performs well on clothes. We found that this is because, in our labeled data set, the table was often labeled as the second best placing area for clothes after the hanging rod in the closet. We tested different semantic features with linear SVM and also compared linear and polynomial SVM on all features combined. In comparison to the best baseline of 0.72, we get an average AUC of 0.90 using polynomial SVM with all the features.

7.7.4 Multiple Objects on Single Area

In this section, we consider placing multiple objects in a very limited space: 14 plates in one dish-rack without any stacking; 17 objects including books, dish-ware and boxes on a small table so that stacking is needed; and five articles of clothing with hangers on a wooden rod. This experiment evaluates stacking

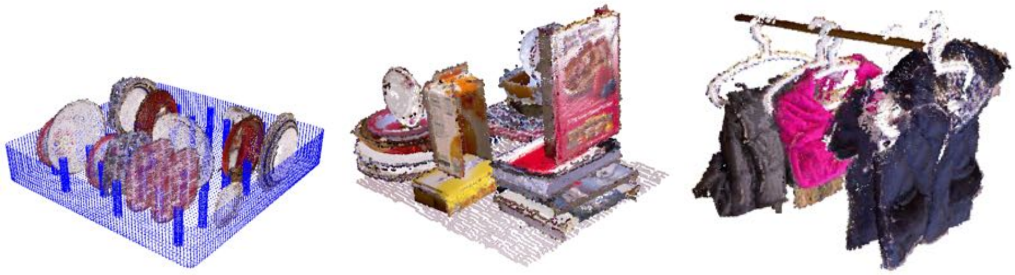


Figure 7.9: Placing multiple objects on a dish-rack (left), a table (middle) and a hanging rod (right). Most objects are placed correctly, such as the plates vertically in the dish-rack, books and plates stacked nicely on the table and the hangers with the clothes aligned on the rod. However, two top-most plates on the table are in wrong configuration and the right-most hanger in the right figure is off the rod.

and our LP relaxation.

The results are shown in Fig. 7.9. In the left image, plates are vertically placed without overlap with spikes and other plates. Most are aligned in one direction to achieve maximum loading capacity. For the second task in the middle image, four piles are formed where plates and books are stacked separately, mostly because of semantic features. Notice that all the dishware, except the top-most two bowls, is placed horizontally. The right image shows all clothes are placed perpendicular upon the rod. However the first and last hangers are little off the rod due to the outliers in the point-cloud which are mistakenly treated as part of the object . This also indicates that in an actual robotic experiment, the placement could fail if there is no haptic feedback.

7.7.5 Placing in Various Scenes

In this experiment, we test the overall algorithm with multiple objects being placed in a scene with multiple placing areas. We took point-clouds from 3 different offices and 2 different apartments. Placing areas such as tables, cabinets, floor, and drawers were segmented out. We evaluated the quality of the final placing layout by asking two human subjects (one male and one female, not associated with the project) to label placement for each object as stable or not, semantically correct or not, and also report a qualitative metric score on how good the overall placing was (0 to 5 scale).

Table 7.7 shows the results for different scenes (averaged over objects and placing areas) and for different algorithms and baselines. We considered baselines where objects were placed *vertically*, *horizontally* or according to configuration *priors* (e.g., flat surface prefers horizontal placing while dish-racks and holders prefer vertical placing). As no semantic cues were used in the baselines, placing areas were chosen randomly. The results show that with our learning algorithm, the end-to-end performance is substantially improved under all metrics, compared to the heuristics. Fig. 7.10 shows the point-cloud of two offices after placing the objects according to the strategy. We have marked some object types in the figure for better visualization. We can see that books are neatly stacked on the left table, and horizontally on the couch in the right image. While most objects are placed on the table in the left scene, some are moved to the ground in the right scene, as the table there is small.

We do not capture certain properties in our algorithm, such as object-object co-occurrence preferences. Therefore, some placements in our results could potentially be improved in the future by learning such contextual relations [e.g., 2]. For example, a mouse is typically placed on the side of a keyboard when placed on a table and objects from the same category are often grouped together.

Table 7.7: End-to-end test results. In each scene, the number of placing areas and objects is shown as $(\cdot,\cdot)$. **St**: % of stable placements, **Co**: % of semantically correct placements, **Sc**: average score (0 to 5) over areas.

	office-1 (7,29)			office-2 (4,29)			office-3 (5,29)			apt-1 (13,51)			apt-2 (8,50)			Average		
	St	Co	Sc	St	Co	Sc	St	Co	Sc	St	Co	Sc	St	Co	Sc	St	Co	Sc
Vert.	36	60	2.4	33	52	2.4	52	59	2.8	38	41	1.9	46	59	3	39	59	2.4
Hori.	50	62	2.9	57	52	2.7	69	60	3.4	54	50	2.5	60	53	2.7	57	58	2.7
Prior	45	52	2.4	64	59	2.8	69	60	3.5	44	43	2.3	61	55	2.7	55	53	2.7
Our approach	78	79	4.4	83	88	4.9	90	81	4.5	81	80	3.8	80	71	4.2	83	82	4.4

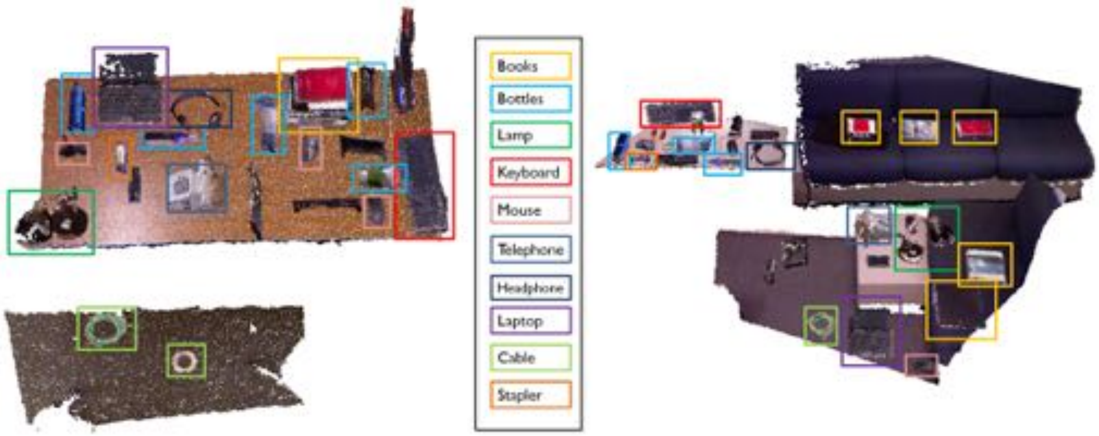


Figure 7.10: Two office scenes after placing, generated by our algorithm. Left scene comprises a table and ground with several objects in their final predicted placing locations. Right scene comprises two tables, two couches and ground with several objects placed.

7.7.6 Robotic Experiment

We tested our approach on both our robots, the PANDA and POLAR, with the Kinect sensor. We performed the end-to-end experiments as follows. (For quantitative results on placing single objects, see Section 7.5.6.)

In the first experiment, our goal was to place 16 plates in a dish-rack. Once the robot inferred the placing strategy (an example from the corresponding of-

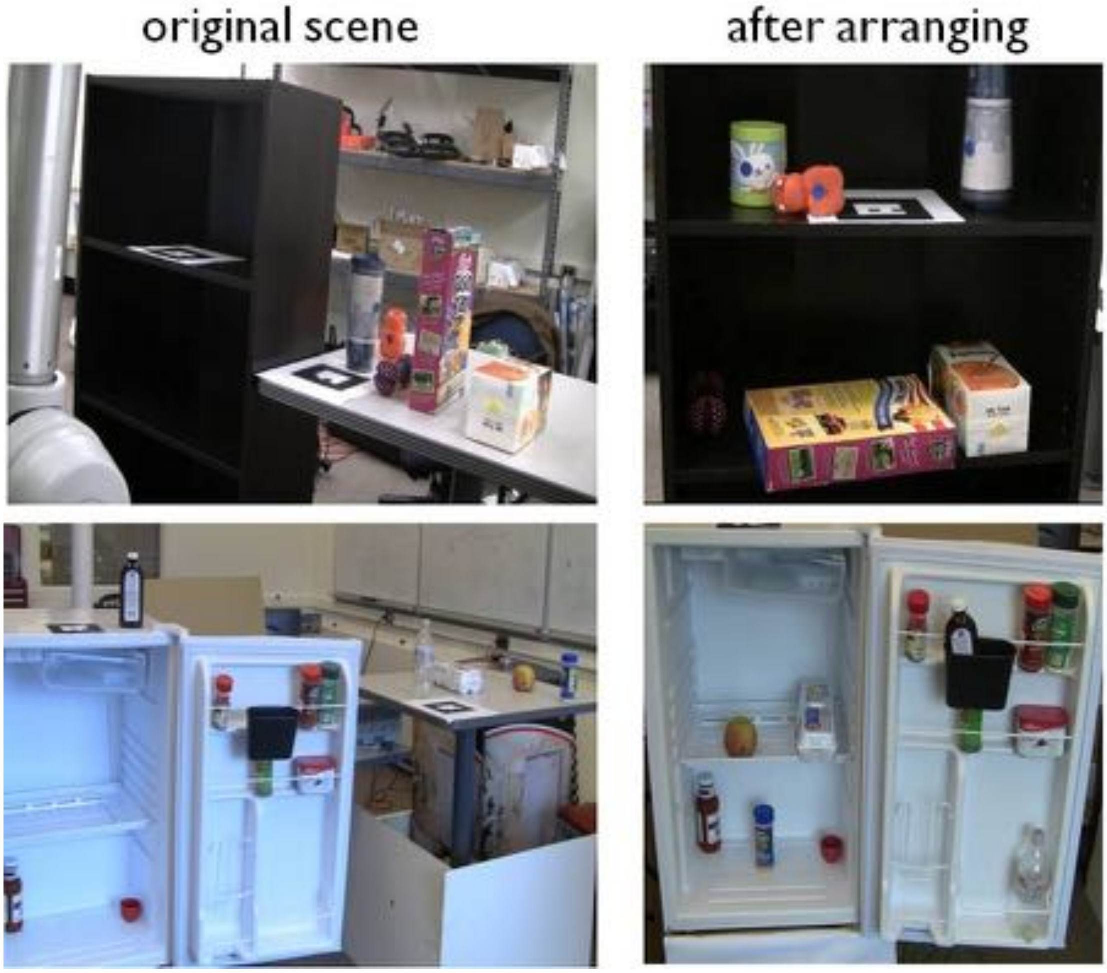
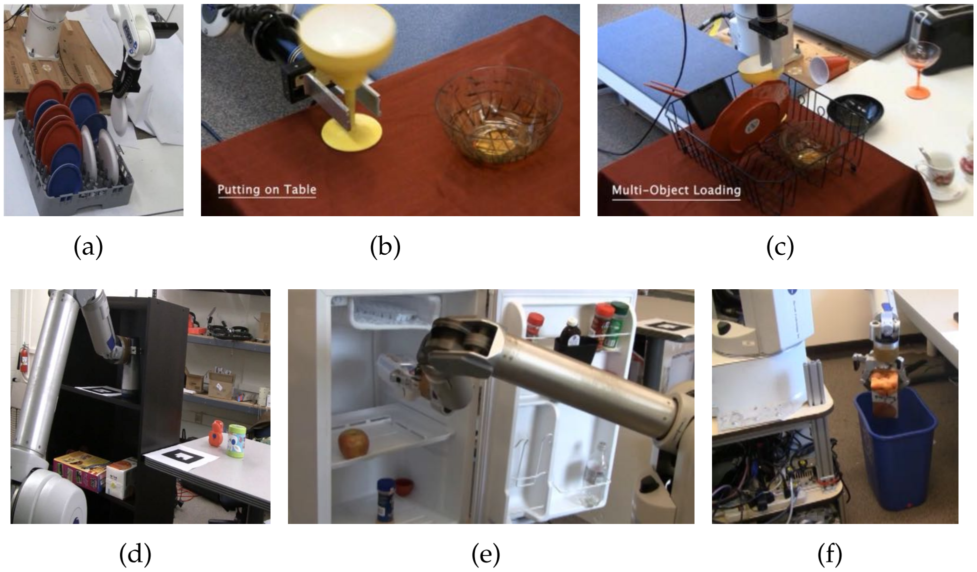


Figure 7.11: Loading a bookshelf (top) and a fridge (bottom). Snapshots of the scenes before and after the arrangement by our robot POLAR using our learning algorithm.

fline experiment is shown in Fig. 7.9-left), it placed the plates one-by-one in the dish-rack (Fig. 7.12(a)). Out of 5 attempts (i.e., a total of 5x16=80 placements), less than 5% of placements failed. This is because the plate moved within the gripper after grasping.

In the second experiment, our aim was to place a cup, a plate and a martini glass together on a table and then into a dish-rack. The objects were originally



(a) (b) (c)

(d) (e) (f)

Figure 7.12: Robots placing multiple objects in different scenes. Top row shows PANDA: (a) loading a dish-rack with plates, (b) placing different objects on a table, and (c) placing different objects in a dish-rack. Bottom row shows POLAR: (d) placing six items on a bookshelf, (e) loading five items in a fridge, and (d) placing an empty juice box in a recycle bin.

placed at pre-defined locations and were picked up using the given grasps. The result is shown in Fig. 7.12(b) and 7.12(c).

In the third experiment, the robot was asked to arrange a room with a bookshelf, a fridge, a recycle-bin and a whiteboard as potential placing areas. The room contained 14 objects in total, such as bottles, rubber toys, a cereal box, coffee mug, apple, egg carton, eraser, cup, juice box, etc. Our robot first inferred the placing strategy and then planned a path and placed each one. In some cases when the object was far from its placing location, the robot moved to the object, picked it up, moved to the placing area and then placed it. Fig. 7.12 (last row) shows some placing snapshots and the scene before and after placing is shown in Fig. 7.11.

CHAPTER 8

CONCLUSION

As humans cast such a substantial influence on our environments, to understand our environment, one needs to reason it through the interplay between the humans and objects. The key motivation of this thesis is to capture the human-object relations, referred as 'object affordances', in an environment. Even for environment with only objects physically present, we argued that it can be modeled better through the hallucinated humans and object affordances.

We first quantitatively defined object affordances through parametric functions which can describe the spatial relation between a given human pose and an object configuration. For a scene that may contain multiple objects and humans, we augmented the classic Conditional Random Field (CRF) to capture both human context and object context through human-object and object-object edges with different potential functions.

The biggest challenge lies in how to model hidden humans and unknown object affordances. We therefore proposed a new non-parametric model, called Infinite Latent Conditional Random Fields (ILCRFs) which can handle: 1) unknown number of latent nodes (for potential human poses), 2) unknown number of edge types (for human-object interactions), and 3) a mixture of different CRFs (for the whole scene). We also presented a Gibbs-sampling based learning and inference algorithm for ILCRFs.

We applied our ILCRFs to two different tasks: 3D scene labeling and 3D scene arrangement. Through extensive experiments and thorough analyses, we not only showed that our ILCRF algorithm outperforms the state-of-the-art results, but we also verified that modeling latent human poses and their relationships to objects are crucial to reason our environment. In addition to hallucinate static human poses, we also proposed a Gaussian Process Latent CRF that can model high-dimensional human motions and thus can be used to anticipate human activities.

We have performed extensive robotic experiments to verify our algorithms throughout this work. Using ILCRFs, our robot correctly inferred potential human poses, object labels and object arrangements in real scenes. Moreover, we also considered basic robotic manipulation tasks such as grasping and placing. We developed learning-based approaches to be able to handle novel objects robustly.

We believe that the idea of hallucinating humans and object affordances can be applied to many scene understanding tasks, such as detecting object attributes, and even detecting functionalities of a scene. This could further facilitate personal robots to arrange scenes according to different functions. In computer graphics, one may also use this idea to automatically generate proper objects in a scene.

The ability to predict human activities (with detailed trajectory) could make a significant difference to robots working in the presence of humans. With more accurate human trajectory prediction, robots can plan more relevant actions and paths [77, 48]. Furthermore, with real-time anticipation, our work can be used for human-robot interaction, such as to improve the efficiency of collaborative tasks [142, 39], or to avoid intrusion/collision during navigation [68].

BIBLIOGRAPHY

[1] A. Anand, H. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for 3d point clouds. *IJRR*, 32(1):19–34, 2012.

[2] Abhishek Anand, Hema Swetha Koppula, Thorsten Joachims, and Ashutosh Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011.

[3] A. Anandkumar, D. Hsu, F. Huang, and S. Kakade. Learning mixtures of tree graphical models. In *NIPS*, 2012.

[4] M.J. Beal, Z. Ghahramani, and C.E. Rasmussen. the infinite hidden markov model. *NIPS*, 2002.

[5] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *IJRR*, 24(1):31–48, 2005.

[6] D. Berenson, R. Diankov, K. Nishiwaki, S. Kagami, and J. Kuffner. Grasp planning in complex scenes. In *Int'l conf Humanoid Robots*, 2009.

[7] A.C. Berg, T.L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

[8] Michel Berkelaar, Kjell Eikland, and Peter Notebaert. lp_solve 5.5, open source (mixed-integer) linear programming system. Software, May 1 2004.

[9] A. Bicchi and V. Kumar. Robotic grasping and contact: a review. In *International Conference on Robotics and Automation (ICRA)*, 2000.

[10] K. Bousmalis, L.P. Morency, S. Zafeiriou, and M. Pantic. A discriminative nonparametric bayesian model: Infinite hidden conditional random fields. In *NIPS Workshop on Bayesian Nonparametrics*, 2011.

[11] Peter Brook, Matei Ciocarlie, and Kaijen Hsiao. Collaborative grasp planning with multiple object representations. In *ICRA*, 2011.

[12] Matei T. Ciocarlie and Peter K. Allen. On-line interactive dexterous grasping. In *Eurohaptics*, 2008.

[13] E.G. Coffman Jr, M.R. Garey, and D.S. Johnson. Approximation algorithms for bin packing: A survey. In *Approximation algorithms for NP-hard problems*, pages 46–93. PWS Publishing Co., 1996.

[14] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[16] E. Demircan, T. Besier, S. Menon, and O. Khatib. Human motion reconstruction and synthesis of human skills. In *Advances in Robot Kinematics: Motion in Man and Machine*, pages 283–292. Springer, 2010.

[17] J. Deng, A. Berg, and L. Fei-Fei. Hierarchical semantic indexing for large scale image retrieval. In *CVPR*, 2011.

[18] R. Diankov, S.S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation planning with caging grasps. In *Humanoid Robots, 2008. Humanoids 2008. 8th IEEE-RAS International Conference on*, pages 285–292. IEEE, 2008.

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

[20] M.R. Dogar and S.S. Srinivasa. A framework for push-grasping in clutter. In *RSS*, 2011.

[21] H. Dreyfuss. *Designing for People*. Allworth Press, 1955.

[22] J.G. Dy and C.E. Brodley. Feature selection for unsupervised learning. *The Journal of Machine Learning Research*, 5:845–889, 2004.

[23] A. Edsinger and C.C. Kemp. Manipulation in human environments. In *Humanoid Robots*, 2006.

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

[25] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003.

[26] M. Fisher and P. Hanrahan. Context-based search for 3d models. *ACM TOG*, 29(6), 2010.

[27] M. Fisher, M. Savva, and P. Hanrahan. Characterizing structural relationships in scenes using graph kernels. *SIGGRAPH*, 2011.

[28] M.L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management science*, pages 1–18, 1981.

[29] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer. Upright orientation of man-made objects. *ACM Trans. Graph.*, 27:42:1–42:7, 2008.

[30] A. Geiger, R. Urtasun, and T. Darrell. Rank priors for continuous nonlinear dimensionality reduction. In *CVPR*, 2009.

[31] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Eurographics Symposium on Geometry Processing*, 2005.

[32] A. Globerson and T. Jaakkola. Fixing max-product: Convergent message passing algorithms for map lp-relaxations. In *NIPS*, 2007.

[33] Jared Glover, Radu Rusu, and Gary Bradski. Monte carlo pose estimation with quaternion kernels and the bingham distribution. In *Proceedings of Robotics: Science and Systems*, 2011.

[34] Helmut Grabner, Juergen Gall, and Luc J. Van Gool. What makes a chair a chair? In *CVPR*, 2011.

[35] T.L. Griffiths and Z. Ghahramani. The indian buffet process: An introduction and review. *JMLR*, 12:1185–1224, 2011.

[36] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popovic. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[37] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011.

[38] J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating (roc) curvel characteristic. *Radiology*, 143(1):29–36, 1982.

[39] K. P Hawkins, N. Vo, S. Bansal, and A Bobick. Probabilistic human action prediction and wait-sensitive planning for responsive human-robot collaboration. In *HUMANOIDS*, 2013.

[40] V. Hedau, D. Hoiem, and D.A. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

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

[42] G Heitz, S Gould, A Saxena, and D Koller. Cascaded classification models: Combining models for holistic scene understanding. In *Neural Information Processing Systems (NIPS)*, 2008.

[43] Geremy Heitz, Stephen Gould, Ashutosh Saxena, and Daphne Koller. Cascaded classification models: Combining models for holistic scene understanding. In *NIPS*, 2008.

[44] Edmond S. L. Ho, Taku Komura, and Chiew-Lan Tai. Spatial relationship preserving character motion adaptation. *ACM Trans. Graph.*, 29(4), 2010.

[45] Kaijen Hsiao, Paul Nangeroni, Manfred Huber, Ashutosh Saxena, and Andrew Y. Ng. Reactive grasping using optical proximity sensors. In *ICRA*, 2009.

[46] K. Ickstadt, B. Bornkamp, M. Grzegorczyk, J. Wieczorek, M. Sheriff, H. Grecco, and E. Zamir. Nonparametric bayesian networks. *Bayesian Statistics*, 9:283–316, 2010.

[47] A. Jain and C.C. Kemp. Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 1807–1814. IEEE, 2010.

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

[49] A. Jalali, P. Ravikumar, S. Sanghavi, and C. Ruan. A Dirty Model for Multi-task Learning. *NIPS*, 2010.

[50] J. Jancsary, S. Nowozin, and C. Rother. Non-parametric crfs for image labeling. In *NIPS Workshop Modern Nonparametric Methods Mach. Learn.*, 2012.

[51] Nikolay Jetchev and Marc Toussaint. Task space retrieval using inverse feedback control. In *ICML*, pages 449–456, 2011.

[52] Zhaoyin Jia, Andy Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d-based reasoning with blocks, support, and stability. In *CVPR*, 2013.

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

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

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

[56] Y. Jiang, S. Moseson, and A. Saxena. Efficient Grasping from RGBD Images: Learning using a new Rectangle Representation. In *ICRA*, 2011.

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

[58] Y. Jiang and A. Saxena. Infinite latent conditional random fields for modeling environments through humans. In *RSS*, 2013.

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

[60] Y. Jiang, C. Zheng, M. Lim, and A. Saxena. Learning to place new objects. In *ICRA*, 2012.

[61] Yun Jiang, John Amend, Hod Lipson, and Ashutosh Saxena. Learning hardware agnostic grasps for a universal jamming gripper. In *ICRA*, 2012.

[62] Yun Jiang and Ashutosh Saxena. Discovering different types of topics: Factored topic models. In *IJCAI*, 2013.

[63] Yun Jiang, Changxi Zheng, Marcus Lim, and Ashutosh Saxena. Learning to place new objects. In *ICRA*, 2012.

[64] T. Joachims. *Making large-Scale SVM Learning Practical*. MIT-Press, 1999.

[65] T. Joachims, T. Finley, and CN. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[66] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE PAMI*, 21(5):433–449, 1999.

[67] D. Katz and O. Brock. Manipulating articulated objects with interactive perception. In *ICRA*, 2008.

[68] W.G. Kennedy, M. D Bugajska, M. Marge, W. Adams, B. R Fransen, D. Perzanowski, A. C Schultz, and J. G. Trafton. Spatial representation and reasoning for human-robot collaboration. In *AAAI*, 2007.

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

[70] H. Kjellström, J. Romero, and D. Kragić. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 115(1):81–90, 2011.

[71] Ellen Klingbeil, Ashutosh Saxena, and Andrew Y. Ng. Learning to open new doors. In *IROS*, 2010.

[72] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

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

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

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

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

[77] H. Koppula and A. Saxena. Anticipatory planning for human-robot teams. In *ISER*, 2014.

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

[79] Sanjiv Kumar and Martial Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *ICCV*, 2003.

[80] J. Lafferty, A. McCallum, and F.C.N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

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

[82] N. D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *NIPS*, 2004.

[83] N. D Lawrence and J. Quiñonero-Candela. Local distance preservation in the gp-lvm through back constraints. In *ICML*, pages 513–520. ACM, 2006.

[84] Q. V. Le, D. Kamm, A. Kara, and A. Y. Ng. Learning to grasp objects with multiple contact points. In *ICRA*, 2010.

[85] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010.

[86] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 17–32, 2004.

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

[88] Congcong Li, Adarsh Kowdle, Ashutosh Saxena, and Tsuhan Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. In *NIPS*, 2010.

[89] Congcong Li, Adarsh Kowdle, Ashutosh Saxena, and Tsuhan Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. In *Neural Information Processing Systems (NIPS)*, 2010.

[90] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *CVPR*, 2008.

[91] H. Liu and L. Yu. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 17(4):491–502, 2005.

[92] A. Lodi, S. Martello, and D. Vigo. Heuristic algorithms for the three-dimensional bin packing problem. *European Journal of Operational Research*, 141(2):410–420, 2002.

[93] T. Lozano-Pérez, J.L. Jones, E. Mazer, and P.A. O'Donnell. Task-level planning of pick-and-place robot motions. *Computer*, 22(3):21–29, 2002.

[94] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *ICRA*, pages 2308–2315. IEEE, 2010.

[95] Andrew T. Miller, Steffen Knoop, Henrik I. Christensen, and Peter K. Allen. Automatic grasp planning using shape primitives. In *ICRA*, pages 1824–1829, 2003.

[96] R.M. Neal. Markov chain sampling methods for dirichlet process mixture models. *J comp graph statistics*, 9(2):249–265, 2000.

[97] V.D. Nguyen. Constructing stable force-closure grasps. In *ACM Fall joint computer conf*, 1986.

[98] J. Nocedal and S. Wright. Numerical optimization, series in operations research and financial engineering. *Springer, New York*, 2006.

[99] M. Novotni and R. Klein. 3d zernike descriptors for content based shape retrieval. In *ACM symp. Solid modeling and applications*, 2003.

[100] D. Pisinger and M. Sigurd. Using decomposition techniques and constraint programming for solving the two-dimensional bin-packing problem. *INFORMS Journal on Computing*, 19(1):36–51, 2007.

[101] J. Ponce, D. Stam, and B. Faverjon. On computing two-finger force-closure grasps of curved 2D objects. *IJRR*, 12(3):263, 1993.

[102] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *NIPS*. Citeseer, 2004.

[103] A. Quattoni, S. Wang, L.P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *PAMI*, 29(10):1848–1852, 2007.

[104] D. Rao, Q.V. Le, T. Phoka, M. Quigley, A. Sudsang, and A.Y. Ng. Grasping Novel Objects with Depth Segmentation. In *IROS*, 2010.

[105] A. Rodriguez, A. Lenkoski, and A. Dobra. Sparse covariance estimation in heterogeneous samples. *Elec. Journal of Stat.*, 5:981–1014, 2011.

[106] C. Rosales, J.M. Porta, and L. Ros. Global optimization of robotic grasps. *RSS*, 2011.

[107] D. Roth and W. Yih. Integer linear programming inference for conditional random fields. In *ICML*, 2005.

[108] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *3DIM*, 2001.

[109] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, 2009.

[110] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[111] R.B. Rusu, Z.C. Marton, N. Blodow, and M. Beetz. Learning informative point classes for the acquisition of object model maps. In *Control, Automation, Robotics and Vision, 2008. ICARCV 2008. 10th International Conference on*, pages 643–650. IEEE, 2008.

[112] A. Safonova, J. K Hodgins, and N. S Pollard. Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Transactions on Graphics (TOG)*, 23(3):514–521, 2004.

[113] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. *IJCV*, 2007.

[114] A. Saxena, S.H. Chung, and A. Ng. Learning depth from single monocular images. In *NIPS 18*, 2005.

[115] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng. Robotic grasping of novel objects. In *NIPS*, 2006.

[116] A. Saxena, J. Driemeyer, and A.Y. Ng. Robotic grasping of novel objects using vision. *IJRR*, 27(2):157, 2008.

[117] A. Saxena, M. Sun, and A.Y. Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE transactions on pattern analysis and machine intelligence (PAMI)*, 2009.

[118] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *NIPS 18*, 2005.

[119] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. 3-d depth reconstruction from a single still image. *International Journal of Computer Vision (IJCV)*, 76(1):53–69, 2008.

[120] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Learning 3-d object orientation from images. In *ICRA*, 2009.

[121] 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.

[122] Ashutosh Saxena, Lawson Wong, and Andrew Y. Ng. Learning grasp strategies with partial shape information. In *AAAI*, 2008.

[123] P. Schnitzspan, S. Roth, and B. Schiele. Automatic discovery of meaningful object parts with latent crfs. In *CVPR*, 2010.

[124] M. Schuster, J. Okerman, H. Nguyen, J. Rehg, and C. Kemp. Perceiving clutter and surfaces for object placement in indoor environments. In *Humanoid Robots*, 2010.

[125] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[126] J. Sturm, K. Konolige, C. Stachniss, and W. Burgard. 3d pose estimation, tracking and model learning of articulated objects from dense depth video using projected texture stereo. In *RSS*, 2010.

[127] Erik B Sudderth, Antonio Torralba, William T Freeman, and Alan S Willsky. Depth from familiar objects: A hierarchical model for 3d scenes. In *CVPR*, 2006.

[128] H. Sugie, Y. Inagaki, S. Ono, H. Aisu, and T. Unemi. Placing objects with multiple mobile robots-mutual help using intention inference. In *ICRA*, 1995.

[129] J. Sung, C. Ponce, B. Selman, and A. Saxena. Human activity detection from RGBD images. In *ICRA*, 2012.

[130] C. Sutton, K. Rohanimanesh, and A. McCallum. Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. In *ICML*, 2004.

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

[132] B Taskar, C Guestrin, and D Koller. Max-margin markov networks. In *NIPS*, 2003.

[133] Y. W. Teh. Dirichlet process. *Encyc. of Mach. Learn.*, 2010.

[134] A. Thomas, V. Ferrar, B. Leibe, T. Tuytelaars, B. Schiel, and L. Van Gool. Towards multi-view object class detection. In *CVPR*, 2006.

[135] A. Torralba, K. Murphy, and W. T. Freeman. Using the forest to see the trees: object recognition in context. *Communications of the ACM, Research Highlights*, 53(3):107–114, 2010.

[136] Marc Toussaint, Nils Plath, Tobias Lang, and Nikolay Jetchev. Integrated motor control, planning, grasping and high-level reasoning in a blocks world using probabilistic inference. *ICRA*, 2010.

[137] R. Urtasun, D. J Fleet, and P. Fua. 3d people tracking with gaussian process dynamical models. In *CVPR*, 2006.

[138] J. Van Gael, Y. W. Teh, and Z. Ghahramani. The infinite factorial hidden markov model. In *NIPS*, 2008.

[139] J M Wang, D J Fleet, and A Hertzmann. Gaussian process dynamical models for human motion. *PAMI*, 30(2):283–298, 2008.

[140] S.B. Wang, A. Quattoni, L.P. Morency, D. Demirdjian, and T. Darrell. Hidden conditional random fields for gesture recognition. In *CVPR*, 2006.

[141] Yang Wang and Greg Mori. Max-margin hidden conditional random fields for human action recognition. In *CVPR*, 2009.

[142] R. Wilcox, S. Nikolaidis, and J. A. Shah. Optimization of temporal dynamics for adaptive human-robot interaction in assembly manufacturing. In *RSS*, 2012.

[143] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. *ICCV*, 2005.

[144] X. Xiong and D. Huber. Using context to create semantic 3d models of indoor environments. In *BMVC*, 2010.

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

[146] C. Yanover, T. Meltzer, and Y. Weiss. Linear programming relaxations and belief propagation–an empirical study. *The Journal of Machine Learning Research*, 7:1887–1907, 2006.

[147] A. Yao, J. Gall, L. V Gool, and R. Urtasun. Learning probabilistic non-linear latent variable models for tracking complex activities. In *NIPS*, 2011.

[148] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A Bagnell, M. Hebert, A. K Dey, and S. Srinivasa. Planning-based prediction for pedestrians. In *IROS*, 2009.