Toward Visual Understanding of Everyday Object

by

Joseph J. Lim

B.A., Computer Science, University of California - Berkeley, 2009
S.M., Electrical Engineering and Computer Science, M.I.T., 2012

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

September 2015

Signature of Author: Signature redacted
Department of Electrical Engineering and Computer Science
June 30, 2015

Certified by: Signature redacted
Antonio Torralba
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Signature redacted
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Committee for Graduate Students

Toward Visual Understanding of Everyday Objects

by Joseph J. Lim

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

The computer vision community has made impressive progress on object recognition using large scale data. However, for any visual system to interact with objects, it needs to understand much more than simply recognizing where the objects are. The goal of my research is to explore and solve object understanding tasks for interaction – finding an object's pose in 3D, understanding its various states and transformations, and interpreting its physical interactions.

In this thesis, I will focus on two specific aspects of this agenda: 3D object pose estimation and object state understanding. Precise pose estimation is a challenging problem. One reason is that an object's appearance inside an image can vary a lot based on different conditions (e.g. location, occlusions, and lighting). I address these issues by utilizing 3D models directly. The goal is to develop a method that can exploit all possible views provided by a 3D model – a single 3D model represents infinitely many 2D views of the same object. I have developed a method that uses the 3D geometry of an object for pose estimation. The method can then also learn additional real-world statistics, such as which poses appear more frequently, which area is more likely to contain an object, and which parts are commonly occluded and discriminative. These methods allow us to localize and estimate the exact pose of objects in natural images.

Finally, I will also describe the work on learning and inferring different states and transformations an object class can undergo. Objects in visual scenes come in a rich variety of transformed states. A few classes of transformation have been heavily studied in computer vision: mostly simple, parametric changes in color and geometry. However, transformations in the physical world occur in many more flavors, and they come with semantic meaning: e.g., bending, folding, aging, etc. Hence, the goal is to learn about an object class, in terms of their states and transformations, using the collection of images from the image search engine.

Thesis Supervisor: Antonio Torralba
Title: Professor of Electrical Engineering and Computer Science
Thesis Committee: Professor Bill Freeman, Professor Jitendra Malik

To my parents and my brother,

and to my wife

Acknowledgments

I would first like to thank my advisor, Professor Antonio Torralba, for his guidance and encouragement throughout my Ph.D. life. His endless passion and enthusiasm toward Computer Vision has been a great source of my motivation throughout my PhD study. His trust, insight, and encouragement always motivated me to step forward one by one. Also, I would like to thank him for his full trust in my research agenda and giving me all the academic freedom.

I would also like to thank to my undergraduate research advisor, who was also on my thesis committee, Professor Jitendra Malik. His great energy and enthusiasm inspired me to start diving into Computer Vision back when I was an undergraduate student at UC Berkeley. His insightful advice and feedback always helped me focusing on the important problems throughout my undergraduate and graduate studies. Also, my another thesis committee member, Professor Bill Freeman, has given me countless number of advice whenever I have academic or career related questions. His dedication to fine understanding has been another inspiration to my research too. Also, my first research advisor, Professor Max Welling, for guiding me when I was a young undergraduate student with little background in research.

Then, I also thank to my collaborators: Ted Adelson, Byoungkwon An, Pablo Arbeláez, Myung Jin Choi, Piotr Dollár, Chunhui Gu, Phillip Isola, Aditya Khosla, Andrew Owens, Hamed Pirsiavash, Ruslan Salakhutdinov, Josh Tenenbaum, Donglai Wei, Jiajun Wu, Tianfan Xue, and Larry Zitnick. They gave a big motivation and inspired me on every problem I worked on. Their intellectual and moral support made my achievement possible.

During my Ph.D. study, I had so much encouragement and collaborative discussions with Phillip Isola, Aditya Khosla, and Andrew Owens. This has ended in several projects and part of my dissertation. I feel grateful to have such talented people as my collaborators and friends.

Also, I owe my appreciation to many MIT visionaries: Yusuf Aytar, Katie Bouman, Adrian Dalca, Abe Davis, Jason Chang, Taeg Sang Cho, George Chen, Fredo Durand, Zoya Gavrilov, Roger Grosse, Ce Liu, Tomasz Malisiewicz, Hossein Mobahi, Aude Oliva, Adria Recasens, Michael Rubinstein, Yichang Shih, Julian Straub, Ramesh Sridharan, Carl Vondrick, Neal Wadhwa, Jianxiong Xiao, Jenny Yuen, and Bolei Zhou. They helped me going through many research obstacles by brain-storming and working together all day and night. I also want to thank to Alex Berg, Fredo Durand, Alyosha Efros, Abhinav Gupta, James Hays, Fei-Fei Li, Subhransu Maji, Jianxiong Xiao, and Larry Zitnick as well for sharing their advice and experience as researchers in the field. This helped me understanding how to contribute and be part of the community.

The National Science Foundation funded my research between 2009-2014 by the NSF Graduate Research Fellowship. I would like to thank to them for awarding the fellowship. Also, I learned so much from internship experiences at Microsoft Research and Adobe; and I would like to thank them for such opportunities.

Finally, I would like to thank my parents, Bonguk and Kyungsun, and my brother, Yongwhan, for shaping myself as who I am and their unconditional love. Also, I thank my wife, Eunice, for her strong support and encouragement. Without them, I would never have been able to fall in love with research or continue to do what I love to do.

Table of Contents

List of Figures

Chapter 1

Introduction

Imagine asking a robot to cook a meal for you. The robot's visual system needs to understand much more than simply recognizing where the ingredients are in the kitchen. It needs to know the exact 3D location of the vegetables and the stove, ensure that the vegetables are fresh enough to cook, know where on the stove to put the pan, and understand how the appearance of the vegetables will change as they get cooked.

Many studies in object recognition have focused on finding the 2D position of objects inside an image [5, 12, 20, 28, 30, 32, 39, 56, 64, 90, 92, 93, 99, 100, 102, 112]. However, there are many other object properties we need to understand. Especially, I am interested in fine-grained understanding and modeling of objects focused around interaction. What would we need in order to build a robot system to use tools, an augmented reality system to guide human how to use tools (e.g. showing a guiding line for an iron), or even video and image analyzing tools?

The key component for such object understanding is to infer various properties of objects related to *where*, *what*, *how*, and *why*. For example, if an agent has to cook: what is the geometry of the stove? How do stoves work? Are the meat and vegetables well cooked? This involves estimating an object's pose [2, 42, 63, 65, 80, 104, 105], state [47], affordances [35, 54, 108, 115], and relationships [9, 16, 26, 50, 62, 113] among others (see Figure 1.1). In the following, I will describe advantages and challenges of understanding each property.

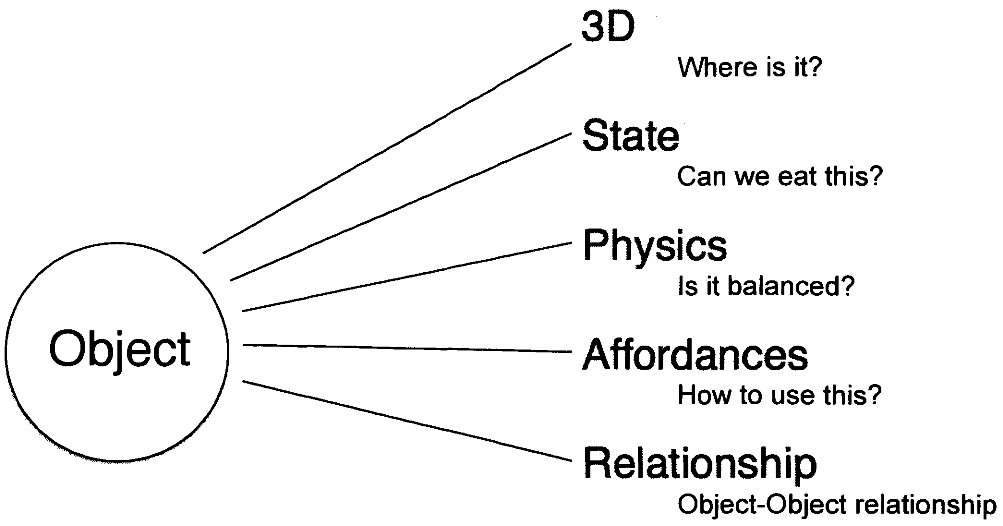


Figure 1.1: What do we need to understand about objects in order to interact with them?

A first natural question for understanding objects is "where is the object?" However, identifying only its location is not enough; we also need to understand its precise 3D pose in order to manipulate it. However, precise pose estimation is a hard problem. One reason is that how an object appears inside a photograph can vary a lot based on different conditions. For example, an object's appearance can change radically based on its location, occlusions, and lighting. Also, many objects have similar but unique shapes, and also they can have deformable parts. In addition to these challenges, 3D perception is an under-constrained problem if we focus on a single image. Hence, we need a robust 3D representation and a strong prior about objects, a learning algorithm that can cover object shape and appearance variations, and an inference engine that can efficiently search objects in photographs.

Modeling shapes of instances and locating them is an important problem, yet this does not quite get us to full object understanding. We need more information to interact with and manipulate objects. For example, we would like to identify whether the vegetables are *raw* or already *cooked.* We also want to know how to *chop* them. This requires understanding the object's physical state and dynamics. Objects go through many different transformations beyond simple changes in color or shape. For example, a banana ripens, fish gets cooked, and a window can break. The challenge is to understand such properties, even for objects seen for the first time. How can we tell whether a "tomato" is fresh or moldy when we have never seen it before? Some of important tasks are identifying the current state of an object (e.g. the tomato is moldy), discovering transformations depicted in a collection of photos of a certain type of object (e.g. peeling and slicing for apple), and ordering images according to each discovered transformation (e.g. vegetables ordered from raw to cooked).

Also, there are several other important object properties, which will not be addressed in this thesis. We need to understand physical behavior of an object (e.g. will the grill be stable when it is hit by another object?), its affordances (e.g. which object(s) do we need to use to boil water?), and its relationships with other objects (e.g. which object(s) do we place on a table?). Finally, how can we work up from these to a holistic understanding of all the world's objects? This is a very challenging task considering the number of objects in the world and their variations of properties (e.g. location, pose, state, and function). One key component is learning to generalize across all different objects and properties. We can build a web of relationships that link object instances into broader physical, functional, and semantic classes. Also, we need to be able to model inter- and intra-class variations too. Combining these at all scales will enable more powerful generalization, which is important for a learning process because we cannot train a machine for an exhaustive set of scenarios. This framework will enable identifying fine-details of objects.

My work in this thesis focuses on two of the previously mentioned problems (i.e. building a 3D object representation, and interpreting object state and dynamics). I describe how to develop computer vision systems to solve such object understanding

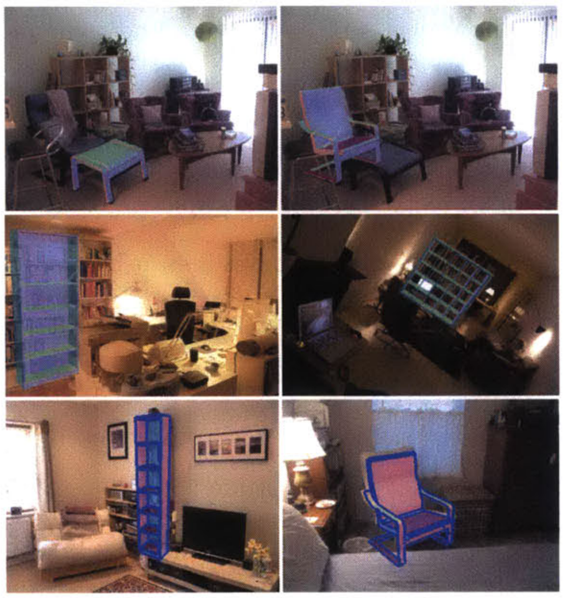


Figure 1.2: Methods in Chapter 2 and Chapter 3 can estimate object poses accurately. Estimated poses are superimposed on top of images.

tasks by developing learning methods that can leverage different types of data. Each of these is an important step toward creating a visual system that can interact with objects and support human activity. In the following, I describe the outline of this thesis, my contributions, and related work.

1.1 Thesis Outline

Chapter 2 - 3D Fine-pose estimation This chapter outlines my first method for the fine-pose estimation problem ("where is the object?", see Figure 1.2). It describes the problem formulation, the dataset I created, and also the approach I took. The method in this chapter uses 3D models to take an advantage of having and modeling the object's geometry explicitly. The goal of this chapter is to develop a method that can compactly represent all possible views of the object by exploiting a 3D model – a single 3D model represents infinitely many 2D views of the same object [63].

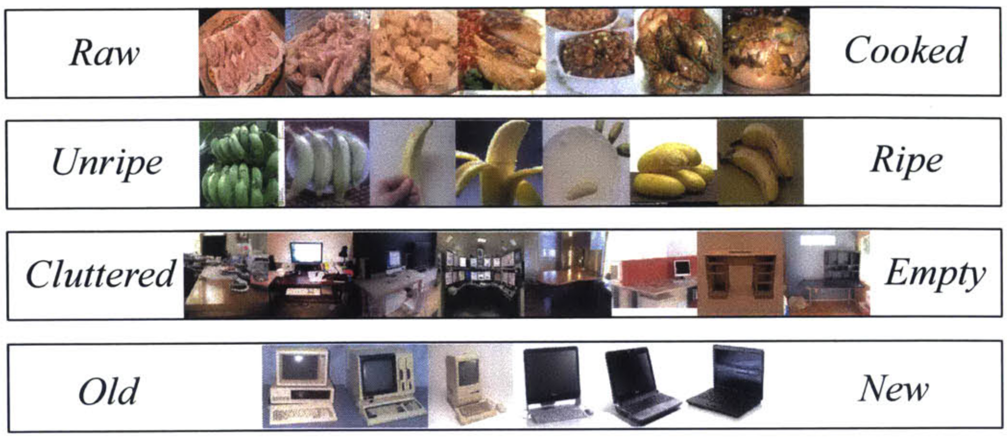


Figure 1.3: The method in Chapter 4 can infer an object's state and its dynamics varying from one state to another.

Chapter 3 - Understanding the real world statistics for 3D pose estimation Despite providing explicit geometric information (in Chapter 2), 3D models typically lack fine-details such as texture, lighting, and exact shape, and any contextual information such as common occlusions and part discriminativeness. This chapter addresses how to learn these real world statistics (e.g. which parts are commonly occluded or discriminative) from a small number of annotated photographs [65].

Chapter 4 - Understanding the physical states of an object This chapter defines the problem and builds methods on understanding states and transformations of objects (see Figure 1.3). It describes an algorithm that can generalize and identify such states and dynamics of objects [47] from a collection of images. This system utilizes a deep learning architecture, which has recently gained much popularity, and a collection of images.

■ 1.2 Contributions

There are three primary contributions in this thesis:

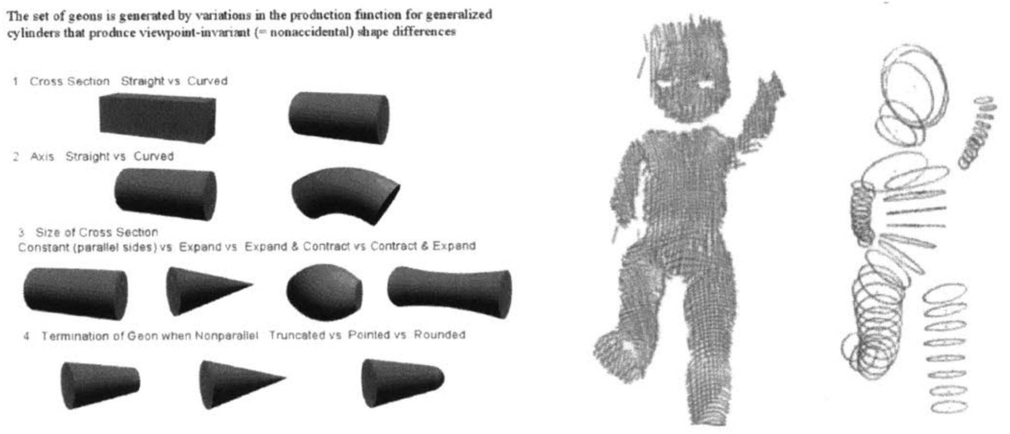
1. **Proposing the fine-pose estimation problem:** We propose a pose estimation problem that aims for a very fine-quality estimation. It has challenges of category level detection while allowing for an accurate representation of the object pose in the image.

2. **Utilizing 3D models for pose estimation:** Learning 3D information from 2D photographs is difficult. We propose a novel solution that unifies contributions from the 1970s with recent advances in object detection. We use 3D models to learn the geometry. While 3D models provide explicit geometric information, there is a modality difference issue between real images and 3D models (discussed in Appendix A). Our methods are developed to overcome such an issue while taking advantage of 3D models.

3. **Parsing object states and transformations:** We propose and solve a parsing problem that can interpret the states and transformations an object class can go through, by looking at the entire collection of images. Our goal is to learn a model that can generalize across object classes: states and transformations learned on one set of objects are used to interpret a similar, or even an entirely new object class. This is one of the first works that proposed the problem of identifying object states and dynamics with large-scale data (see Figure 1.3).

1.3 Related Work

The field of computer vision has made a tremendous progress on object classification and detection [5, 12, 20, 28, 30, 32, 39, 56, 64, 90, 92, 93, 99, 100, 102, 112]. The focus of this thesis is to discuss and solve other problems that need to be solved in order to interact with objects. In the following, we discuss the most relevant papers to the broad topics that will be covered in this thesis.


The set of geons is generated by variations in the production function for generalized cylinders that produce viewpoint-invariant (= nonaccidental) shape differences
1 Cross Section Straight vs Curved
2 Axis Straight vs Curved
3 Size of Cross Section
Constant (parallel sides) vs Expand vs Expand & Contract vs Contract & Expand
4 Termination of Geon when Nonparallel Truncated vs Pointed vs Rounded

(a) Geon (b) Generalized Cylinder

Figure 1.4: Early work represented an object with their explicit 3D geometry. Both methods in this thesis also use 3D CAD models explicitly for pose estimation.

■ 1.3.1 3D Understanding

Detecting objects from single images and estimating their pose in 3D was a popular topic in the early days of computer vision [73]. Since then, the major focus of the field has been on 2D object classification and detection until recently, but 3D pose estimation has re-gained much interest in the last decade [5, 36, 42, 44, 52, 71, 86, 88, 94, 97, 99, 107, 110, 116].

The traditional approaches [67, 73] were dominated by using accurate geometric representations of 3D objects with an emphasis on viewpoint invariance (shown in Figure 1.4). Objects would appear in almost any pose and orientation in the image. Most of the approaches were designed to work at instance level detection as it was expected that the 3D model would accurately fit the model in the image. Instance level detection regained interest with the introduction of new invariant local descriptors that dramatically improved the detection of interest points [83]. Those models assumed knowledge about geometry and appearance of the instance. Having access to accurate knowledge about those two aspects allowed precise detections and pose estimations of the object on images even in the presence of clutter and occlusions.

In the last few years, various works have tackled the problem of category-level pose estimation [24, 43, 107]. Many of them [23, 24, 36, 41, 43, 79, 105, 107, 109] extended 2D constellation models to include 3D information in the object representation. In other words, they make assumptions that a major structure is shared between objects, e.g. cuboid, and hence can use this set of relatively few models to infer the pose of an object. This requires the models to be generic and flexible, as they have to deal with all the variations in shape and appearance of the instances that belong to the same category. Therefore, the shape representations used for those models are coarse (e.g., modeled as the constellation of a few 3D parts or planes) and the expected output is at best an approximate fitting of the 3D model to the image.

In Chapter 2 and Chapter 3, we introduce a detection task that is in the intersection of the traditional approach and the recent approach. Our goal is more generic than detecting instances, but we assume richer models than the ones typically used in category level detection. In particular, we assume that accurate CAD models of the objects are available. Hence, there is little variation on the shape of the instances that form one category. We focus on detection and pose estimation of objects *in the wild* given their 3D CAD models.

Potentially, both methods in this thesis can be combined with other 3D parsing work. The recent availability of affordable depth sensors has also led to an increase in the use of RGB-D data [3, 24, 43, 58, 74] to extract various types of 3D information from scenes such as depth, object pose, and intrinsic images. There have also been several works [35, 40, 45, 46, 85, 110] addressing the problem of estimating the structure of a scene from a single image. The goal of these works is to recover a scene reconstruction, usually based on vanishing point estimations. Ultimately, these methods can be combined with the methods in this thesis: we can utilize the depth information to enhance the match quality and/or prune the search space by the scene prior.

■ 1.3.2 State Understanding

While there has been little focus on understanding object states and transformations, they have been addressed as part of "attributes" in the computer vision community [4, 19, 33, 34, 76–78, 91]. Most of this work has dealt with one image at a time and has not extensively catalogued the state variations that occur in an entire image class. We define the state understanding problem as to focus on studying objects' states and their transformations from one to another, and also understanding an object as whole.

Understanding image collections is a relatively unexplored task, although there is growing interest in this area. Several methods attempt to represent the continuous variation in an image class using subspaces [11, 72] or manifolds [49]. Photo collections have also been mined for storylines [53] as well as spatial and temporal trends [59], and systems have been proposed for more general knowledge discovery from big visual data [4, 7, 68, 91].

In Chapter 4, we focus on understanding variation in image *collections*. In addition, we go beyond previous attributes work by linking up states into pairs that define a transformation: e.g., *raw↔cooked, rough↔smooth, and deflated↔inflated.* We investigate discrete, nameable transformations, like *crinkling*, rather than working in a hard-to-interpret parameter space. At the end, we explain image collections both in terms of their states (unary states) and transformations (antonymic state pairs). We also show how state pairs can be used to extract a continuum of images depicting the full range of the transformation (Figure 4.1 bottom-left). Our method differs from previous work in that we focus on *physical state transformations*, and in addition to discovering states we also study state pairs that define a transformation.

■ 1.4 Previously Published Work

This thesis is largely based on work that appeared in the 2013 IEEE International Conference on Computer Vision (ICCV) [63], 2014 European Conference on Computer Vision (ECCV) [65], and 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [47]. All the accompanying materials referenced here, including software and data, are available for the research community through the thesis web page: `http://people.csail.mit.edu/lim/PhDThesis`.

Chapter 2

Parsing IKEA objects

Just as a thought experiment imagine that we want to detect and fit 3D models of IKEA furniture in the images as shown in Figure 1. We can find surprisingly accurate 3D models of IKEA furniture, such as *billy* bookcase and *ektorp* sofa, created by IKEA fans from Google 3D Warehouse and other publicly available databases. Therefore, detecting those models in images could seem to be a very similar task to an instance detection problem in which we have training images of the exact instance that we want to detect. But, it is not exactly like detecting instances. In the case of typical 3D models (including IKEA models from Google Warehouse), the exact appearance of each piece is not available, only the 3D shape is. Moreover, appearance in real images will vary significantly due to a number of factors. For instance, IKEA furniture might appear with different colors and textures, and with geometric deformations (as people building them might not do it perfectly) and occlusions (e.g., a chair might have a cushion on top of it).

The problem that we introduce in this chapter and Chapter 3 is detecting and accurately fitting exact 3D models of objects to real images, as shown in Figure 2.1. While fine pose estimation for simple 3D shapes has been studied since the early days of computer vision [67], estimating the fine pose of articulate 3D objects has received little attention. Given the recent success of the computer vision community in addressing object detection [20, 27] and 3D scene understanding [10, 25, 36, 48, 87], fine pose esti-

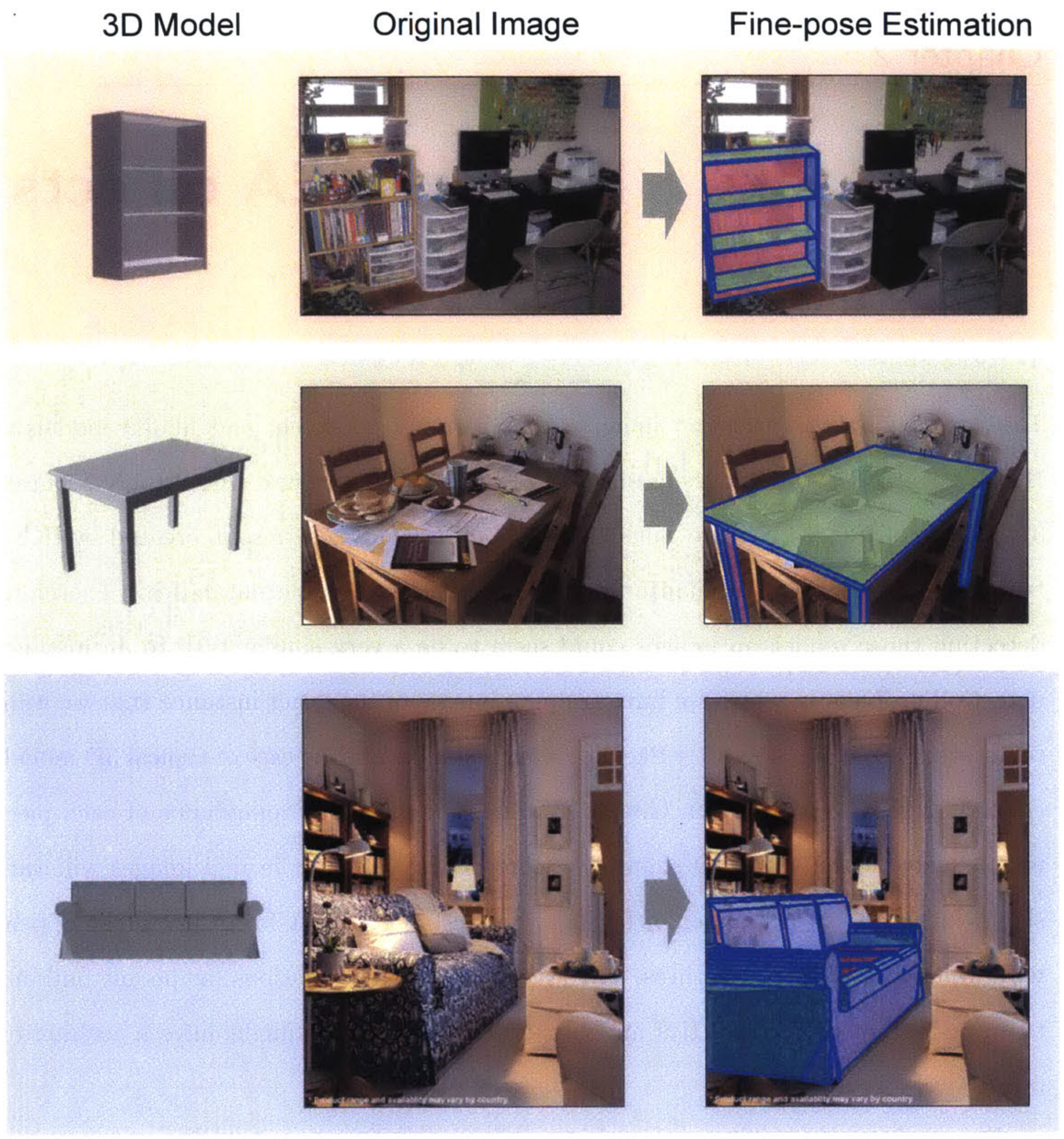


Figure 2.1: The goal in this chapter is to detect and estimate the fine-pose of an object in the image given an exact 3D model.

mation is becoming more and more approachable. In this work, we tackle the problem of instance-level fine pose estimation given only a single-view image. Specifically, we want to estimate the fine pose of objects in images, given a set of exact 3D models.

We restrict our attention to instance-level detection as it provides several advantages over category-level detection. For example, we can get pixel-perfect alignment with per-instance exact 3D models. This would not be possible if we instead used a single generic model for a category, as categories contain large variation in shape across instances.

Recently, there has been an increasing interest in utilizing 3D models for object detection and pose estimation [23, 41, 63, 94, 116]. Despite this, using explicit 3D models for fine pose estimation presents various challenges. First, the space of possible poses of an object is extremely large. Even if we assume that the intrinsic parameters of the camera are known, we still have to estimate at least 6 parameters (translation and rotation) to determine the exact pose of the model. This is *three* more parameters than the typical 2D object detection problem (x, y, and *scale*), which already has a search space of millions of possible bounding boxes. Second, there is the issue of domain adaptation. While rendered images of 3D models look similar to real objects, there are some key differences that make it difficult to use the same set of features as we do for real images. For example, rendered images are often texture-less and do not contain occluding objects or surfaces. This is more thoroughly analyzed in Appendix A.

Our contributions are three folds: (1) Proposing a detection problem that has some of the challenges of category level detection while allowing for an accurate representation of the object pose in the image. This problem will motivate the development of better 3D object models and the algorithms needed to find them in images. (2) We propose a novel solution that unifies contributions from the 1970s with recent advances in object detection. (3) And we introduce a new dataset of 3D IKEA models obtained from Google Warehouse and real images containing instances of IKEA furniture and annotated with ground truth pose.

2.1 Methods

We now propose the framework that can detect objects and estimate their poses simultaneously by matching to one of the 3D models in our database.

2.1.1 Our Model

The core of our algorithm is to define the final score based on multiple sources of information such as local correspondence, geometric distance, and global alignment. Training with mltiple sources was a necessary, because training object detectors [20] with rendered images can suffer from the modality difference as I analyze in Appendix A. The basic idea is to use local shape and geometric information from 3D models and global information from real images.

Suppose we are given the image I containing an object for which we want to estimate the projection matrix P with 9 degrees of freedom[1]. We define our cost function S with three terms as follows:

$$S(P, c) = L(P, c) + w^D D(P, c) + w^G G(P) \tag{2.1}$$

where c refers to the set of correspondences, L measures error in local correspondences between the 3D model and 2D image, D measures geometric distance between correspondences in 3D, and G measures the global dissimilarity in 2D.

This scoring function S measures how well each pose candidate P aligns to the given image I. Note that we designed our model to be linear in weight vectors of w^D and w^G. We will use this linearity later to learn our discriminative classifier.

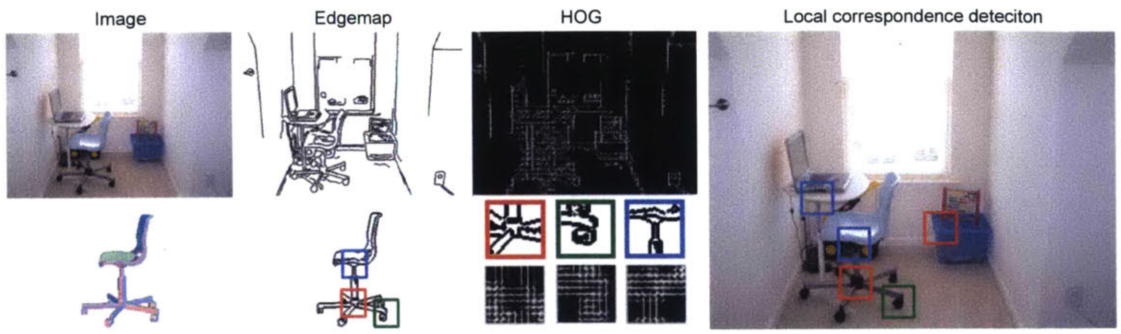


Figure 2.2: **Local correspondence:** for each 3D interest point $\mathbf{X}_i$ (red, green, and blue), we train an LDA patch detector on an edgemap and use its response as part of our cost function. We compute HOG on edgemaps to ensure a real image and our model share the modality.

■ 2.1.2 Local correspondence error

The goal of the first term L is to measure the local correspondences. Given a projection P, we find the local shape-based matching score between the rendered image of the CAD model and the 2D image. Figure 2.2 illustrates this step.

Because our CAD model contains only shape information, a simple distance measure on standard local descriptors fails to match the two images. In order to overcome this modality difference, we compute HOG on the edgemap of both images since it is more robust to appearance change but still sensitive to the change in shape. Moreover, we see better performance by training an LDA classifier to discriminate each keypoint from the rest of the points. This is equivalent to a dot product between descriptors in the whitened space. One advantage of using LDA is its high training speed compared to other previous HOG-template based approaches.

[1]We assume the image is captured by a regular pinhole camera and is not cropped, so the the principal point is in the middle of image.

More formally, our correspondence error is measured by

$$L(P,c) = \sum_i H(\beta_i^T \Phi(\mathbf{x}_{c_i}) - \alpha) \tag{2.2}$$

$$\beta_i = \Sigma^{-1}(\Phi(P(\mathbf{X}_i)) - \mu) \tag{2.3}$$

where β_i is the weight learned using LDA based on the covariance Σ and mean μ obtained from a large external dataset [38], $\Phi(\cdot)$ is a HOG computed on a 20×20 pixel edgemap patch of a given point, $\mathbf{x}_{c_i}$ is the 2D corresponding point of 3D point $\mathbf{X}_i$, $P(\cdot)$ projects a 3D point to 2D coordinate given pose P, and lastly $H(x-\alpha)$ binarizes x to 0 if $x \geq \alpha$ or to ∞ otherwise.

2.1.3 Geometric distance between correspondences

Given a proposed set of correspondences c between the CAD model and the image, it is also necessary to measure if c is a geometrically acceptable pose based on the 3D model. For the error measure, we use euclidean distance between the projection of $\mathbf{x}_i$ and its corresponding 2D point $\mathbf{x}_{c_i}$; as well as the line distance defined in [67] between the 3D line l and its corresponding 2D line c_l.

$$D(P,c) = \sum_{i \in V} ||P(\mathbf{X}_i) - \mathbf{x}_{c_i}||^2 + \sum_{l \in L} \left|\left| \begin{bmatrix} \cos\phi_{c_l}, \sin\phi_{c_l}, -\rho_{c_l} \end{bmatrix} \begin{bmatrix} P(\mathbf{X}_{l_1}) & P(\mathbf{X}_{l_2}) \\ 1 & 1 \end{bmatrix} \right|\right|^2 \tag{2.4}$$

where $\mathbf{X}_{l_1}$ and $\mathbf{X}_{l_2}$ are end points of line l, and ϕ_{c_l} and ρ_{c_l} are polar coordinate parameters of line c_l.

The first term measures a pairwise distance between a 3D interest point $\mathbf{X}_i$ and its 2D corresponding point $\mathbf{x}_{c_i}$. The second term measures the alignment error between a 3D interest line l and one of its corresponding 2D lines c_l [67]. We use the Hough transform on edges to extract 2D lines from I. Note that this cost does not penalize displacement along the line. This is important because the end points of detected lines on I are not reliable due to occlusion and noise in the image.

■ 2.1.4 Global dissimilarity

One key improvement of our work compared to traditional works on pose estimation using a 3D model is how we measure the global alignment. We use recently developed features to measure the global alignment between our proposal and the image:

$$G(P) = [f_{HOG}, f_{region}, f_{edge}, f_{corr}, f_{texture}], \tag{2.5}$$

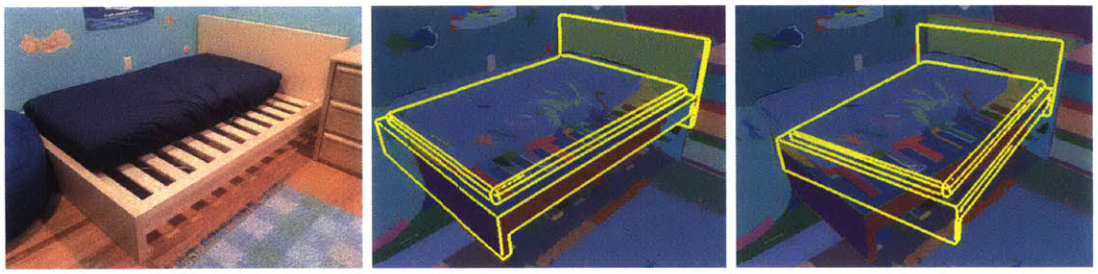
The description of each feature follows:

HOG-based: While D and L from Eq 2.1 are designed to capture fine local alignments, the local points are sparse and hence it is yet missing an object-scale alignment. In order to capture edge alignment per orientation, we compute a fine HOG descriptor (2x2 per cell) of edgemaps of I and the rendered image of pose P. We use a similarity measure based on cosine similarity between vectors .

$$f_{HOG} = \left[\frac{\Phi(I)^T \Phi(P)}{\|\Phi(P)\|^2}, \frac{\Phi(I)^T \Phi(P)}{\|\Phi(I)M\|^2}, \frac{\Phi(I)^T \Phi(P)}{\|\Phi(I)M\| \|\Phi(P)\|} \right], \tag{2.6}$$

where $\Phi(\cdot)$ is a HOG descriptor and M is a mask matrix for counting how many pixels of P fall into each cell. We multiply $\phi(I)$ by M to normalize only within the proposed area (by P) without being affected by other area.

Regions: We add another alignment feature based on super-pixels. If our proposal P is a reasonable candidate, super-pixels should not cross over the object boundary, except in heavily occluded areas. To measure this spill over, we first extract super-pixels, R_I, from I using [22] and compute a ratio between an area of a proposed pose, $|R_P|$, and an area of regions that has some overlap with R_P. This ratio will control the spillover effect as shown in Figure 2.3.



(a) Original image (b) Candidate 1 (c) Candidate 2

Figure 2.3: **Region feature:** One feature to measure a fine alignment is the ratio between areas of a proposed pose (yellow) and regions overlapped with the proposed pose. (a) is an original image I, and (b) shows an example to encourage, while (c) shows an example to penalize.

$$f_{region} = \left[\frac{\sum_{|R_P \cap R_I| > 0.1|R_P|} |R_P|}{|R_I|} \right] \tag{2.7}$$

Texture boundary: The goal of this feature is to capture appearance by measuring how well our proposed pose separates object boundary. In other words, we would like to measure the alignment between the boundary of our proposed pose P and the texture boundary of I. For this purpose, we use a well-known texture classification feature, Local Binary Pattern (LBP) [75].

We compute histograms of LBP on inner boundary and outer boundary of proposed pose P. We define an inner boundary region by extruding the proposed object mask followed by subtracting the mask, and an outer boundary by diluting the proposed object mask. Essentially, the histograms will encode the texture patterns of near-inner/outer pixels along the proposed pose P's boundary. Hence, a large change in these two histograms indicates a large texture pattern change, and it is ideal if the LBP histogram difference along the proposed pose's boundary is large. This feature will discourage the object boundary aligning with contours with small texture change (such as contours within an object or contours due to illumination).

Edges: We extract edges [64] from the image to measure their alignment with the edgemap of estimated pose. We used a modified Chamfer distance from Satkin, et. al. [85].

$$f_{edge} = \left[\frac{1}{|R|}\sum_{a\in R}\min(\min_{b\in I}\|a-b\|,\tau), \frac{1}{|I|}\sum_{b\in I}\min(\min_{a\in R}\|b-a\|,\tau)\right] \tag{2.8}$$

where we use $\tau \in \{10, 25, 50, \infty\}$ to control the influence of outlier edges.

Number of correspondences: f_{corr} is a binary vector, where the i'th dimension indicates if there are more than i good correspondences between the 3D model and the 2D image under pose P. Good correspondences are the ones with local correspondence error (in Eq 2.2) below a threshold.

■ 2.1.5 Optimization & Learning

Our cost function $S(P, c)$ from Eq 2.1 with $G(P)$ is a non-convex function and is not easy to solve directly. Hence, we first simplify the optimization by quantizing the space of solutions. Because our $L(P, c)$ is ∞ if any local correspondence score is below the threshold, we first find all sets of correspondences for which all local correspondences are above the threshold. Then, we find the pose P that minimizes $L(P, c) + w^D(P, c)$. Finally, we optimize the cost function in the discrete space by evaluating all candidates.

```
For each initial seed pose,
while less than $n$ different candidates found do
  Choose a random 3D interest point and its 2D correspondence (this determines a displacement)
  for i = 1 to 5 do
    Choose a random local correspondence agreeing with correspondences selected (over $i - 1$ iterations)
  end for
  Estimate parameters by solving least squares
  Find all correspondences agreeing with this solution
  Estimate parameters using all correspondences
end while
```

Algorithm 1: Pose set $\{P\}$ search

We use RANSAC in populating a set of candidates by optimizing $L(P, c)$. Our RANSAC procedure is shown in Alg 1. We then minimize $L(P, c) + w^D D(P, c)$ by estimating pose P for each found correspondence c. Given a set of correspondences c, we estimate pose P using the Levenberg-Marquardt algorithm minimizing $D(P, c)$.

We again leverage the discretized space from Alg 1 in order to learn weights for Eq 2.1. For the subset of $\{(P, c)\}$ in the training set, we extract the geometric distance and global alignment features, $\left[D(P, c), G(P)\right]$, and the binary labels based on the distance from the ground truth (defined in Eq 2.9) . Then, we learn the weights using a linear SVM classifier.

■ 2.2 Evaluation

■ 2.2.1 Dataset

In order to develop and evaluate fine pose estimation based on 3D models, we created a new dataset of images and 3D models representing typical indoor scenes. We explicitly collected IKEA 3D models from Google 3D Warehouse, and images from Flickr. The key difference of this dataset from previous works [85, 107] is that we align exact 3D models with each image, whereas others provided coarse pose information without using

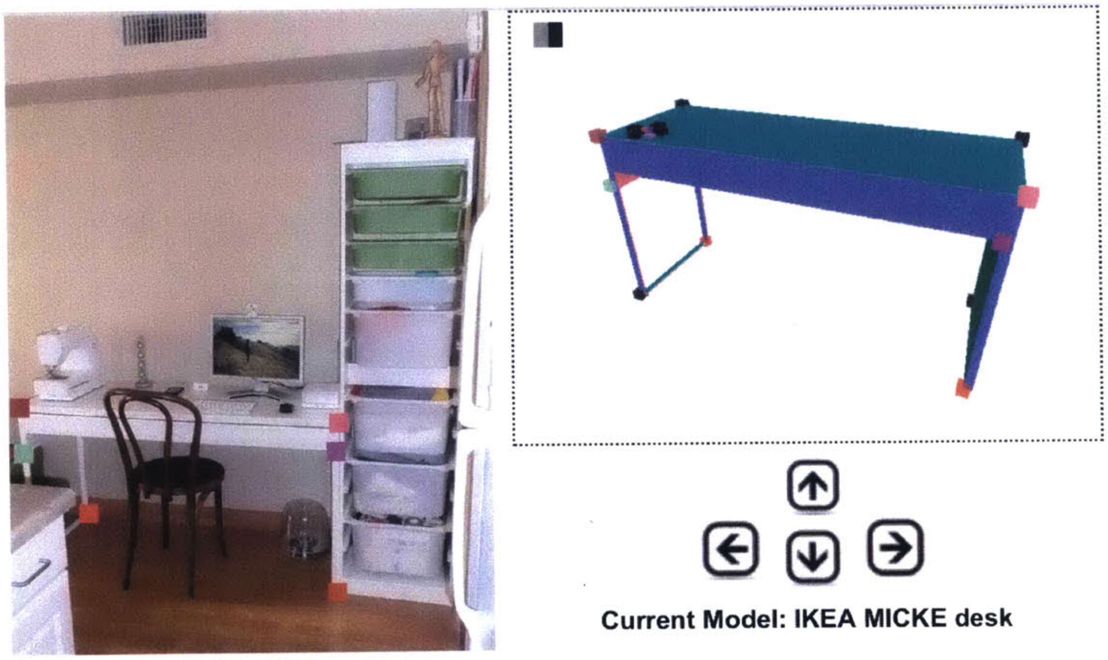
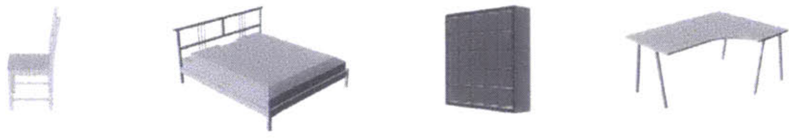
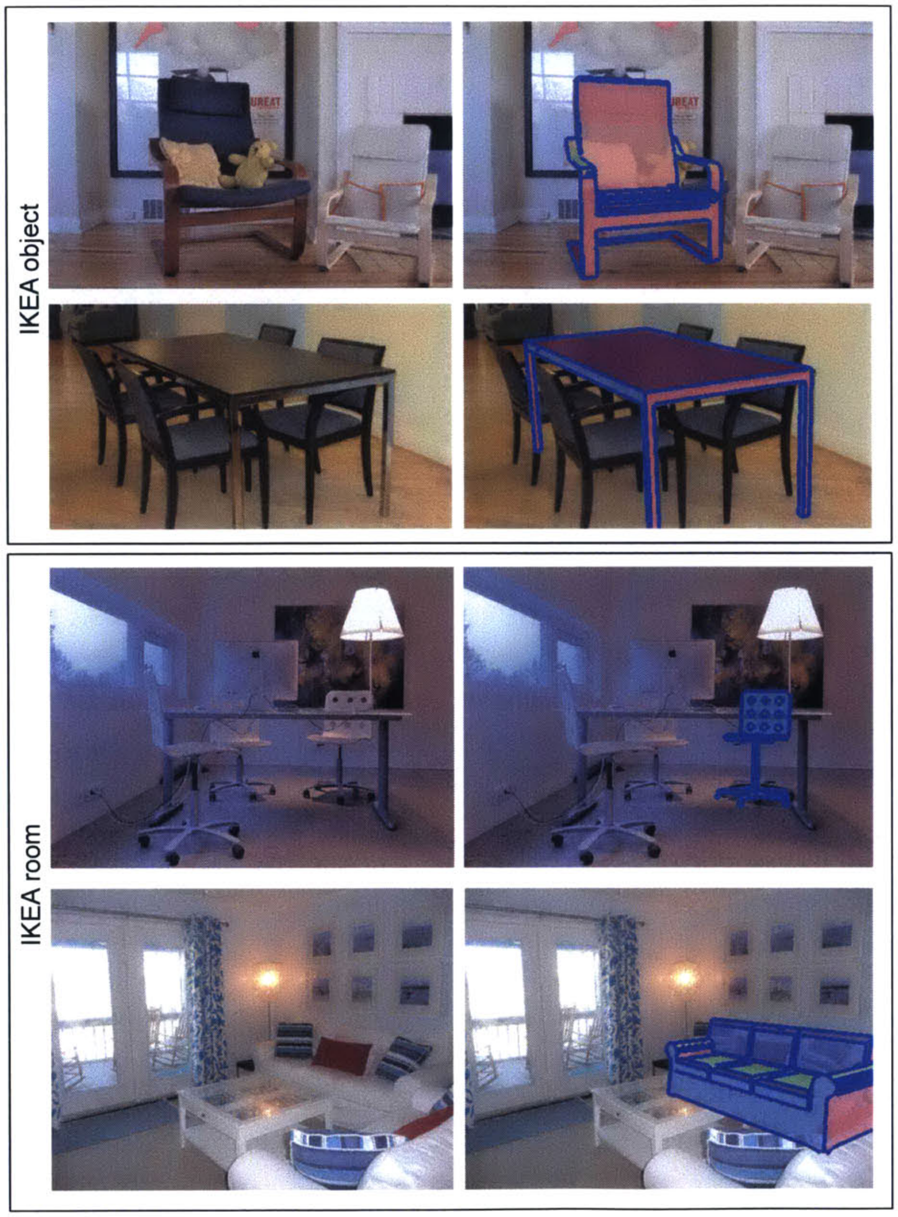


Figure 2.4: **Labeling tool:** our labeling tool enables users to browse through 3D models and label point correspondences to an image. The tool provides a feedback by rendering estimated pose on the image and an user can edit more correspondences.



(a) 3D models



(b) Aligned Images

Figure 2.5: **Dataset:** (a) examples of 3D models we collected from Google Warehouse, and (b) ground truth images where objects are aligned with 3D models using our labeling tool. For clarity, we show only one object per image when there are multiple objects.

exact 3D models.

For our dataset, we provide 800 images and 225 3D models. All 800 images are fully annotated with 90 different models. Also, we separate the data into two different splits: **IKEAobject** and **IKEAroom**. IKEAobject is the split where 300 images are queried by individual object name (e.g. *ikea chair poang* and *ikea sofa ektorp*). Hence, it tends to contain only a few objects at relatively large scales. IKEAroom is the split where 500 images are queried by *ikea room* and *ikea home*; and contains more complex scene where multiple objects appear at a smaller scale. Figure 4.3ab show examples of our 3D models and annotated images.

For alignment, we created an online tool that allows a user to browse through models and label point correspondences (usually 5 are sufficient), and check the model's estimated pose as the user labels. Given these correspondences, we solve the least square problem of Eq 2.4 using Levenberg-Marquardt. Here, we obtain the full intrinsic/extrinsic parameters except the skewness and principal points. Figure 2.4 shows a screenshot of our tool.

2.2.2 Error Metric

We introduce a new error metric for fine-pose estimation. Intuitively, we use the average 3D distance between all points in the ground truth and the proposal. When the distance is small, this is close to the average error in viewing angle for all points. Formally, given an estimated pose P_e and a ground truth pose P_{gt} of image I, we obtain corresponding 3D points in the camera space. Then, we compute the average pair-wise distance between all corresponding points divided by their distance to the camera. We consider P is correct if this average value is less than a threshold.

$$score(P_e, P_{gt}) = \frac{\sum_{X_i} \|E_e X_i - E_{gt} X_i\|_2}{\sum_{X_i} \|E_{gt} X_i\|_2} \tag{2.9}$$

2.3 Results

2.3.1 Correspondences

First of all, we evaluate our algorithm on finding good correspondences between a 3D model and an image. This is crucial for the rest of our system as each additional poor correspondence grows the search space of RANSAC exponentially.

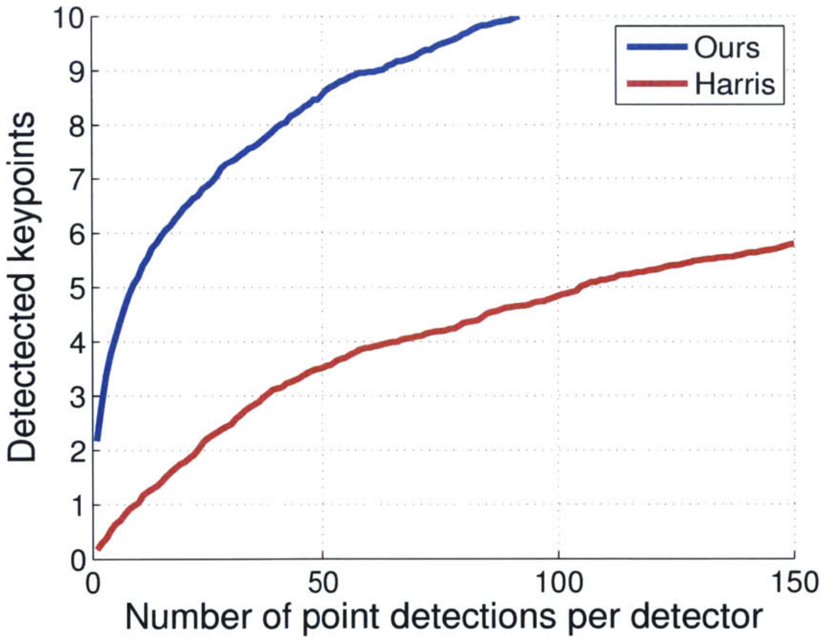


Figure 2.6: **Correspondence evaluation:** we are comparing correspondences between our interest point detector and Harris detector. The minimum number of interest points we need for reliable pose estimation is 5. Ours can recall 5 correct correspondences by considering only the top 10 detections per 3D interest point, whereas the Harris detector requires 100 per point. This results in effectively 10^5 times fewer search iterations in RANSAC.

Figure 2.6 shows our evaluation. We compare Harris corner detectors against our detector based on LDA classifiers. On average, to capture 5 correct correspondences with our method, each interest point has to consider only its top 10 matched candidates. In contrast, using the Harris corner detector, one needs to consider about 100 candidates. This helps finding good candidates in the RANSAC algorithm.

■ 2.3.2 Initial Poses

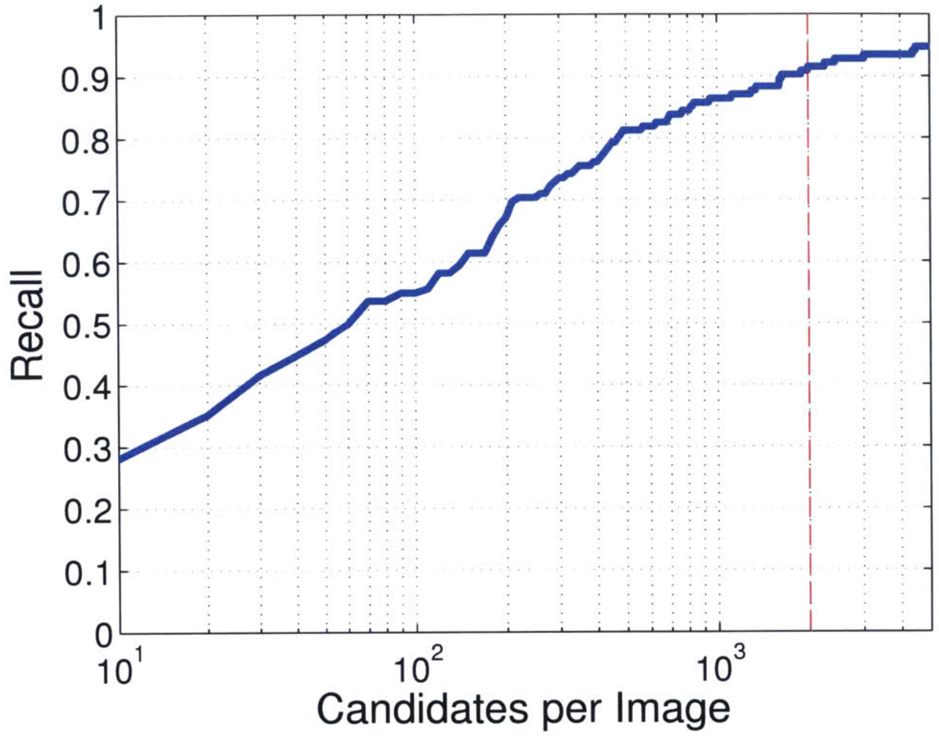


Figure 2.7: **RANSAC evaluation:** we evaluated how many pose candidates per image we need in order to obtain a certain recall. We need only about 2000 candidates in order to have 0.9 recall. This allows us to perform a computationally expensive feature extraction on only a small number of candidates.

In order for us to find the pose P minimizing $S(P, c)$, we need to ensure that our set of poses $\{(P, c)\}$ minimizing $L(P, c) + D(P, c)$ contains at least one correct pose. The recall of our full model is upper bonded by that of this set. Figure 2.7 shows a semi-log plot of recall vs minimum number of top candidates required per image. Considering the top 2000 candidate poses (shown with a red line) from RANSAC, we can obtain 0.9 recall. In other words, the later optimization, where feature extraction is computationally heavy, can run with only the top 2000 candidate poses and still have

	chair *poang*	bookcase *billy1*	sofa *ektorp*	table *lack*	bed *malm1*	bookcase *billy2*
DPM	0.02	0.27	0.08	0.22	0.24	0.35
ELDA	0.29	0.24	0.35	0.14	0.06	0.77
D+L	0.045	0.014	0.071	0.011	0.007	0.069
D+L+HOG	4.48	2.91	0.17	5.56	0.64	9.70
D+L+H+Region	17.16	11.35	2.43	7.24	2.37	17.18
Full	**18.76**	**15.77**	**4.43**	**11.27**	**6.12**	**20.62**

	desk *expedit*	bookcase *billy3*	bed *malm2*	bookcase *billy4*	stool *poang*	mean
DPM	0.09	0.18	0.66	0.11	0.76	0.27
ELDA	0.03	0.20	0.60	0.41	0.13	0.29
D+L	0.008	0.587	0.038	0.264	0.003	0.10
D+L+HOG	0.12	5.05	**15.39**	7.72	0.79	4.78
D+L+H+Region	1.23	7.70	14.24	9.08	3.42	8.49
Full	**6.87**	**7.71**	14.56	**15.09**	**7.20**	**11.67**

Table 2.1: **AP Performances on Pose Estimation:** we evaluate our pose estimation performance at a fine scale in the IKEAobject database. As we introduce more features, the performance significantly improves. Note that DPM and ELDA are trained using rendered images.

	chair *poang*	bookcase *billy1*	sofa *ektorp*	table *lack*	bed *malm1*	bookcase *billy2*
LDA @ 0.5	15.02	5.22	8.91	1.59	15.46	3.08
DPM @ 0.5	**27.46**	**24.28**	**12.14**	10.75	3.41	13.54
Ours @ 0.5	23.17	24.21	6.27	**13.93**	**27.12**	**26.33**
LDA @ 0.8	4.71	0.62	7.49	1.24	3.52	0.11
DPM @ 0.8	7.78	0.56	**10.13**	0.01	1.25	0.00
Ours @ 0.8	**21.74**	**18.55**	5.24	**11.93**	**11.42**	**25.87**

	desk *expedit*	bookcase *billy3*	bed *malm2*	bookcase *billy4*	stool *poang*	mean
LDA @ 0.5	37.62	**34.52**	1.85	**46.92**	0.37	15.51
DPM @ 0.5	**46.13**	34.22	0.95	0.12	0.53	15.78
Ours @ 0.5	18.42	23.84	**22.34**	32.19	**8.16**	**20.54**
LDA @ 0.8	18.76	**21.73**	1.27	7.09	0.17	6.06
DPM @ 0.8	**35.03**	12.73	0.00	0.00	0.44	6.18
Ours @ 0.8	8.99	10.24	**18.14**	**17.92**	**7.38**	**14.31**

Table 2.2: **AP Performances on Detection:** we evaluate our method on detection against [20] and [38] at two different bounding box intersection over union thresholds (0.5 and 0.8) in the IKEAobject database. The gap between our method and [20] becomes significantly larger as we increase the threshold; which suggests that our method is better at fine detection.

a recall 0.9.

■ 2.3.3 Final Pose Estimation

Table 2.1 shows the evaluation of our method with various sets of features as well as two state-of-the-art object detectors: Deformable part models [20] and Exemplar LDA [38] in IKEAobject database.

For both [20] and [38], we trained detectors using the code provided by the authors. The training set contains rendered images (examples are shown in Figure 4.3b) in order to cover various poses (which are not possible to cover with real images). Note the low performances of [20] and [38]. We believe there are two major issues: (1) they are trained with a very large number of mixtures. If there are some mixtures with high false positive rates, then the whole system can break down, and (2) they are trained with rendered images due to their requirement of images for each different mixture. Having a different modality can result in a poor performance.

Our average precision (AP) is computed based on our error metric (Eq. 2.9). We score each pose estimation P as a true detection if its normalized 3D space distance to the ground truth is within the threshold; and for the rest, we precisely follow a standard bounding box criterion [18]. Here, we would like to emphasize that this is a much harder task than a typical bounding box detection task. Hence, low AP in the range of 0-0.02 should not be misleading.

$D + L$, based on RANSAC, has a low performance by itself. Though it is true that the top candidates generally contain a correct pose (as shown in Figure 2.7), it is clear that scores based only on local correspondences (D and L) are not robust to false positives, despite high recall.

We also evaluated two other features based on HOG and Region features. Adding these two features greatly boosts performance. HOG and Region features were added for capturing global shape similarity. Our full method takes about 12 minutes on a single core for each image and model.

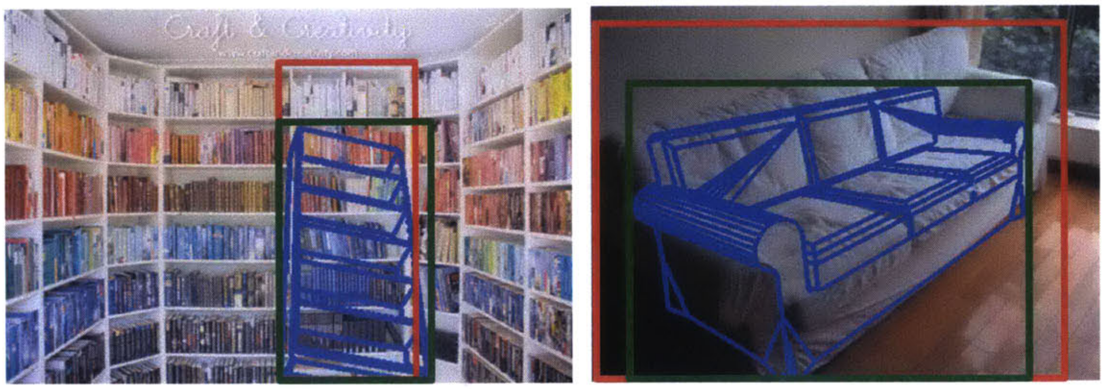


Figure 2.8: **Corner cases:** these two images illustrate cases which are incorrect under our pose estimation criteria, but are still correct under standard bounding box detection criterion.

Also, we compared our method against [20, 38] on a detection task as shown in Table 2.2. For our method, we ran the same full pipeline and extracted bounding boxes from the estimated poses. For the detection task, we trained [20, 38] with real images, because there exist enough real images to train a small number of mixtures. We train/test on 10 different random splits. Because our method is designed for capturing a fine alignment, we measured at two different thresholds on bounding box intersection over union. One is the standard threshold of 0.5 and the other one is a higher threshold of 0.8. As the table shows, our method does not fluctuate much with a threshold change, whereas both [38] and [20] suffer and drop performances significantly. Figure 2.8 shows several detection examples where pose estimation is incorrect, but still bounding box estimation is correct with a threshold of 0.5.

Lastly, Figure 2.9 shows our qualitative results. We first show our pose predictions by drawing blue outlines, and predicted normal directions. To show that our algorithm obtained an accurate pose alignment, we also render different novel views with a simple texture mapping.

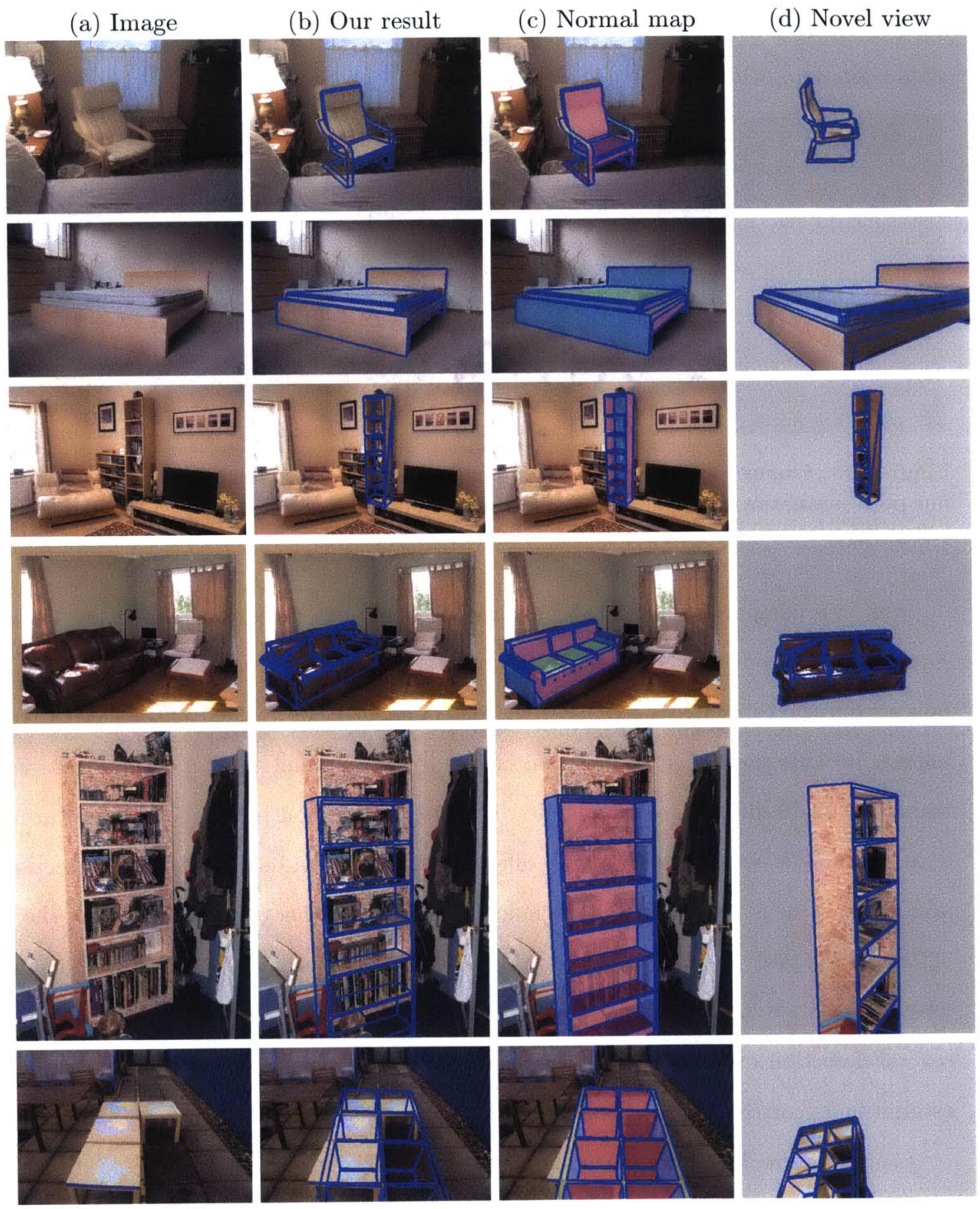


Figure 2.9: **Final results:** we show qualitative results of our method. (b) shows our estimated pose, (c) shows a normal direction, and (d) shows a synthesized novel view by rotating along y-axis. First 4 rows are correct estimations and last 2 rows are incorrect. Note that an error on the 5th row is misaligning a bookcase by 1 shelf. Near symmetry and repetition cause this type of top false positives.

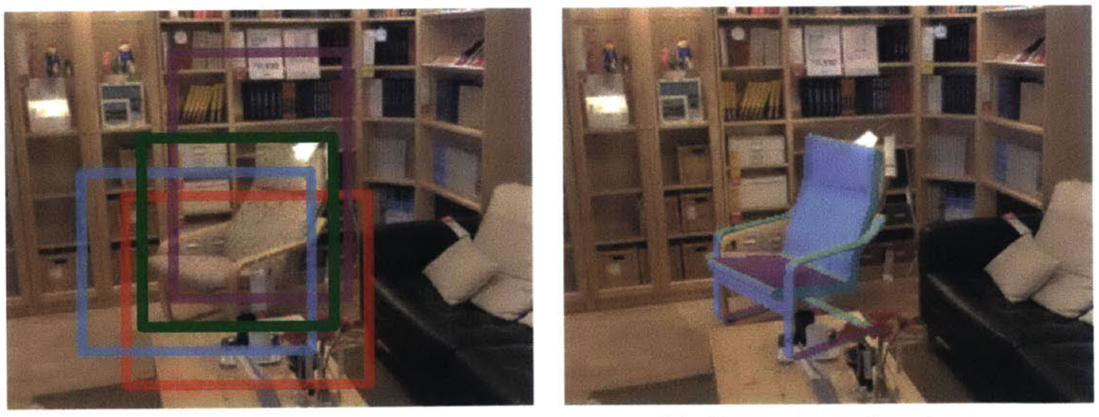
■ 2.4 Conclusion

We have introduced a novel problem and model of estimating fine-pose of objects in the image with exact 3D models, combining traditionally used and recently developed techniques. Moreover, we also provide a new dataset of images fine-aligned with exactly matched 3D models, as well as a set of 3D models for widely used objects. We believe our approach can extend further to more generic object classes, and enable the community to try more ambitious goals such as accurate 3D contextual modeling and full 3D room parsing.

Chapter 3

FPM: Fine pose Parts-based Model with 3D CAD models

Imagine an autonomous agent attempting to find a chair to sit on. Using a state-of-the-art object detection system, the agent finds a number of *correct* detections (as defined by the 0.5 intersection-over-union criterion) of chairs as shown in Figure 3.1(a). With the given set of *correct* detections, the agent could end up sitting anywhere from the bookshelf, to the floor! In order to better understand and interact with our environment, it is important to tackle the problem of fine pose estimation as shown in Figure 3.1(b).



(a) standard object detection (b) fine pose estimation

Figure 3.1: If an autonomous agent attempted to sit on a chair based on *correct* detections by an object detector (a), who knows where it might end up? Fine pose estimation, shown in (b), is one possible method to tackle this problem.

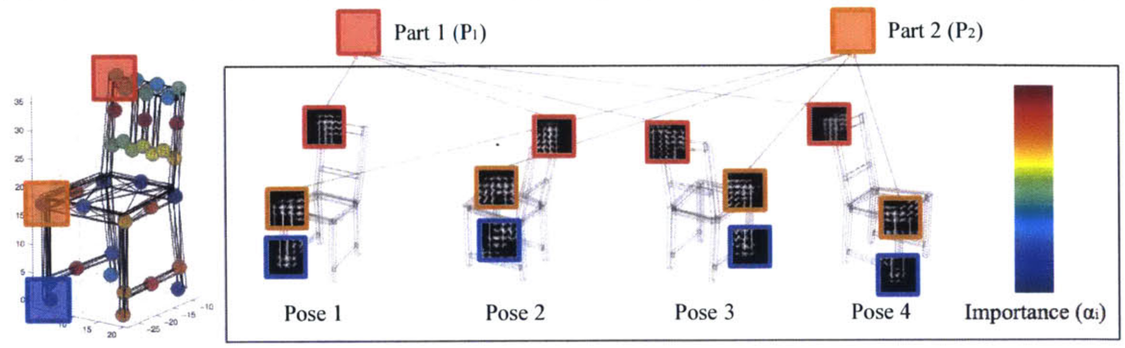


Figure 3.2: Our goal is to learn a model that can estimate the fine pose of objects from a single image. Like deformable part models [20], we approach this problem by training root and part templates for a large number of views. While each part P_i has its own shape template w_i trained from rendered images, its importance weight α_i is trained with a small set of annotated real images. The goals of importance weight α_i for each part P_i are: (1) to be shared across different views in order to be able to train them only with a small set of real images, and (2) to learn the importance of each part. For example, some parts are occluded by other objects more often than others, and some parts are more discriminative than others. The importance weight α_i will encode this information.

We discussed the same problem in Chapter 2. In this chapter, we propose an improved algorithm that combines appearance information from images with geometric information from 3D models for efficient fine pose estimation. Specifically, we introduce the notion of 3D part sharing that is enabled through the use of 3D models. As illustrated in Figure 3.2, 2D parts [20, 21] are significantly affected by viewpoint, while 3D parts are shared across views. For each 3D part, we learn an *importance score* that measures the visibility and discriminative-ness of the given part. Our algorithm outperforms existing state-of-the-art methods in both speed and accuracy.

■ 3.1 FPM: Fine pose Parts-based Model

Given a set of CAD models, our goal is to accurately detect and pose-align them in RGB images if they contain instances of those objects. Let us formulate this problem in the standard object detection framework, say DPM [20]: we would (1) divide a single object into multiple mixture components - one corresponding to each 3D pose of the object we want to detect, (2) perform detection and (3) attempt to determine the exact mixture component that is responsible for the detected object so that we could identify the pose of the object. Let us do a simple thought experiment to determine the number of mixtures required: first, to parameterize the pose of an object, we require 6 values (3 rotation and 3 translation parameters). Then, we perform a rather coarse discretization of these parameters: say 5 x- and y-translations, 5 depths, 30 y-rotations, 5x- and z-rotations. This leads to 93750 mixture components[1] compared to the 2 or 3 usually used in DPM!

Now, let us assume that it were possible to do inference efficiently with 94k mixtures (each containing multiple parts!), but what about training these models? Even if we used a single example for training each mixture, we would need to take 94k images of a single object. This can be extremely difficult, if not impossible, to capture using physical objects.

In this work, we use CAD models to address the drawbacks above to efficiently and accurately detect and estimate the fine pose of objects using a small set of real images. We summarize the **advantages** of using CAD models below:

- CAD models allow us to render virtually an infinite set of views and discern between them relative to a given point of reference.
- Since we have exact CAD models of objects, we do not need to allow significant deformation between parts. Thus, given the pose, we can estimate the exact location of parts in the image.
- With a CAD model, parts are defined in 3D space allowing us to share information of each part across views. This allows us to use a limited set of training data to learn the importance of each part in 3D, which can then generalize to novel views.

[1]In our implementation we actually have 324,000 components.

While CAD models provide an appealing source of data, they have one major **disadvantage**: they can be difficult to combine with real images, as the statistics of rendered and real images are often significantly different. For example, as shown in Figure 3.3(b), occlusions by other objects do not usually occur in rendered images, and further, the appearance of the two domains is also significantly different e.g. the CAD model does not have well textured surfaces.

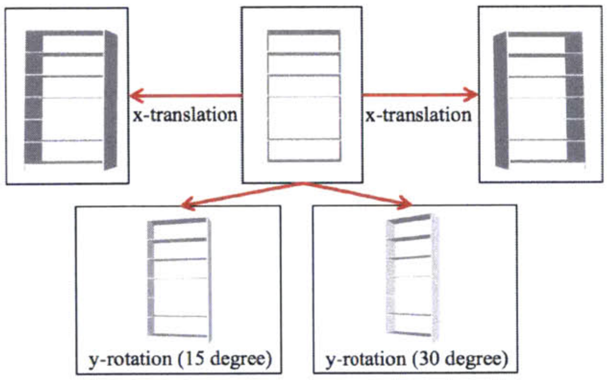
Thus, we need a model that combines the advantages of CAD models as described above, while addressing the difference in modality. To achieve this, we propose the Fine pose Parts-based Model (FPM) that combines the deformable part model (DPM)[20] trained on rendered images with objectness scores measured on real images. Specifically, we extend the DPM to have 3D shared parts that are trained using a large number of rendered images and a small number of real images. While the DPM with shared parts allows us to accurately estimate the pose of the object, the objectness scores allow us to leverage an unlabeled set of images to better estimate the location of the objects in the image. We describe our model in Section 3.1.1, the learning algorithm in Section 3.1.2 and finally the method of inference in Section 3.1.3.

■ 3.1.1 Model

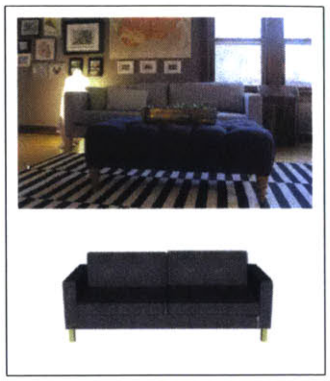
Here we define the problem more precisely: given a set of CAD models, and a small set of real images with the pose of the corresponding CAD models present in it, we want to train a model to perform fine pose estimation on new images containing the same set of objects, but with potentially different poses. Thus, we need a model that can robustly identify poses in test images that were not present in the training images.

In order to do this, we define a function $F_\Theta(x)$ that determines how well the pose, Θ, of a CAD model fits a rectangular image window, x:

$$F_\Theta(x) = \boldsymbol{\alpha}^\top \mathbf{S}_\Theta(x) + \boldsymbol{\beta}^\top \mathbf{O}_\Theta(x) + \boldsymbol{\gamma}^\top \mathbf{Q}_\Theta \tag{3.1}$$



(a) Different shapes from translation and rotation



(b) Difference of modalities

Figure 3.3: **(a)** This figure illustrates different shapes from various poses. It shows that fine poses cannot be covered by simply rotating objects at the center of the image. For example, y-rotation yields a different shape from x-translation. The line directions of the shelves do not match due to perspective projection. **(b)** Comparison between a real image and a rendered image. Rendered images do not have occlusion by other objects and lack other appearance details often seen in real images.

where $\mathbf{S}_\Theta$, $\mathbf{O}_\Theta$ and $\mathbf{Q}_\Theta$ are the scores from the DPM with shared parts, objectness and model quality, respectively. Further, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are the parameters defining the relative weight between the three terms learned using a max-margin framework as described in Section 3.1.2. Our goal is to maximize F_Θ for positive images/poses while minimizing it for negative images/poses. We describe the three terms, $\mathbf{S}_\Theta$, $\mathbf{O}_\Theta$ and $\mathbf{Q}_\Theta$ in detail below:

$\mathbf{S}_\Theta(x)$ - DPM with 3D shared parts:

In this section, we describe how to extend the deformable parts based model (DPM) [20] to work in our scenario. This is a challenging problem because we do not have real images for all possible poses of the objects that we want to be able to identify, but have CAD models instead with a small set of pose-aligned images. Here, we extend the standard formulation of DPM to better exploit the extra information available through the use of CAD models, such as self-occlusion. Further, we show how to propagate information from a small set of pose-aligned images to detect novel poses in images through the use of 3D shared parts.

First, we begin by describing a simple parts-based model that can be trained using rendered images alone. For an image window x, and a pose Θ of a particular CAD model, the score, s_Θ, is given by:

$$(x) = \max_{P_i} \left[s_\Theta^r(x) + \sum_i s_\Theta^p(P_i, x) \right] \tag{3.2}$$

where $s_\Theta^r(x) = \mathbf{w}_\Theta \cdot x_{HOG}$ is the score of the root template with pose Θ, and $s_\Theta^p(P_i, x) = \mathbf{w}_\Theta^{P_i} \cdot x_{HOG} - \phi \cdot d_i$ is the score of part template i with location P_i relative to the window x. x_{HOG} refers to the HOG [13] features extracted from the image window x, and ϕ refers to the deformation cost of part i being at a distance d_i from its expected location in the image. We can consider each discretized Θ as a traditional mixture (i.e. $324,000$ mixtures in our case). $\mathbf{w}_\Theta$ refers to the weight vector for the root template and $\mathbf{w}_\Theta^{P_i}$ refers to weight vector for part template i at pose Θ. Since exact 3D models are used here, we do not expect significant deformation of parts relative to the root template given a particular pose. Thus, we manually fix ϕ to a specific value.

Now, we describe the modifications made to the above model for our setting:

Obtaining parts: Unlike [20], we do not treat parts as latent variables. Instead, we find parts in 3D space by identifying 'joints' in the 3D model as shown in Figure 3.4(a). Further, when adjacent joints connected via the mesh exceed a certain distance, an additional part is added in between. This results in about 10 to 30 parts per CAD model.

Learning mixture components: Given the large number of mixture components, it can be intractable to learn the weights $\mathbf{w}_\Theta$ using an SVM even when using rendered images because of computationally expensive steps such as hard-negative mining. As described in [38], we can use exemplar-LDA to efficiently learn the weights for the root and part templates. Thus, for a given CAD model, the weights, $\mathbf{w}_\Theta$, can be approximated as:

$$\hat{\mathbf{w}}_\Theta = \Sigma_{real}^{-1}(\mu_\Theta^+ - \mu_{real}^-) \tag{3.3}$$

where μ_Θ^+ refers to the mean of the HOG features of images rendered at pose Θ, while Σ_{real} and μ_{real} are the covariance matrix and mean vector computed from *real* images. It is important to learn Σ_{real} and μ_{real} from real images in order to account for the difference in modalities. Note that we do not require annotated images to learn these matrices[2], and they can be efficiently modified for templates with different sizes as described in [38]. We follow a similar procedure to learn the weights for the parts.

Part visibility: Since we use 3D CAD models, we can easily determine when a part becomes invisible due to self-occlusion. Thus, we multiply the s_Θ^p term in Eq. 3.2 with a binary variable $v_\Theta(P_i)$ for the visibility of each individual part P_i at pose Θ. It is set to 1 if a part is visible and 0 otherwise.

3D shared parts: As mentioned above, the train data contains a small set of real images with aligned poses - specifically, while we want to distinguish between 324,000 unique poses, we only have at most 50 real images with aligned poses for training per CAD model in our dataset. How can we use this rather restricted amount of data to learn a model that generalizes to all poses?

Here, we propose to use parts shared in 3D space to propagate the information from the observed poses to the unobserved poses i.e., if we marginalize out the viewpoint, we only need to learn the importance of each 3D part of the model. This allows us to leverage information such as occlusion patterns, and the discriminativeness of different parts from real images. For example, in Figure 3.3(b), we would expect the sitting area of a sofa to be occluded by other objects (e.g. cushion) more often than the handles of

[2]We use the covariance matrix and image mean provided by the authors of [38].

the sofa. Further, some parts (e.g. corners) of a sofa could be more discriminative than others (e.g. lines). Thus, using 3D shared parts, we can propagate this information to all views.

Specifically, we define importance weights, α_i for each 3D shared part. Using 3D CAD models, obtaining part locations and correspondences between two views is trivial. Using the correspondence, we only have to enforce the same α_i for each P_i for all Θ. Figure 3.4(b) illustrates some of the learned α_i for two different models. We observe that parts that are typically occluded by other objects tend to have lower weights, while parts with more discriminative shapes tend to have higher weights. Similar to [2], α_i is used to rescale the part scores, but in this case we enforce α_i to be shared across views, and learn it directly from real images. Further, it is worth noting that learning α_i would be infeasible in our scenario without sharing.

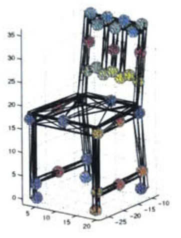
Thus, the term $\mathbf{S}_\Theta(x)$ is given by:

$$\mathbf{S}_\Theta(x) = \begin{bmatrix} s^r_\Theta(x) & v_\Theta(P_1)s^p_\Theta(P_1, x) & \dots & v_\Theta(P_N)s^p_\Theta(P_N, x) \end{bmatrix}^\top \tag{3.4}$$

where N is the number of parts in our model, and thus $\boldsymbol{\alpha} \in \mathbb{R}^{N+1}$. Note that $\boldsymbol{\alpha}$ here is the same as that mentioned in Eq. 3.1. We learn its value jointly with $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ in a max-margin framework, as described in Section 3.1.2.

$\mathbf{O}_\Theta(x)$ - objectness:

While the first term, $\mathbf{S}_\Theta$, in Eq. 3.1 largely deals with variation in geometry (and some appearance) across views, it suffers from the fact that rendered images are used for training, while real images are used for testing. While we have a limited set of pose-aligned images, we can use the generic concept of objectness [1] to identify whether an image window contains an object or not. We can simply obtain the objectness score for each image window, x, by using a typical objectness classifier [1]. However, in


(a) Object parts

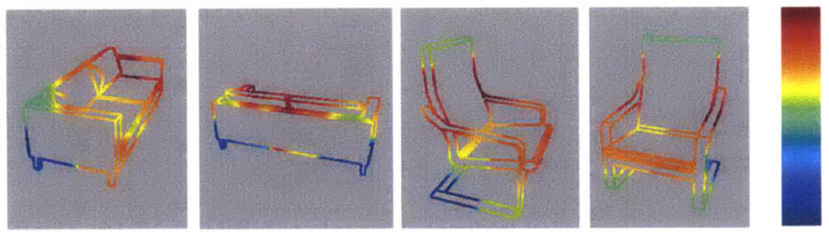

(b) Part importance (increases from blue to red)

Figure 3.4: **(a)** Example of object parts for the given model (indicated by the different colored spheres) **(b)** Here, we visualize the importance weights, α_i, for each of the parts $i \in \{1 \ldots N\}$ as described in Section 3.1.1. Each part has its own shape template similar to other part-based methods [20]. However, our parts have additional (importance) weights which encode the importance of particular parts. The sharing of these weights enables us to train with rendered images using only a small number of annotated real images. The goal is to learn which part is frequently occluded by other objects or does not contain discriminative shapes from the real data, as this information cannot be derived from 3D CAD models alone. For example, the rear bottom parts of sofa and chair are often occluded by other objects and hence have very low weights.

order to leverage the representational ability of the recent state-of-the-art deep learning features [56], we simply re-train the objectness classifier (Linear SVM) using selective search [100] and deep learning features extracted using Decaf [17] on the PASCAL VOC dataset [18]. For efficiency, we cache the objectness scores of all the selective search windows in both the train and test splits of our data.

Note that since we use selective search here, we cannot find the objectness scores for the exact windows used in our fine pose detection framework. Instead, given an arbitrary image window x, we find the selective search window that has the highest value of intersection over union and use its objectness score. In this way, we can find the value of function $\mathbf{O}_\Theta(x)$ for any image window x.

$\mathbf{Q}_\Theta$ - pose quality:

When training with rendered images, the typical confident false-positives are from the views that are too *general* e.g., the back and side views of a bookcase are simply rect-

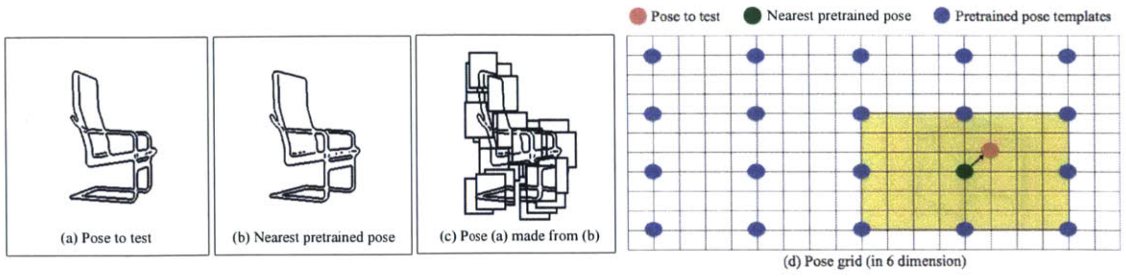

Pose to test
Nearest pretrained pose
Pretrained pose templates
(a) Pose to test
(b) Nearest pretrained pose
(c) Pose (a) made from (b)
(d) Pose grid (in 6 dimension)

Figure 3.5: **Pose sliding window:** During inference, when we match pose Θ (red dot) to the given image, we use the part templates from the nearest pre-trained pose Θ' (green dot). The only change from Θ' is the part location. Part locations are estimated based on the target pose Θ, and the part templates are trained based on Θ'. Blue dots indicate the poses that are pre-trained and hence have part templates, and yellow region indicates the area where the green dot could be covering.

angles; they are not very discriminative and can easily match to a door, a whiteboard, or even a TV screen. We observe that the more near-empty cells a view of a model has, the more it suffers from false-positives. To address this, we use two terms that are suggestive of the *emptiness* of a given model view: (1) the norm of the root weight template at pose Θ, $||\hat{\mathbf{w}}_\Theta||$, and (2) the number of visible mesh surfaces at pose Θ, n_Θ. Thus, $\mathbf{Q}_\Theta = [||\hat{\mathbf{w}}_\Theta||,\ n_\Theta]^\top$.

■ 3.1.2 Learning

In this section, we describe how we learn the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ of the model defined in Eq. 3.1. Note that all the functions $\mathbf{S}_\Theta$, $\mathbf{O}_\Theta$ and $\mathbf{Q}_\Theta$ as described in Section 3.1.1 are well defined i.e. they do not contain parameters that we need to learn here. Thus, Eq, 3.1 becomes a linear system that we can solve in a max-margin framework. Specifically, we learn $W = [\boldsymbol{\alpha}\ \boldsymbol{\beta}\ \boldsymbol{\gamma}]^\top$ using a linear SVM where the positive pose-aligned images are given in the training set, and we obtain negatives using random sampling of pose across different images. Further, we refine the weights using hard negative mining, similar to [20]. The SVM hyperparameter, C, is found using 5 fold cross-validation. We apply the same procedure to all CAD models independently.

■ 3.1.3 Inference

During inference, we evaluate the function F as defined in Eq. 3.1 for each pose Θ. Given the difficulty of training separate classifiers for a continous pose space Θ, we instead discretize the pose into 9 x- and y-translations, 5 depths, 32 y-rotations, $5x$- and z-rotations leading to 324,000 discretized poses $\{\Theta'_i\}$. Then, during inference, for each fine pose Θ, we find the nearest neighbor Θ'_i and borrow its trained weights for the root ($\hat{\mathbf{w}}_\Theta$) and part templates ($\hat{\mathbf{w}}_\Theta^{(i)}$). Figure 3.5 illustrates our approach. In essence, this is extremely similar to the sliding window approach, where each model from a discretized view is only run within the neighboring region of pose space where no discretized poses are present.

After all high scoring pose candidates are obtained, we also perform non-maximal suppression in pose space to obtain the final detection results. Further, to make the sliding window computation efficient, we use sparse coding on the weights of the root and part models as described in [29] allowing us to search this rather large space of fine poses in a reasonable amount of time.

■ 3.2 Experiments

In this section, we evaluate the performance of our approach in detail, focusing on its ability to recover fine-pose. In particular, we evaluate both the average precision of pose estimation (Sec 3.2.1) and recall among top predictions (Sec 3.2.2). Further, we compare our method on a standard bounding box detection problem to other state-of-the-art methods (Sec 3.2.3).

Dataset: We use the dataset provided in [63] that contains pose-aligned 3D CAD models with real images. Specifically, we use the harder of the two datasets provided in [63]: the IKEA Room dataset. This dataset contains about 500 images where object poses are annotated for all available 3D CAD models.

■ 3.2.1 Fine Pose Estimation

We compare the previous state-of-the-art method [63] and our method on various settings. We used average precision (AP) measure as suggested by [63] to evaluate our performance. Note that this AP measure takes pose into account, and not only the bounding box location. The normalized 3D space distance between the ground truth pose and the predicted pose is thresholded to determine true or false prediction. The results are summarized in Table 3.1.

We first evaluate our method at various settings: *only root*, *root+part without* α, *root+part with* α, and *all*. In case of *only root* and *root+part without* α, the detectors suffer from many false positives as compared to [63]. Many false positives occur by some poses having abnomarlly higher confidences than others. However, when we add sharing α and the full set of terms as proposed in Eq 3.1, our performance increases significantly as we leverage information learned from real images (part importance and objectness).

We further analyze how well our method performs at estimating pose when the bounding box localization is correct. For each ground truth object, we find its first estimated pose that has the highest confidence and has at least 0.7 intersection over union with the ground truth. Then, we compute the ratio of the number of correctly pose estimated objects to the total number of objects. Table 3.2 shows the result. The average performance of pose estimation given the ground truth bounding box is quite high, 0.84. This indicates that the pose alignment is reliable when the method can localize the target object.

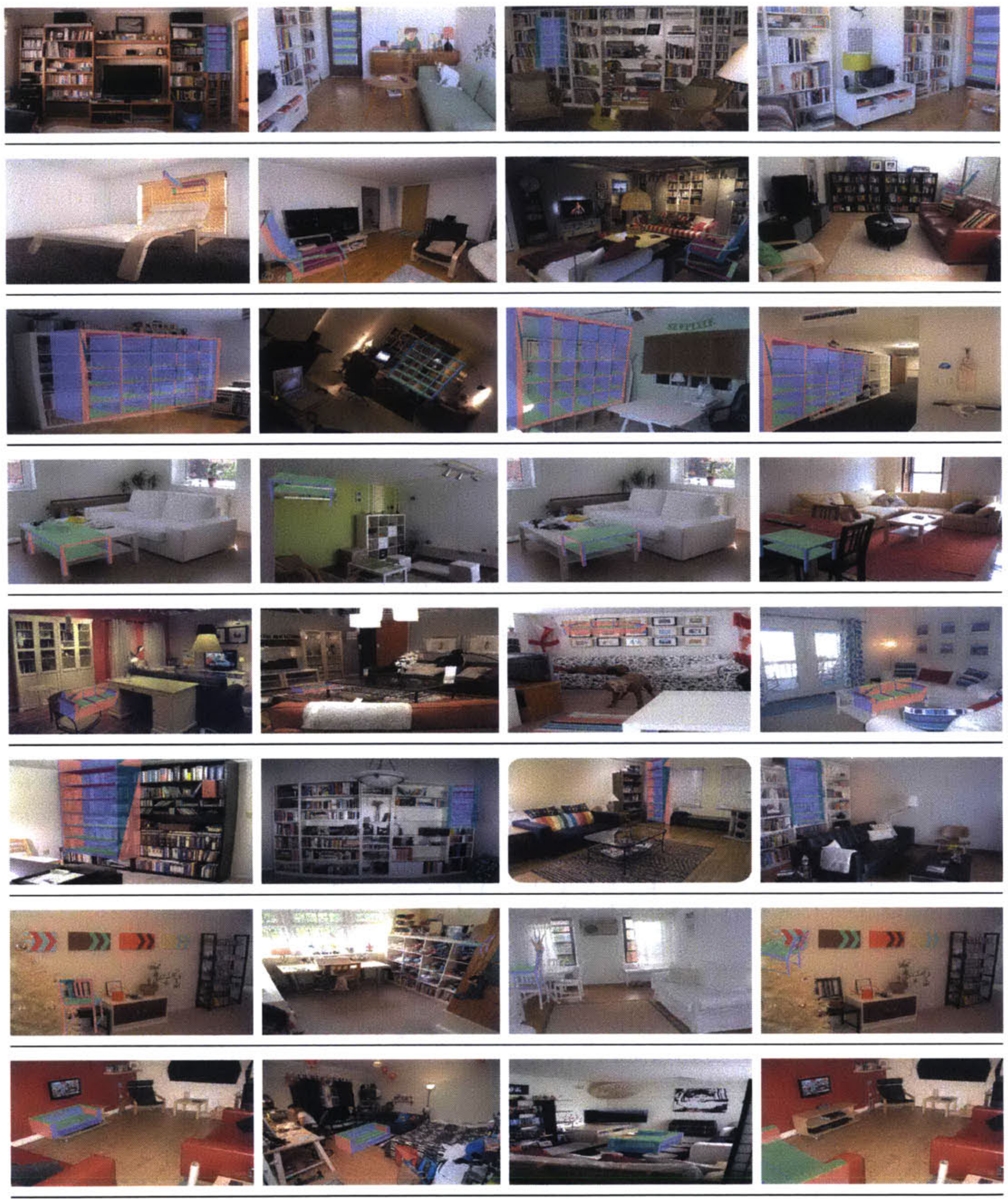
Finally, we show qualitative results in Figure 3.6 and Figure 3.7. Figure 3.6 shows the top 4 false positive results with highest confidence from each class. There are interesting common behaviors. Many misalignments are due to shift in the global position but most of major parts are well-aligned. For example, most bookcase false positives are just one shelf unit shifted ground truth. This repetitive structure of bookcases is a common source of false positives and causes the major performance drop. Also, table and chair have most parts well-aligned to the ground truth, but are often assigned the wrong scale or produce flipped orientations. The major performance drop on the sofa class is due to the fact sofas undergo heavy occlusion, have no strong boundaries, and have complex textures compared to other classes. These differences make training with rendered images harder. Figure 3.7 shows some of the top correct detections. It is clear that when the correct pose is found, the alignment is reliable.

	bookcase *billy1*	chair *poang*	bookcase *expedit*	table *lack2*	sofa *karlstad*	bookcase *billy5*	chair *stefan*	sofa *ektorp*	mean
IKEA [63]	7.53	9.43	3.28	9.14	3.22	4.21	14.87	0.48	6.52
R	1.45	1.90	0.24	5.86	0.19	1.19	4.10	0.00	1.87
R+P	3.11	7.24	3.21	13.90	2.84	4.05	7.61	0.36	5.29
R+P+S	6.37	9.11	6.78	14.00	6.23	5.34	9.66	1.80	7.41
Full FPM	**10.52**	**14.02**	**9.30**	**15.32**	**7.15**	**6.10**	**16.00**	**5.66**	**10.51**

Table 3.1: **Pose AP score:** we compare our method against the previous state-of-the-art algorithm on IKEA room dataset for fine pose estimation. R, P and S refer to Root, Parts and Sharing respectively. When we use Root and/or Parts that are trained only on rendered images, the performance suffers. However, when combining with part sharing and objectness, our performance improves significantly and outperforms the previous state-of-the-art IKEA algorithm by Lim et. al. [63] (Chapter 2).

bookcase *billy1*	chair *poang*	bookcase *expedit*	table *lack2*	sofa *karlstad*	bookcase *billy5*	chair *stefan*	sofa *ektorp*	mean
0.85	0.93	0.96	0.90	0.82	0.75	0.90	0.63	0.84

Table 3.2: **Pose estimation given the oracle:** Here we investigate how well our pose estimation method works when the object detection step was successful. For each ground truth object, we evaluate pose estimation only within the top detected bounding box that has at least 0.7 intersection over union with ground truth. The numbers indicate the proportion of pose estimates, within this set, that are correct. The average performance of pose estimation given the ground truth is quite high, 0.84.



1st false positive 2nd false positive 3rd false positive 4th false positive

Figure 3.6: **Top 4 false positives per class:** (1) Many bookcase false positives are one shelf unit shifted from the ground truth. The repetitive structure of bookcase is a common source of false positives. (2) The major performance drop on the sofa class is due to the fact sofas undergo heavy occlusion, have no strong boundaries, and have complex textures compared to other classes. These difficulties make training with rendered images harder.

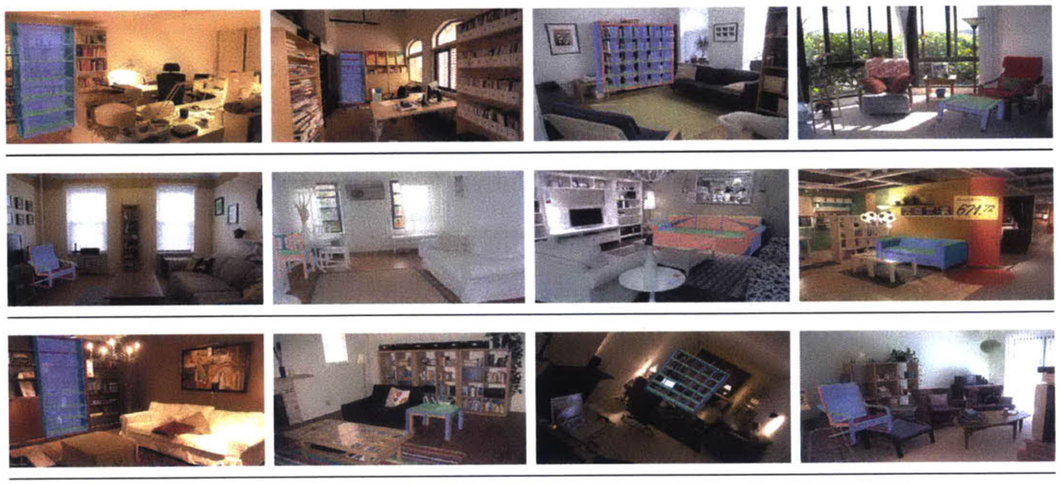


Figure 3.7: **Top correct detections:** We show some of the top detections from each class.

	bookcase *billy1*	chair *poang*	bookcase *expedit*	table *lack2*	sofa *karlstad*	bookcase *billy5*	chair *stefan*	sofa *ektorp*	mean
[63]	0.83	0.86	0.89	0.83	0.37	0.88	0.77	0.56	0.75
FPM	**0.87**	**0.93**	**0.94**	**0.88**	**0.83**	**0.91**	**0.99**	**0.91**	**0.91**

Table 3.3: **Pose Proposal:** we measure the recall of our method among the top 2000 windows. It shows that our method can recover most of the poses for all objects within the top 2000 predictions. We also outperform [63].

■ 3.2.2 Pose Proposal

Recently, bounding box proposal algorithms have received a lot of attention [1, 27, 100]. The main motivation is to prune search windows by proposing a set of bounding boxes with a high recall on each image that can later be processed using an algorithm that is computationally more expensive. Following this motivation, we evaluate our algorithm for *pose proposal*. Instead of proposing a set of possible good bounding boxes, our algorithm can generate a set of pose candidates. The algorithm stays the same; however the emphasis of evaluation is now focused more on recall than precision, as compared to the AP measure. For example, with the AP measure, if the top 20% predictions are all correct without any further recall, one can still achieve 0.20 AP score; while this would not be able to serve the role of *pose proposal.*

Table 3.3 shows the result of recall for all methods. Because all methods will have a full recall in the limit, we limit the number of predictions to be under 2,000. Our method shows quite reliable recalls at all classes, while [63] fails severely to recall some classes. We believe this is because our model can handle flexible views without being limited to the preset variations of views. [63] depends on a bounded number of views and uses a RANSAC process that can fall into local minima. On the other hand, our method exhaustively examines all possible views. From this result, we can conclude that our method can effectively propose candidate poses for other post-processing algorithms (e.g. context modeling or segmentation).

■ 3.2.3 Bounding Box Detection

While object bounding box detection is not our main target problem, our approach can nonetheless be used for this purpose. For pose estimation algorithms, we extract bounding boxes from predicted poses and apply non-max suppression from [20] before evaluation. We however train [20] with real images using the author-provided code. We evaluate the results at two different thresholds on bounding box intersection over union to examine an ability to capture fine details.

At a lower threshold of 0.5, DPM [20] performs strongly, and our method is relatively weak (in Table 3.4a). However, at a higher threshold of 0.8, our method outperforms [20] (in Table 3.4b). This result shows that [20] has a good coarse localization ability but fails capturing fine details. Because our method learns based on each fine pose, the result tends to have a better overlap with the ground truth than that of DPM. Note that DPM is fully trained with real images, and our method is trained based on rendered images and used real images to adapt. Further, we could improve our performance at the 0.5 detection threshold by incorporating scores from DPM into our model.

	bookcase *billy1*	chair *poang*	bookcase *expedit*	table *lack2*	sofa *karlstad*	bookcase *billy5*	chair *stefan*	sofa *ektorp*	mean
(a) Intersection over Union ≥ 0.5									
IKEA [63]	24.41	28.32	21.73	11.12	22.65	11.22	28.57	2.37	18.80
DPM [20]	**49.89**	**51.63**	**71.87**	**48.85**	34.01	**42.11**	**45.34**	**28.80**	**46.56**
FPM	23.51	29.83	37.26	38.16	**35.85**	33.00	30.52	27.13	31.91
(b) Intersection over Union ≥ 0.8									
IKEA [63]	**20.34**	14.43	15.74	9.14	15.32	7.73	20.45	1.58	13.09
DPM [20]	9.41	15.58	15.47	10.02	20.12	3.05	20.44	11.59	13.21
FPM	17.37	**22.36**	**22.89**	**29.88**	**22.26**	**8.71**	**24.31**	**12.64**	**20.05**

Table 3.4: **Bounding box AP measure:** we compare our method against other state-of-the-art algorithms on the IKEA room dataset for bounding box detection. As we increase the threshold to 0.8, the performance of DPM drops significantly; while our method maintains the performance. It shows that our method is capable for fine detection. Our method also significantly outperforms the previous state-of-the-art algorithm [63] in both scenarios, obtaining higher AP for most of the object categories.

3.3 Conclusion

In this chapter, we introduced a novel approach for fine-pose estimation that leverages geometric information from CAD models to address the problem of fine pose estimation. We show that our method is successfully able to combine the appearance information from relatively few real training images with rendered images from CAD models, outperforming existing approaches in various tasks. Notably, our algorithm significantly outperforms deformable part-based models [20] for high overlap detection, and significantly improves pose recall as compared to [63]. We believe that our work provides a platform to tackle higher level tasks such as contextual reasoning in scenes using the fine poses of objects.

Chapter 4

Discovering States and Transformations in Image Collections

In Chapter 2 and Chapter 3, we studied how to find objects and estimate their pose in 3D. Now, this chapter will focus on a different dimension of properties of an object – its states. While a single object class can consist of various states, much work in computer vision has focused on the problem of invariant object recognition [14, 18], scene recognition [106, 111], and material recognition [89]. In other words, the goal in each of these cases is to build a system that is invariant to all within-class variation. Nonetheless, the variation in a class is quite meaningful to a human observer. Consider Figure 4.1. The collection of photos on the left only shows tomatoes. An object recognition system should just see "tomato". However, we can see much more: we can see peeled, sliced, and cooked tomatoes. We can notice that some of the tomatoes are riper than others, and some are fresh while others are moldy.

Given a collection of images of an object, what can a computer infer? Given 1000 images of tomatoes, can we learn how tomatoes work? In this chapter, we take a step toward that goal. From a collection of photos, we infer the states and transformations depicted in that collection. For example, given a collection of photos like that on the

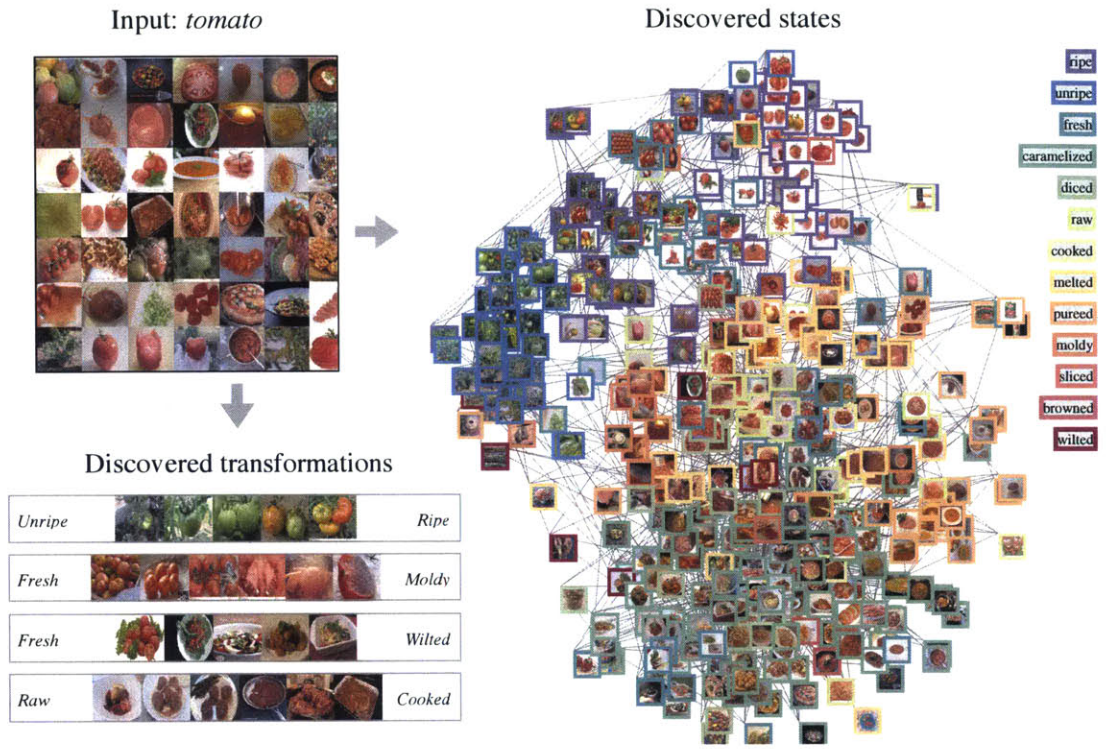


Figure 4.1: **Example input and automatic output of our system:** Given a collection of images from one category (top-left, subset of collection shown), we are able to parse the collection into a set of states (right). In addition, we discover how the images transform between antonymic pairs of states (bottom-left). Here we visualize the transformations using the technique described in Section 4.4.5.

left of Figure 4.1, we infer that tomatoes can be undergo the following transformations, among others: ripening, wilting, molding, cooking, slicing, and caramelizing. Our system does this without having ever seen a photo of a "tomato" during training (although overlapping classes, such as "fruit", may be included in the training set). Instead we transfer knowledge from other related object classes.

To demonstrate our understanding of states and transformations, we test on three tasks. As input we take a set of images depicting a noun class our system has never seen before (e.g., *tomato*; Figure 4.1). We then parse the collection:

- Task 1 – Discovering relevant transformations: What are the transformations that the new noun can undergo in (e.g., a *tomato* can undergo *slicing*, *cooking*, *ripening*, etc).

- Task 2 – Parsing states: We assign a state to each image in the collection (e.g., *sliced*, *raw*, *ripe*).
- Task 3 – Finding smooth transitions: We recover a smooth chain of images linking each pair of antonymic states.

Similarly to previous works on transfer learning [8, 15, 62, 95], our underlying assumption is the transferrability of knowledge between adjectives (state and transformation) (see Fig 4.2). To solve these problems, we train classifiers for each state using convolutional neural nets (CNNs) [17]. By applying these classifiers to each image in a novel image set, we can discover the states and transformations in the collection. We globally parse the collection by integrating the per-image inferences with a conditional random field (CRF).

Note that these tasks involve a hard generalization problem: we must transfer knowledge about how certain nouns work, like *apples* and *pears*, to an entirely novel noun, such as *banana*. Is it plausible that we can make progress on this problem? Consider Figure 4.2. *Melted chocolate*, *melted sugar*, and *melted butter* all look sort of the same. Although the material is different in each case, "meltedness" always produces a similar visual style: smooth, goopy, drips. By training our system to recognize this visual style on *chocolate* and *sugar*, we are able to detect the same kind of appearance in *butter*. This approach is reminiscent of Freeman and Tenenbaum's work on separating style and content [96]. However whereas they focused on classes with just a single visual style, our image collections contain many possible states.

Our contribution in this chapter is threefold: (1) introducing the novel problem of parsing an image collection into a set of physical states and transformations it contains (2) showing that states and transformations can be learned with basic yet powerful techniques, and (3) building a dataset of objects, scenes, and materials in a variety of transformed states.

■ 4.1 States and transformations dataset

The computer vision community has put a lot of effort into creating datasets. As a result, there are many great datasets that cover object [14, 18, 66, 84, 103], attribute [4, 19, 55], material [89], and scene categories [106, 111]. Here, our goal is to create an extensive dataset for characterizing state variation that occurs within image classes. How can we organize all this variation? It turns out language has come up with a solution: adjectives. Adjectives modify nouns by specifying the state in which they occur. Each noun can be in a variety of adjective states, e.g., *rope* can be *short*, *long*, *coiled*, etc. Surprisingly little previous work in computer vision has focused on adjectives [33, 57].

Language also has a mechanism for describing transformations: verbs. Often, a given verb will be related to one or more adjectives: e.g., *to cook* is related to *cooked* and *raw*. In order to effectively query images that span a full transformation, we organize our state adjectives into antonym pairs. Our working defining of a transformation is

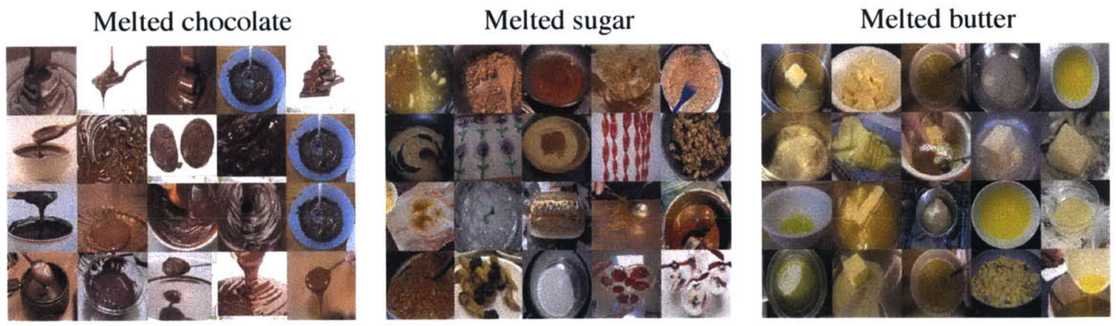


Figure 4.2: **Transferrability of adjective:** Each adjective can apply to multiple nouns. *Melted* describes a particular kind of state: a blobby, goopy state. We can classify images of *chocolate* as *melted* because we train on classes like *sugar* and *butter* that have similar appearance when *melted*.

thus a pair {adjective, antonym}.

We collected our dataset by defining a wide variety of {adjective, antonym, noun} triplets. Certain adjectives, such as *mossy* have no clear antonym. In these cases, we instead define the transformation as simply an {adjective, noun} pair. For each transformation, we perform an image search for the string "adjective noun" and also "antonym noun" if the antonym exists. For example, search queries included *cooked fish*, *raw fish*, and *mossy branch.*

■ 4.1.1 Adjective and noun selection

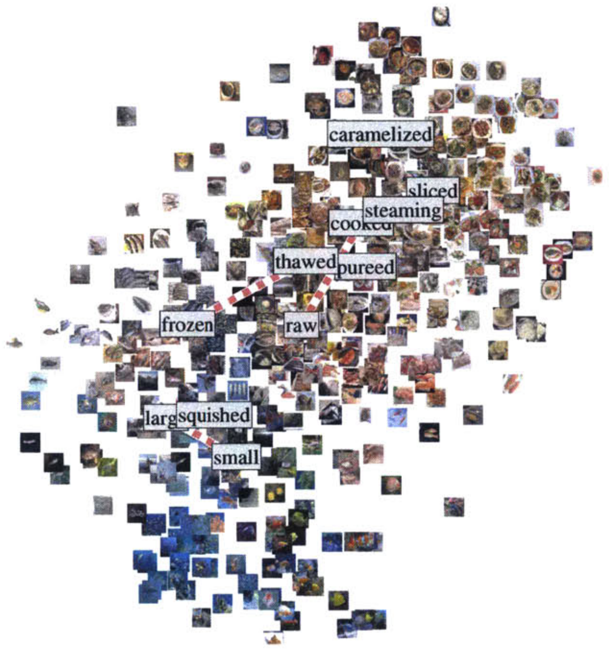
We generated 2550 "adjective noun" queries as follows. First we selected a diverse set of 115 adjectives, denoted $\mathcal{A}$ throughout this chapter, and 249 nouns, denoted $\mathcal{N}$. For nouns, we selected words that refer to physical objects, materials, and scenes. For adjectives, we selected words that refer to specific physical transformations. Then, for each adjective, we paired it with another antonymic adjective in our list if a clear antonym existed.

Crossing all 115 adjectives with the 249 nouns would be prohibitively expensive, and most combinations would be meaningless. Each noun can only be modified by certain adjectives. The set of relevant adjectives that can modify a noun tell about the noun's properties and affordances. We built our dataset to capture this type of information: each noun is paired only with a subset of *relevant adjectives.*

N-gram probabilities allow us to decide which adjectives are relevant for each noun. We used Microsoft's Web N-gram Services[1] to measure the probability of each {adj noun} phrase that could be created from our lists of adjectives and nouns. For each noun, $N \in \mathcal{N}$, we selected adjectives, $A \in \mathcal{A}$, based on pointwise mutual information, PMI:

[1]http://research.microsoft.com/en-us/collaboration/focus/cs/web-ngram.aspx

(a) Fish



(b) Room

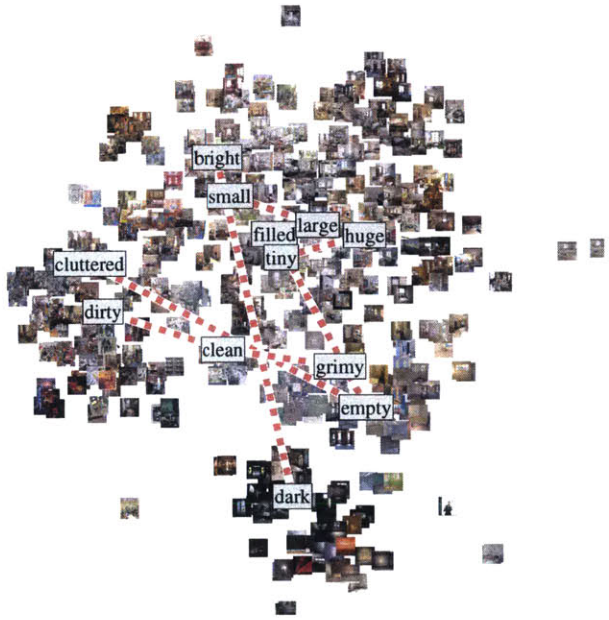


Figure 4.3: **Example categories in our dataset:** *fish* and *room*. Images are visualized using t-SNE [101] in CNN feature space. For visualization purposes, gray boxes containing ground truth relevant adjectives are placed at the median location of the images they apply to. Dotted red lines connect antonymic state pairs. Notice that this feature space organizes the states meaningfully.

$$\text{PMI}(A, N) = \log \frac{P(A, N)}{P(A)P(N)}, \tag{4.1}$$

where we define $P(A, N)$ to be the probability of the phrase "A N". PMI is a measure of the degree of statistical association between A and N.

For each noun N, we selected the top 20 adjectives A with highest

$$\min(\text{PMI}(A, N), \text{PMI}(ant(A), N))$$

, where $ant(A)$ is the antonym of A if it exists (otherwise the score is just PMI(A,N)). We further removed all adjectives from this list whose $\text{PMI}(A, N)$ was less than the mean value for that list. This gave us an average of 9 adjectives per noun.

■ 4.1.2 Image selection

We scraped up to 50 images from Bing by explicitly querying {adj, noun} pair, in addition to querying by only noun. While we scraped with an exact target query, the returned results are quite often noisy. The main causes of noise is {adj, noun} pairs being either a product name, a rare combination, or a hard concept to be visualized.

Hence, we cleaned up the data through an online crowd sourcing service, having human labelers remove any images in a noun category that did not depict that noun. Figure 4.3 shows our data for three noun classes, with relevant adjective classes overlaid.

■ 4.1.3 Annotating transformations between antonyms

While scraped images come with a weak state label, we also collected human labeled annotations for a subset of our dataset (218 {adj, antonym adj, noun} pairs). For these annotations, we had labelers rank images according to how much they expressed an adjective state. This data gives us a way to evaluate our understanding of the full transformation from "fully in state A's antonym" to "fully in state A" (referred to as ranking $ant(A)$ to A henceforth). Annotators split each noun category into 4 sections as the followings. We give examples for $A = open$ and $N = door$:

- "Fully A" – For example, fully open door images fall into this category.
- "Between-A and $ant(A)$" – Half-way open door images fall into this category.
- "Fully $ant(A)$" – Fully closed door images fall into this category.
- "Irrelevant image" – For example, an image of broken door lying on the ground.

We ask users to rank images accordingly by drag-and-drop.

■ 4.2 Methods

Our goal in this chapter is to discover state transformations in an image collection. Unlike the traditional recognition task, rather than recognizing an object (noun) or one attribute, we are interested in understanding an object (noun)'s states and the transformations to and from those states. There are various scenarios that we study for

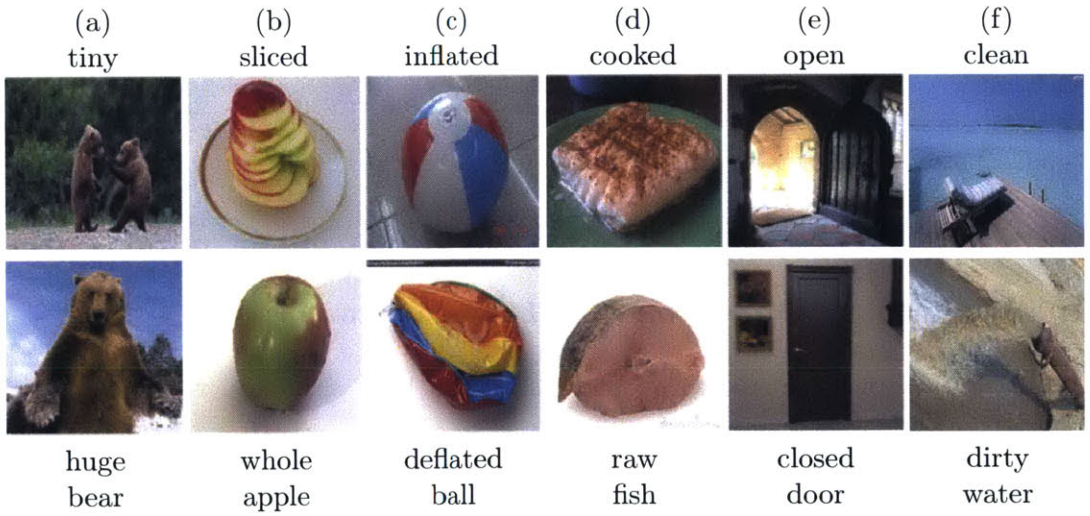


Figure 4.4: **Examples of objects in a variety of transformed states and their antonym states:** Notice that each state of an object has at least one antonym state.

this: singe image state classification, relevant transformation retrieval from the image collection, and ordering by transformation.

The common theme is to learn states and transformations that can generalize over different nouns. The reason behind this generalization criterion is from the fact that it is impossible to collect all training examples that can cover the entire space of $\{noun\} \times \{adjective\}$. Hence, in the following problem formulations, we always assume that no image from the specific target noun has been shown to the algorithm. For example, no apple image is used during training if we want to order images for the transformation to *sliced* apple.

■ 4.2.1 Collection parsing

Rather than classifying each image individually, we can do better by parsing the collection as a whole. This is analogous to the image parsing problem, in which each pixel in an image is assigned a label. Each image is to a collection as each pixel is to an image. Therefore, we call this problem *collection parsing*. We formulate it as a conditional random field (CRF) similar to what has been proposed for solving the pixel parsing problem (e.g., [90]). For a collection of images, $\mathbf{I}$, to which we want to assign per-image states $\mathbf{A}$, we optimize the following conditional probability:

$$\log p(\mathbf{A}|\mathbf{I}) = \sum_i g(A_i|I_i) + \lambda \sum_{i,j\in\mathcal{N}} \psi(A_i, A_j|I_i, I_j) + \log Z, \tag{4.2}$$

where Z normalizes, g serves as our data term, and the pairwise potential ψ is a similarity weighted Potts model:

$$\psi(A_i, A_j|I_i, I_j) = \mathbb{1}(A_i \neq A_j)\left(\frac{\xi + e^{-\gamma\|f(I_i)-f(I_j)\|2}}{\xi + 1}\right). \tag{4.3}$$

4.2.2 State classification

First, a simple task is to classify what is the most relevant adjective that describes a single image. Figure 4.4 shows examples of images in our dataset. Can we tell that the dominant state of Figure 4.4b is *sliced*? Also, can we tell how *sliced* the apple image it is? As mentioned above, we put a hard constraint that we never saw any apple image (including *sliced* apple) during the training stage. Our goal is to learn what it means to be *sliced* apart from all other nouns and be able to transfer the knowledge to a new category (e.g. apple) and infer the state of the image.

Our approach for solving this problem is training a logistic regression model. Let $N \in \mathcal{N}$ be the query noun that will be excluded from our test set. Then, using all non-N images, we split them into the positive and negative sets. To train a classifier for adjective $A \in \mathcal{A}$, the positive set is all images of A, and the negative set is all images not related to A. Then, the score of A for image I, denoted $g(A|I)$, can be easily computed by:

$$g(A|I) = \sigma(-w_A^T f(I)), \tag{4.4}$$

where σ is the sigmoid function, $f(I)$ is a feature vector of I, and w_A is a weight vector trained using a logistic regression model.

It is worth noting that each image can be in the mix of different states (e.g. an image of fish can be *sliced* and *raw*). However, for the simplicity, we assume each image has one dominant state that we want to classify.

4.2.3 Transformation ranking

Each image in our dataset depicts an object, scene, or material in a particular state. Unfortunately, since images are static, a single image does not explicitly show a transformation. Instead, we arrange multiple images in order to identify a transformation.

At present, we only investigate a simple class of transformations: transitions from "anytonym of some state A" to "fully in state A" ($ant(A)$ to A).

Figure 4.5 shows our goal. Given a set of images and an adjective A, we sort images $\{I_i\}$ based on $g(A|I_i) - g(ant(A)|I_i)$ (Eqn. 4.4).

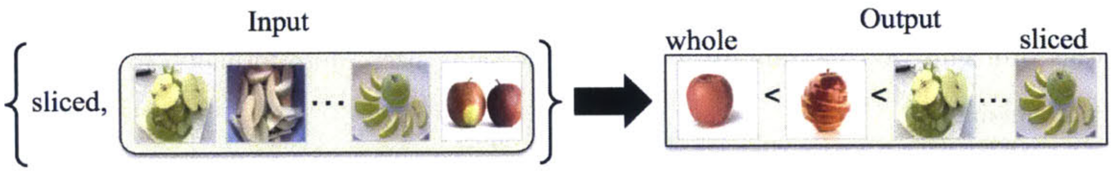


Figure 4.5: **Discovering transformation orders:** given a particular adjective A and a collection of images, our goal is to sort the images according to the transformation from $ant(A)$ to A. In this figure, we order images from *whole* to *sliced*. Note that we do not use any apple images while training.

■ 4.3 Which states are depicted in the image collection?

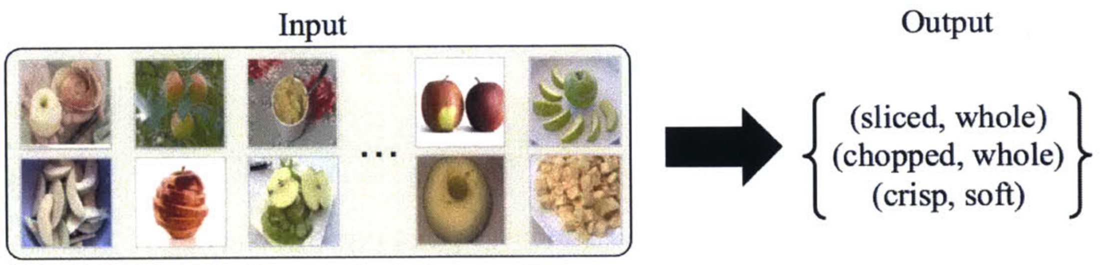


Figure 4.6: **Discovering transformations:** our goal is to find the set of relevant adjectives depicted in a collection of images representing one specific noun. In this figure, we want to predict the transformations that describe that describe a collection of apple images.

Our last problem is to discover the relevant set of transformations that are depicted in a *collection* of images. Figure 4.6 describes our task. We are given a set of *apple* images scraped from the web. While we assume our algorithm has never seen any of *apple* image, can we tell if this image collection contains the transformations between pairs of adjectives and their antonyms – (*sliced*, *whole*), (*chopped*, *whole*), and (*crisp*, *sofa*)?

We now formalize this task. We want to find the best adjective set, $\{\mathcal{A}_j\}_{j\in J}$, that can describe the collection of images, $\{I_i\}_{i\in\mathcal{I}}$, representing a single noun. We abbreviate this set as $\mathcal{A}_J$. Then, our goal is to predict what the most relevant set of adjectives and antonyms describing transformations, $\mathcal{A}_J$, for the given collection of images. In this problem, we constrain all J to have the same size. More formally, we find J by maximizing

$$J = \underset{J',|J'|=k}{\arg\max} \sum_{j\in J'} \sum_{i\in\mathcal{I}} \left[e^{\lambda g(\mathcal{A}_j|I_i)} + e^{\lambda g(ant(\mathcal{A}_j)|I_i)}\right]. \tag{4.5}$$

Rather than taking the sum over the raw $g(\cdot)$ scores, we take the exponential of this value, with λ being a free parameter that trades off between how much this function is like a sum versus like a max. In our experiments, we set λ to 10. Thus, only large values of $g(\mathcal{A}_j|I_i)$ contribute significantly to making $\mathcal{A}_j$ appear relevant for the collection.

4.4 Results

We evaluate three tasks:

1. Identification of relevant transformations for an image collection.
2. State classification per image.
3. Ranking images by transformation from $ant(A)$ to A.

4.4.1 Discovering relevant transformations

To implement $g(\cdot)$ (Equation 4.4), we used logistic regressions trained on CNN features [17] (Caffe Reference ImageNet Model[2], layer fc7 features). Figure 4.7 shows typical results for retrieval sets of size $|J| = 5$ (Equation 4.5). In order to ensure most effective

[2]http://caffe.berkeleyvision.org

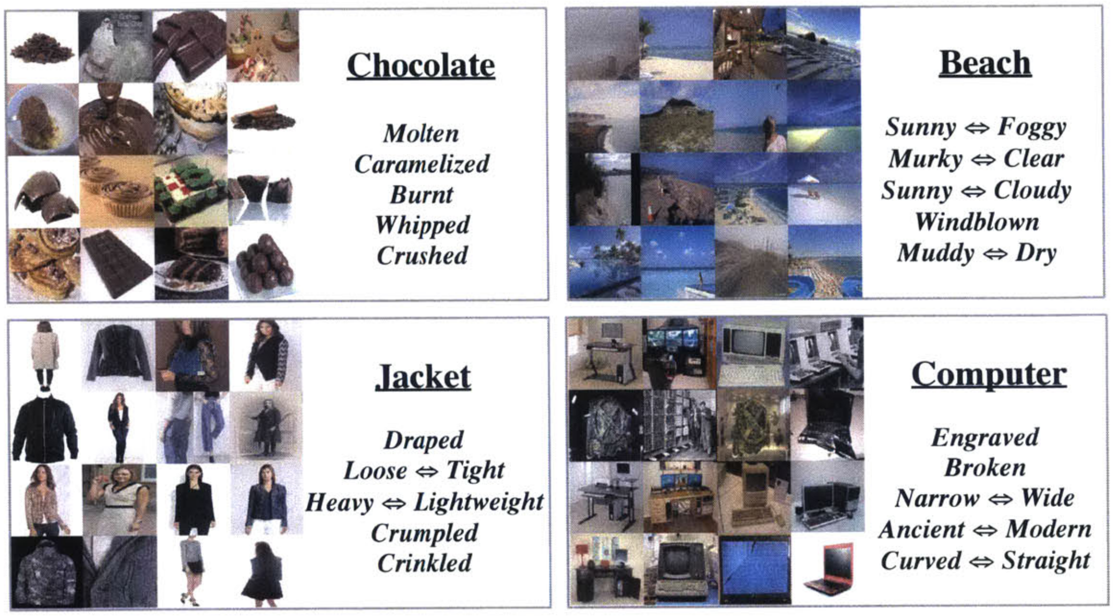
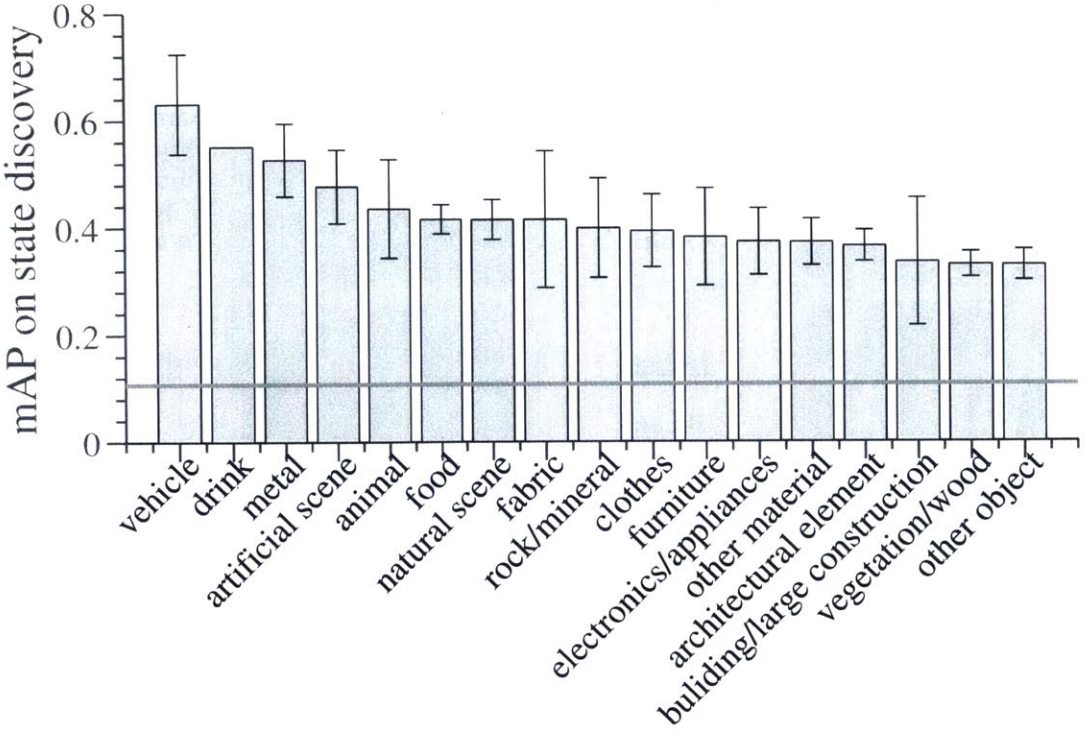


Figure 4.7: **Example results on discovering states:** Subset of image collection for each noun is to the left of discovered states. Our system does well on foods, scenes, and clothing (top row and left on the borrom row collections), but performs more poorly on objects like *computer* (bottom-right).

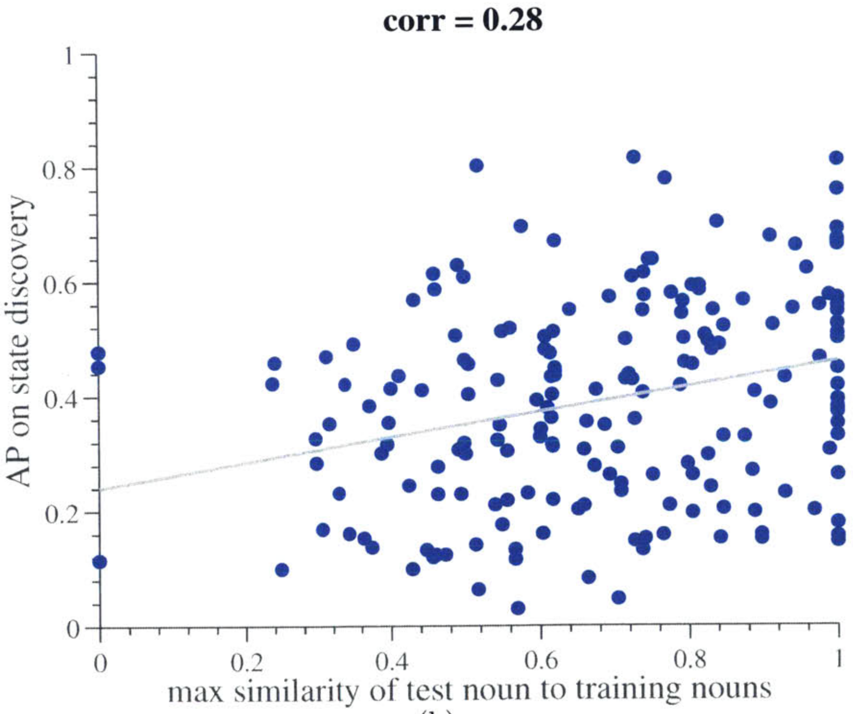
examples can be used, we priortize negative examples from nouns that contain the particular adj we are interested in. This type of generalizaiton technique has been explored in [19] as well.

We evaluated transformation discovery in an image collection as a retrieval task. We defined the ground truth relevant set of transformations as those {adjective, antonym} pairs used to scrape the images (Section 4.1.1). Our retrieved set was given by Equation 4.5. We retrieved sets of size $|J| = 1..|\mathcal{A}|$. We quantify our retrieval performance by tracing precision-recall curves for each noun. mAP over all nouns reaches 0.39 (randomly ordered retrieval: mAP = 0.11). Although quantitatively there is room to improve, qualitatively our system is quite successful at transformation discovery (Figure 4.7).

In Figure 4.8(a), we show performance on several meta-classes of nouns, such as



(a)



(b)

Figure 4.8: (a) Mean AP at discovering states for different classes of noun. (b) Performance correlates with the semantic similarity between the training set and the test noun, but this does not fully explain why some image collections are easier to understand than others.

"metals" (e.g., *silver*, *copper*, *steel*) and "food" (e.g., *salmon*, *chicken*, *fish*). Our method does well on material and scene categories but struggles with many object categories. One possible explanation is that the easier nouns have many synonyms, or near synonyms, in our dataset. To test this hypothesis, we measured semantic similarity between all pairs of nouns, using the service provided by [37]. In Figure 4.8(b), we plot semantic similarity versus AP for all nouns. There is indeed a correlation between synonymy and performance (r = 0.28): the more synonyms a noun has, the easier our task. This makes since because it is easier to generalize to a novel noun when the training set contains many similar nouns. We investigate our ability to generalize across dissimilar nouns in Section 4.4.4.

■ 4.4.2 State classification

To implement state classification we optimize our CRF model from Equation 4.2 using the method from [6]. This gives us a maximum a posteriori (MAP) configuration of states per image in the collection. We evaluated the MAP classifications by measuring mean accuracy at correctly classifying the state of each image across all collections. We used the states from our set of discovered transformations as the label space for the CRF. It is also possible to run the CRF with all adjectives as candidates classes. However, using all adjectives does worse: mean accuracy drops from 12.46% to 11.72%. Thus, the relevant transformation discovering acts as a holistic context that improves the classification of each individual image.

In Figure 4.9 we show how performance varies as a function not the CRF parameters λ and γ (Section 4.2.1). The rightmost subplot shows the mean accuracy as a function of a grid of settings of λ and γ. The left two subplots show the accuracy profile for λ setting γ to its best value and vice versa. We can consider the data term g alone by setting λ to zero. This performs worse than when we include the pairwise potential,

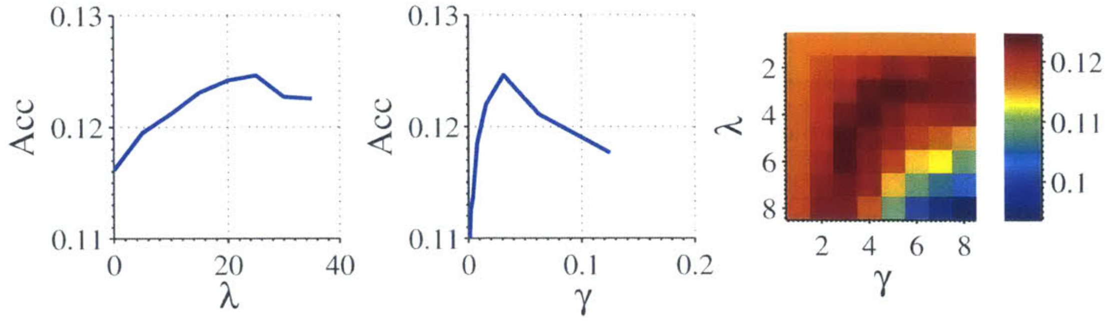


Figure 4.9: **Performance of CRF over various parameters:** We show the results of our CRF method on the collection parsing problem. The parameters λ and γ correspond to those in Equation 4.2. Note that the accuracy improves as we increase the weights on pairwise smoothing term.

demonstrating that parsing the collection holistically is better than treating each image individually.

Even though state classification per image is quite noisy, because each image collection contains many images, these noisy classifiers add up to give fairly accurate characterizations of entire image collections, as demonstrated by the success of discovering the relevant transformations in the collection (Section 4.4.1).

■ 4.4.3 Ranking images by transformation

We also evaluated how well we perform at ranking images from $ant(A)$ to A. As ground truth, we use the transformation annotations provided by human labelers (Section 4.1.3). We only consider images that fall in the Mid-A section. Our method achieves $\bar{\rho} = 0.46$. We visualize the rankings in Section 4.4.5. There is ample room for improvement on this difficult task, which we hope will inspire future work.

■ 4.4.4 How well does our system generalize across dissimilar noun classes?

Our tasks are all about transferring knowledge from a training set of noun classes to novel nouns. Sometimes this task is very easy: transferring from *laptop* to *computer*

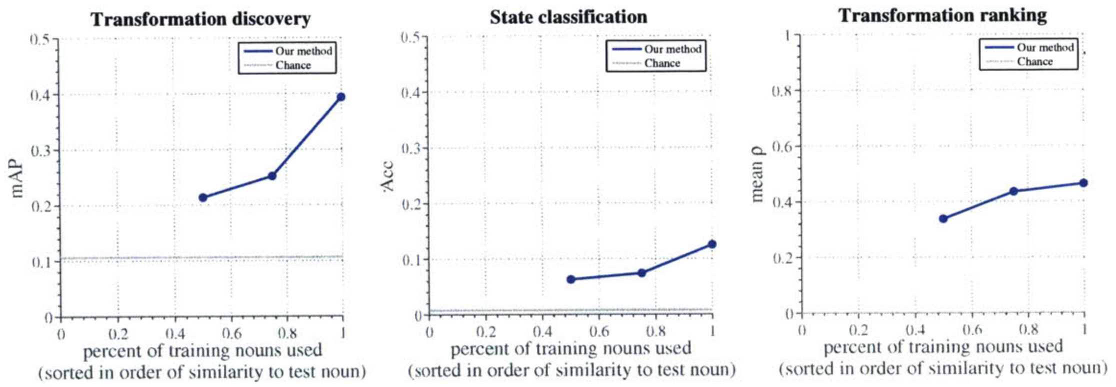


Figure 4.10: Performance versus percent of training nouns used, taking nouns in order of semantic similarity to test noun. Performance increases as more similar nouns are included in the training set, but is still well above chance even when the training set only includes dissimilar nouns (chance = gray line, estimated by assigning random scores to images).

might not require much generalization. To test how well our method generalizes, we restricted our training set, for each query noun, to only include nouns that a certain semantic distance from the query noun. As in Figure 4.8(b), we again use semantic similarity scores obtained from [37]. In Figure 4.10, we plot how performance increases, on each of our tasks, as the training set grows to include more and more nouns that are semantically similar to the query noun. Clearly, including synonyms helps, but performance is still well above chance even when the training set only contains nouns quite distinct from the query noun: our system can generalize state transformations over fairly dissimilar noun classes.

We visualize the effect of removing semantically similar nouns in Figure 4.11. Here we show ranked transformations for *old↔new computer* and *cluttered↔empty room*, using the visualization method described in Section 4.4.5. As we restrict the training set to include fewer and fewer similar nouns, the qualitative results degrade, as did the quantitative results. However, some classes generalize better than others. For example, *old↔new computer* may rely on having very similar examples in the training set (in

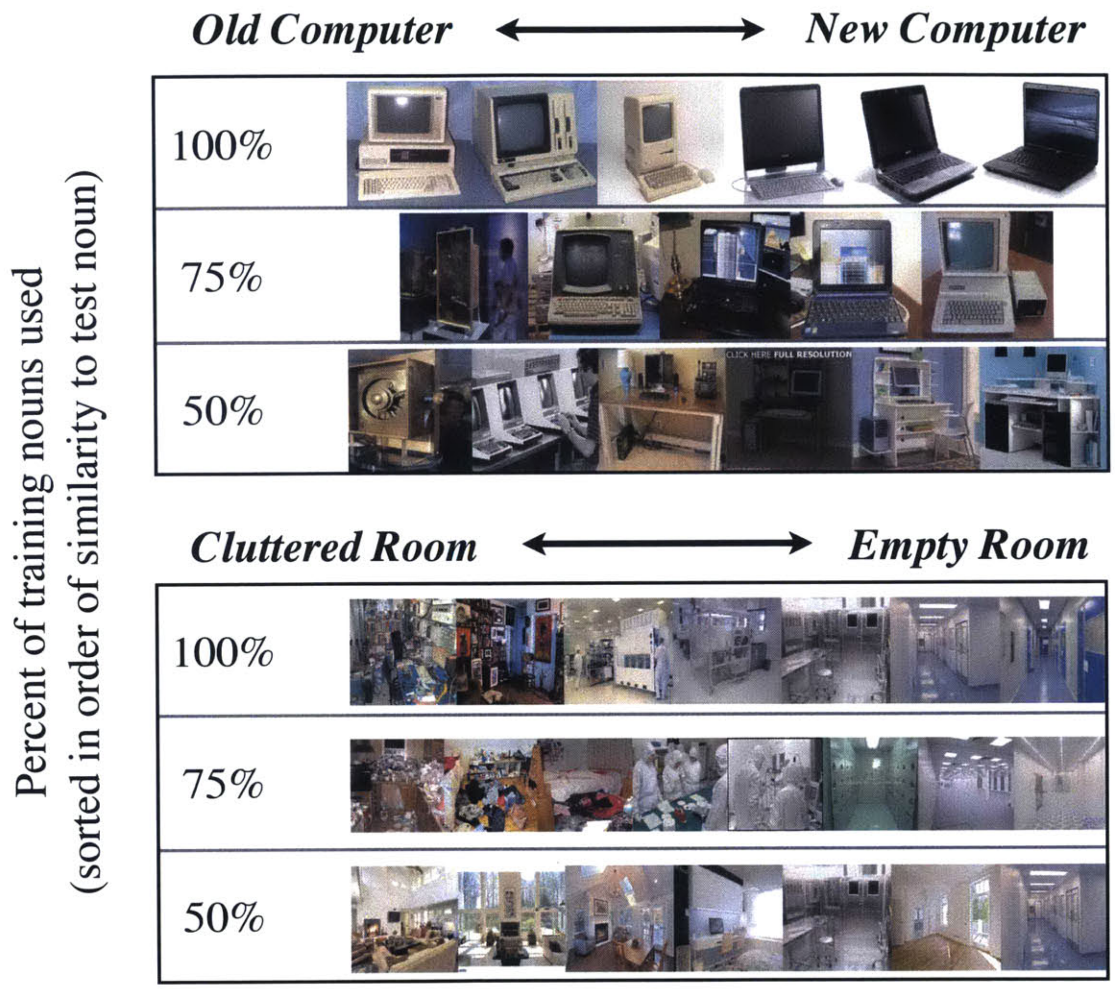

Old Computer
New Computer
Cluttered Room
Empty Room
100%
75%
50%
100%
75%
50%
Percent of training nouns used
(sorted in order of similarity to test noun)

Figure 4.11: Some classes generalize poorly, others generalize better. In this example we visualize how performance degrades as semantically similar nouns are removed from the training sets for *computer* and *room* (visualized using the "transformation taxi" method from Section 4.4.5). Notice that *old↔new computer* degrades rapidly whereas *cluttered↔empty* may be more easily generalizable across dissimilar noun classes.

particular, *old↔new laptop*) in order perform well; removing *laptop* from the training set has a big effect. On the other hand, many classes can undergo the transformation *cluttered↔empty* and this transformation tends to look alike between classes: a busy textural scene becomes flat and homogeneous in appearance. Correspondingly *cluttered↔empty room* is less rapidly affected by removing similar nouns from the the

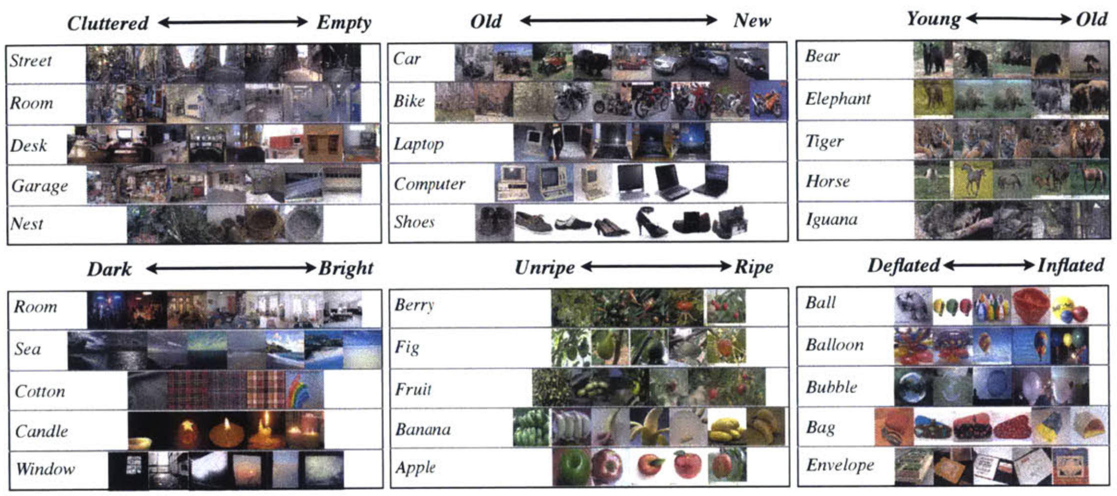


Figure 4.12: **Example transformation visualizations:** each transformation was discovered from the image collection of a noun class that was not included in the algorithm's training.

training set.

■ 4.4.5 Visualizing transformations

A transformation can be visualized by finding a smooth sequence of images from a starting state (A) to a transformed ending state ($ant(A)$). We use a method similar to "image taxis" [51].

First, we convert the input image collection to a graph. Each image is connected to its k-nearest neighbors in feature space ($k = 5$ in our implementation). For adjective A, we find a path $\mathcal{P}$ through the graph that optimizes the following cost function:

$$\arg\min_{\mathcal{P}} g(A|I_s) + g(ant(A)|I_t) + \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|} \|\mathbf{f}_{\mathcal{P}_i} - \mathbf{f}_{\mathcal{P}_{i+1}}\|_1, \tag{4.6}$$

where $g(\cdot)$ is given by Equation 4.4, s is the starting node of $\mathcal{P}$, and t is the ending node. For features we use our adjective classifier scores: $\mathbf{f}_i = \{f_j(I_i)\}_{j=1}^{|\mathcal{A}|}$ and $f_j(I_i) = g(\mathcal{A}_j|I_i)$. In addition, we multiply the feature channel for A and $ant(A)$ by a constant (20 in our implementation) in order to encourage smoothness most of all in that channel. This cost function says: 1) starting image should be A, 2) ending image should be highly $ant(A)$, and 3) path should be smooth in feature space. This cost can be optimized efficiently using Djistra's algorithm. As an additional constraint we only consider values of s among the top 5 images according to $g(A|I_s)$, and t among the top 5 images according to $g(ant(A)|I_t)$.

Example transformation visualizations are shown in Figure 4.12. Here we show several transformations each for several noun classes. For simple color and transformations, such as *dark↔bright* and *cluttered↔empty*, the method is reasonably effective. For geometric transformations, such as *deflated↔inflated*, the results are much worse. Future work should focus on capturing these difficult types of transformations.

4.5 Conclusion

In this chapter, we have introduced the novel problem of discovering and characterizing states and transformations in image collections. We have shown that simple yet powerful techniques are sufficient to make progress on this problem. We will publicly release our dataset and code to promote further work on this difficult problem.

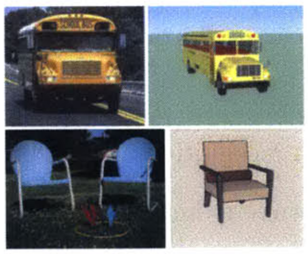
Chapter 5

Conclusion

In this dissertation, I focused on building computer systems to visually understand objects. Object understanding is a complex problem, because it consists of multiple sub-topics. We need to be able to interpret objects' 3D and shapes, their state and transformation, as well as their physics and interaction with the outer environment (e.g. other objects, human, and world). In this thesis, I mainly focused on 3D pose estiamtion of objects and understanding their states and transformaitons. For 3D pose estimation, I used the explict geometric information from 3D models. For understanding states and transformations, I utilized the collection of images of the target class as whole to learn the knowledge about them. These are yet a small subset toward object understanding. I however hope that this thesis will inspire other researchers to get one step closer to eventually solving "object understanding".

Appendix A

Analysis of Object Detectors trained with Rendered Images


(a) Real and synthetic objects

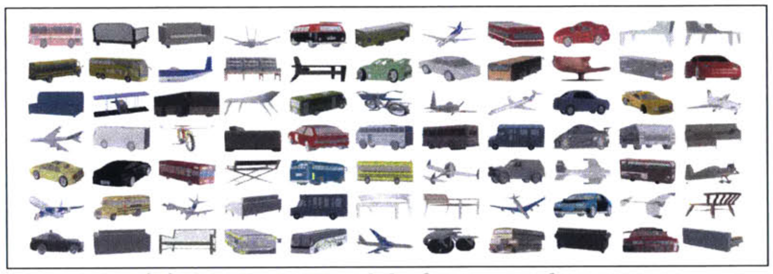

(b) Synthetic models from our dataset

Figure A.1: Our goal is to train a detector with synthetic 3D models. However, training a detector with rendered images of synthetic 3D models has several issues. In this appendix, we will analyze the key missing information from synthetic models.

Object detection models traditionally have been trained using large numbers of images, annotated with additional information such as bounding boxes and segmentation masks. While there is a large quantity of data in these datasets, the quality and richness of any given object instance is inherently limited. It's challenging, for example, to predict what an object instance would look like from another viewing direction, what its precise shape is, or how its appearance changes under other lighting conditions. In

This appendix is from selected relevant part of unpublished joint work with Andrew Owens and Antonio Torrlaba.

response to these limitations, recent work has trained using synthetic 3D models instead. These virtual models have the advantage of containing complete meta-data for an object (e.g. pose and dense geometry), and they make it possible to synthesize new training examples from arbitrary viewpoints and scene configurations. This has led to success in fine-pose detection [63], 3D viewpoint estimation [80], pedestrian detection [81], and 3D scene understanding [85].

Despite these advantages, images of synthetic models differ from real images in many important ways (see Figure A.1), and this makes it challenging to use them for training. For example, training a state-of-the-art object detector [20] with synthetic positive examples instead of positives from real images (perhaps surprisingly) results in very poor performance (Section A.2). The resulting detectors are biased by the virtual examples that they were trained on and fail to generalize to real-world variations in texture and backgrounds and visibility.

Despite the widespread use of synthetic object models in recognition, the differences between real and synthetic images and the trade-offs between them have not been carefully studied. In many ways, training with synthetic data shifts the challenges of the usual real-world datasets. Rather than using high quality images and low quality meta-data, as is done in the traditional regime, training with virtual models means using low-quality images and high-quality meta-data.

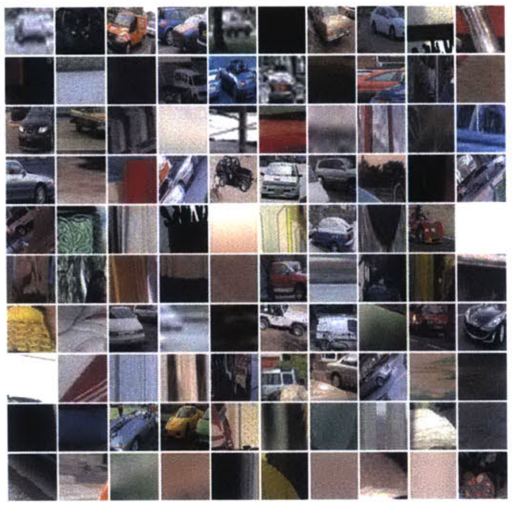
The underlying problem we are addressing is one of dataset bias, an issue that has been studied more broadly in vision [82, 98]. Here our training distribution (synthetic images) differs significantly from the test distribution (real images), which makes generalization difficult. Liebelt et. al. [60] trained detectors from textured synthetic models using a novel feature map and voting scheme, and [61] learned geometric models from synthetic examples. Marin et. al. [70] and Pishchulin et. al. [81] addressed a pedestrian detection problem using CVC dataset [70]. They however rendered pedestrian models in a realistic environment (using a game engine).

In this appendix, we identify several factors that degrade the performance of recognition models trained using synthetic data, and we show experiments how the impact of these factors can be reduced. To do this, we introduce a large dataset of synthetic object objects. We then evaluate the contribution of each factor by re-rendering the 3D models and applying a transformation to each example designed to mitigate the effect of the factor; we then train an object detector using the synthetic positive examples, and test it on PASCAL VOC [18]. We find several factors that make synthetic objects challenging to train with: the lack of realistic backgrounds; the mean and covariance of the image features; the choice of detector; and the choice of viewpoint.

■ A.1 Synthetic Dataset

Although there have been several works [63, 80, 81] using synthetic images, there is no public dataset of synthetic object images that is specifically designed for object detection. Hence, we collected a dataset of 3D object models from Google 3D Warehouse [31], a website where users upload and share 3D models they created. These models, some of which are shown in Figure A.1b, are free to download and vary widely in quality. We searched for object categories and manually pruned the models that are mislabeled or which contain multiple objects. We chose six categories corresponding to rigid and medium-to-large-sized objects that appear in the PASCAL VOC 2007 dataset [18], namely *aeroplane*, *bicycle*, *bus*, *car*, *chair*, and *sofa*. We excluded categories such as *person* and *cat*, where we expected the low-cost models to be highly unrealistic. For each class, we collected about 1000+ models.

For rendering, we used Blender 2.6 with the default setting except the light source was chosen as *hemi*. We rendered at various views (32 azimuth $\times$ 8 altitude $\times$ 3 tilts) at one fixed distance relative to the size of an object. While we collected many 3D models,


(a) Class A

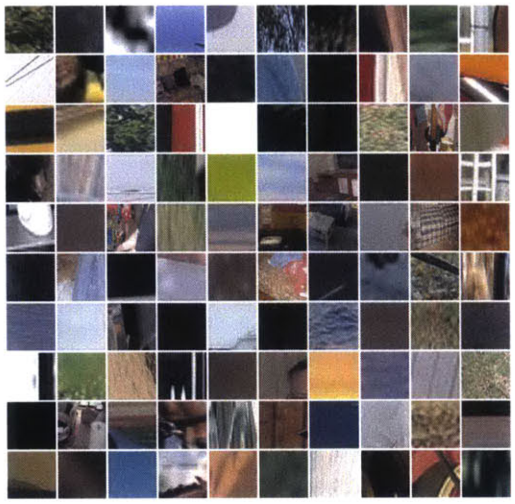

(b) Class B

Figure A.2: **Top detection results:** Could you get what the classes of (a) and (b) are? As shown, the top detection results are largely focused on clean, textureless image patches, which are common in synthetic images. We trained with synthetic (a) car and (b) sofa images.

we picked 100 random models from our dataset for rendering and later analysis.

■ A.2 Analysis of Training with Synthetic Data

With the recent popularity of using 3D CAD models for recognition problems [2, 23, 63, 80, 85], it is natural to ask how well would an off-the-shelf object detection system perform if it were given synthetic examples during training and real images during testing? The answer is: not very well! In Figure A.2ab, we show the 100 most confident detection windows in PASCAL VOC 2007 for two object categories – each was trained using an off-the-shelf detection system [20] using synthetic examples similar to those shown in Figure A.1. Can you guess what the object categories were? (See the caption for the answer.)

These detectors also perform poorly on quantitative metrics. For example, the difference in average precision between real images and these synthetic images is 20.4

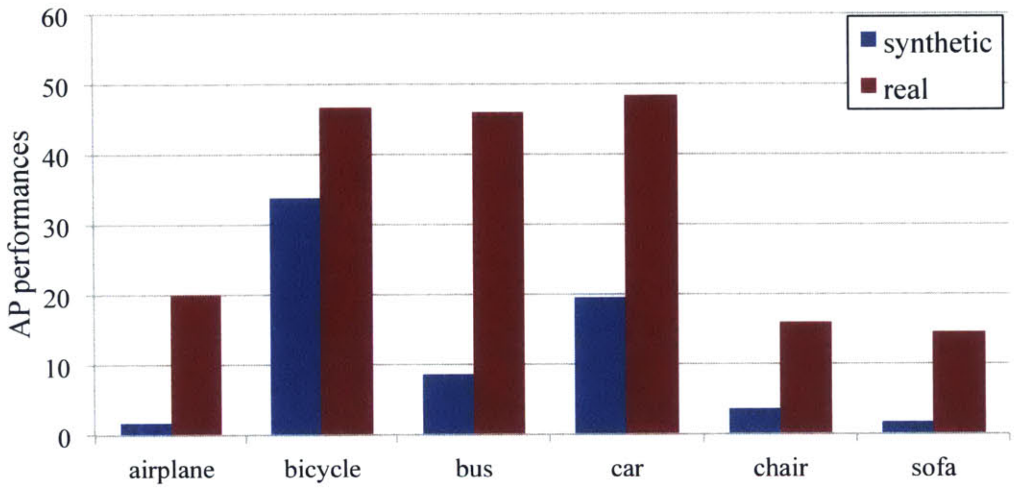


Figure A.3: **Detection performance on PASCAL 2007:** we compare the detection performance of an off-the-shelf detector [20]. We trained with three mixtures with parts using synthetic and real images. Note the large difference that comes from training with synthetic versus real images.

(Figure A.3) – a dramatic example of dataset bias [98]. We now turn our attention to the issues that lead to this gap in performance.

■ A.2.1 Experiment design

Now that we have seen, empirically, that models trained with synthetic examples have poor generalization ability, we consider the qualitative differences between synthetic and real images, such as those shown in Figure A.1. How much does it matter that the synthetic models are not in front of realistic backgrounds, and that the texture does not exactly match images with real-world materials and lighting? How do we deal with variations in viewpoint when we don't have real images of these objects and thus do not know the distribution of viewpoints?

We consider these questions using a *diagnosis by detection* approach. For each possible factor, we introduce a way that its influence can be mitigated; we then train a detector using this new method, and we verify that it improves performance on detection benchmarks. We now describe the procedures that we use to perform this analysis.

Detection algorithm The detectors we examine in this work are based on sliding windows with HOG [13] features. Unless otherwise noted, we train synthetic detectors using a deformable parts model (DPM) [20] with parts disabled, and with 200 training examples per mixture component. There are also several recent variations on this object-detection paradigm, such as the Exemplar-SVM (E-SVM) [69], and Exemplar-LDA (E-LDA) [38] which we examine as well.

Evaluation To evaluate the effectiveness of different object detection models, we use the PASCAL VOC 2007 [18] dataset. This dataset contains real-world images of 20 different object categories (varying from *airplane* to *tv monitor*). Since meta-data transfer is an important use case for synthetic training examples, we also evaluate localization and viewpoint estimation using the 3DObject dataset [105]. This dataset contains 8 different object classes taken from multiple canonical views. For the localization metric, the average precision score is used, and the viewpoint estimation is computed using the confusion matrix only on the correctly detected objects. Since the models we evaluate are based on synthetic images, we do not use the training set provided by this dataset. For the overall analysis, we tested on 6 different categories (*airplane*, *car*, *bus*, *bicycle*, *chair*, and *sofa*), of which 3D models are widely available and are more rigid (without large deformable pose variations).

■ A.2.2 Image Background

One of notable differences between real and synthetic images is the lack of a background in the latter (*c.f.* Figure A.4(a) and (d)).

In order to verify the general importance of having a background, we first trained detectors on *real* images but removed the background, setting to a constant value all

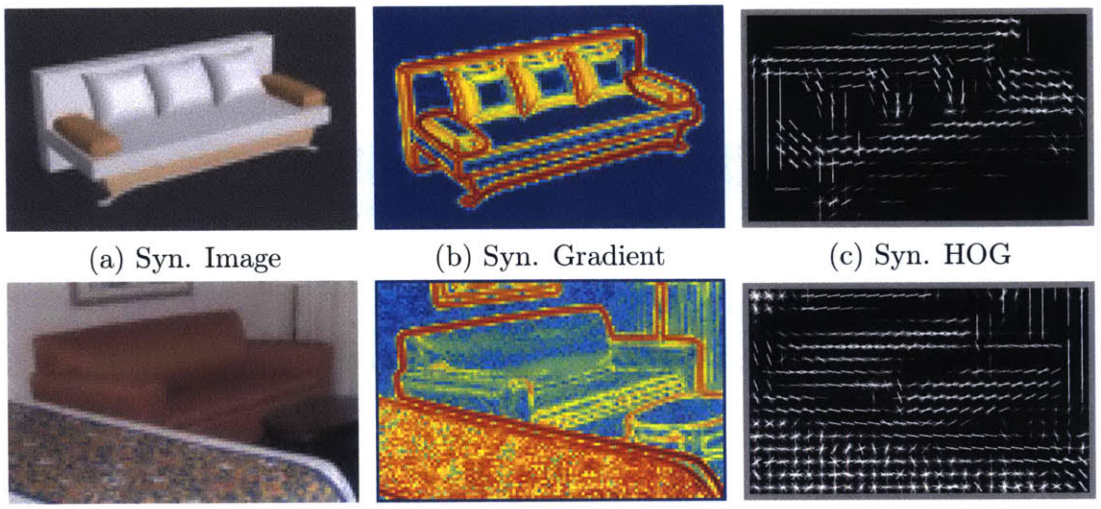


Figure A.4: **Comparing real and synthetic images:** These images show the differences between synthetic and real images. Synthetic images are missing background, texture, and clutter.

	aeroplane	bicycle	bus	car	chair
Real−BG	5.5	35.8	30.7	22.7	1.3
Real−BG +RandBG	17.4	48.5	37.5	40.2	11.1
Real	20.1	46.7	45.9	48.3	15.9
Synthetic	1.8	33.9	8.5	19.6	3.7
Synthetic + RandBG	4.3	36.3	10.1	23.8	4.3

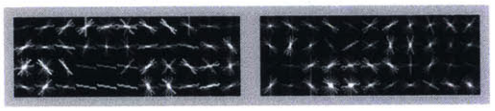
Table A.1: **Importance of Background:** We compare three scenarios where we differ the training examples by using (1) synthetic or real images, (2) background-subtracted real images (Real−BG), (3) inserting a random background to images (+RandBG). When we remove background from real images, the detection performance drops significantly. However, it becomes closer to Real by adding a random image patch as the background.

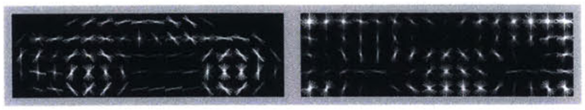
of the pixels outside of a segmentation mask (provided by PASCAL VOC 2011). As shown in Table A.1, this model (called *Real-BG*) significantly underperformed the original examples (*Real*), and the performance is closer to that of the synthetic images (*Synthetic*).

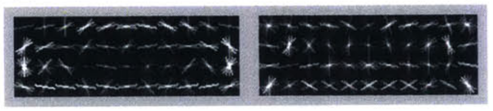
Similarly, if we render the synthetic models in front of blank backgrounds, then

(a) Real car images

(b) Synthetic car images


(c) SVM model trained with real images


(d) SVM model trained with synthetic images


(e) LDA model trained with real images

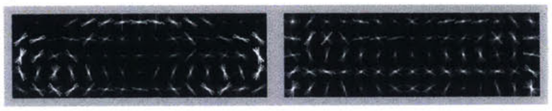

(f) LDA model trained with synthetic images

Figure A.5: **Real and synthetic images of cars and their learned detectors**: comparing (c) and (d), and (e) and (f) show that detectors trained on synthetic images have large negative values on the background cells. (left column: positive weights, right column: negative weights). Note that the HOG templates differ in size because we allowed the DPM training procedure to select the template shapes automatically.

the result is also poor. In our case, the HOG features have a value close to 0 near a blank background, while real images are more likely to have non-zero gradients in the same locations. As a result, classifiers learn to negatively score gradients outside of the object boundaries. This can be observed in Figure A.5. With real images, the classifiers learn both positive and negative values in HOG background cells (Figure A.5ce), but these same weights all become strongly negative when the same model is trained with synthetic images (Figure A.5df).

To analyze this problem, we used a similar approach to that of [81], which composited photo-realistic pedestrian models with random backgrounds from images of urban scenes. Here we composite each synthetic image with a random (real) image patch in the background, based on the assumption that object positions are uniformly distributed across all images. This results in the background pixels having natural image statistics (similar to real negatives), and hence the algorithm focuses on the foreground area more. The result showed a clear boost in performance (in Table A.1). We can

Split Method	airplane	bicycle	bus	car	chair	sofa	mean
Aspect Ratio	2.9	39.2	15.6	25.0	9.4	4.1	11.4
Viewpoint	6.1	45.9	19.5	31.3	9.7	4.9	19.6
Real	15.1	45.7	38.1	42.3	11.9	8.6	27.0

Table A.2: **Viewpoint:** We show the average precision on PASCAL 2007 when mixture components correspond to clusters based on aspect ratio [20] and viewpoint (Section A.2.3), and also real images with aspect-ratio clustering.

conclude that missing background affects performances and we need a better algorithm that can either borrow or generate realistic background.

A.2.3 Viewpoint

When training with real images, the classifier implicitly learns the distribution of viewpoints (e.g. the rotation angles) but this information is missing from synthetic examples. On the other hand, one of the benefits of training with synthetic 3D models is that we can synthesize an image of an object from any viewpoint.

While it is common to cluster real images by aspect ratio [20], recent work [114] has shown that forming mixture models based on the object's viewpoint can improve performance. Furthermore, it is unclear whether the aspect ratio grouping method is well suited to the case of synthetic models, where the viewpoint distribution is unknown. Thus to analyze whether viewpoint-clustering models are useful for synthetic examples, we form mixture models based on 3D viewpoint. As described in Section A.1, each rendered example was parameterized by a discrete azimuth and altitude parameter, and to form mixture components we group examples by azimuth-altitude pair.

We found that this example-grouping scheme significantly improves performance over clustering by aspect ratio (Table A.2).

■ A.3 Conclusion

Training a detector with synthetic images to test on real images means dealing with a significant form of dataset bias. In this appendix, we diagnosed several difficulties and their effects that are uniquely derived from synthetic images.

Bibliography

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80, 2010.

[2] Mathieu Aubry, Daniel Maturana, Alexei Efros, Bryan Russell, and Josef Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[3] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[4] Tamara L Berg, Alexander C Berg, and Jonathan Shih. Automatic attribute discovery and characterization from noisy web data. In *European Conference on Computer Vision*, pages 663–676. Springer, 2010.

[5] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[6] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 23(11):1222–1239, 2001.

[7] Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. NEIL: Extracting Visual Knowledge from Web Data. In *IEEE International Conference on Computer Vision*, 2013.

[8] Jonghyun Choi, M. Rastegari, A. Farhadi, and L.S. Davis. Adding unlabeled samples to categories by learned attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 875–882, June 2013.

[9] Myung Jin Choi, Joseph J. Lim, Antonio Torralba, and Alan S. Willsky. Exploiting hierarchical context on a large database of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[10] W. Choi, Y. W. Chao, C. Pantofaru, and S. Savarese. Understanding indoor scenes using 3d geometric phrases. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[11] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 23(6):681–685, 2001.

[12] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, June 2005.

[13] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[14] J. Deng, W. Dong, R. Socher, L.J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[15] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*, 2014.

[16] S.K. Divvala, D. Hoiem, J.H. Hays, A.A. Efros, and M. Hebert. An empirical study of context in object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2009.

[17] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv preprint arXiv:1310.1531*, 2013.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results.

[19] Ali Farhadi, Ian Endres, Derek Hoiem, and David Forsyth. Describing objects by their attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[20] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4.

[21] Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[22] PedroF. Felzenszwalb and DanielP. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[23] Sanja Fidler, Sven J. Dickinson, and Raquel Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *Advances in Neural Information Processing Systems*, 2012.

[24] Matthew Fisher and Pat Hanrahan. Context-based search for 3d models. *ACM Trans. Graph.*, 29(6), December 2010.

[25] David F. Fouhey, Vincent Delaitre, Abhinav Gupta, Alexei A. Efros, Ivan Laptev, and Josef Sivic. People watching: Human actions as a cue for single-view geometry. In *European Conference on Computer Vision*, 2012.

[26] Carolina Galleguillos, Brian McFee, Serge Belongie, and Gert R. G. Lanckriet. Multi-class object localization by combining local contextual interactions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[28] Ross Girshick. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015.

[29] Ross Girshick, Hyun Oh Song, and Trevor Darrell. Discriminatively activated sparselets. In *International Conference on Machine Learning*, 2013.

[30] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[31] Google. Google 3d warehouse.

[32] Chunhui Gu, Joseph J. Lim, Pablo Arbelez, and Jitendra Malik. Recognition using regions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[33] Abhinav Gupta. Beyond nouns and verbs, 2009.

[34] Abhinav Gupta and Larry S Davis. Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *European Conference on Computer Vision*, pages 16–29. Springer, 2008.

[35] Abhinav Gupta, Scott Satkin, Alexei A. Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[36] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from RGB-D images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[37] Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. Umbc ebiquity-core: Semantic textual similarity systems. *Proceedings of the Second Joint Conference on Lexical and Computational Semantics*, 2013.

[38] Bharath Hariharan, Jitendra Malik, and Deva Ramanan. Discriminative decorrelation for clustering and classification. In *European Conference on Computer Vision*, 2012.

[39] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[40] Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking inside the box: using appearance models and context based on room geometry. In *European Conference on Computer Vision*, pages 224–237, 2010.

[41] Mohsen Hejrati and Deva Ramanan. Analyzing 3d objects in cluttered images. In *Advances in Neural Information Processing Systems*, 2012.

[42] Mohsen Hejrati and Deva Ramanan. Analysis by synthesis: 3d object recognition by object reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[43] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments. In *RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*, 2010.

[44] D. Hoeim, C. Rother, and J. Winn. 3d layoutcrf for multi-view object class recognition and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, US, 2007.

[45] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *IEEE International Conference on Computer Vision*, pages 654–661, 2005.

[46] Derek Hoiem, Varsha Hedau, and David Forsyth. Recovering free space of indoor scenes from a single image. *IEEE Conference on Computer Vision and Pattern Recognition*, 0:2807–2814, 2012.

[47] Phillip Isola, Joseph J. Lim, and Edward H. Adelson. Discovering states and transformations in image collections. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[48] Zhaoyin Jia, Andy Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d-based reasoning with blocks, support, and stability. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[49] Michael J Jones and Tomaso Poggio. Multidimensional morphable models: A framework for representing and matching object classes. *International Journal of Computer Vision*, 29(2):107–131, 1998.

[50] Joseph J. Lim, Pablo Arbeláez, Chunhui Gu, and Jitendra Malik. Context by region ancestry. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[51] Biliana Kaneva, Josef Sivic, Antonio Torralba, Shai Avidan, and William T Freeman. Infinite images: Creating and exploring a large photorealistic virtual space. *Proceedings of the IEEE*, 98(8):1391–1407, 2010.

[52] Kevin Karsch, Zicheng Liao, Jason Rock, Jonathan T. Barron, and Derek Hoiem. Boundary cues for 3d object shape recovery. *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[53] Gunhee Kim and Eric P. Xing. Reconstructing Storyline Graphs for Image Recommendation from Web Community Photos. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

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

[55] Adriana Kovashka, Devi Parikh, and Kristen Grauman. Whittlesearch: Image Search with Relative Attribute Feedback. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2012.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[57] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 33(4), 2014.

[58] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3d scenes. In *IEEE International Conference on on Robotics and Automation*, 2012.

[59] Yong Jae Lee, Alexei A. Efros, and Martial Hebert. Style-aware mid-level representation for discovering visual connections in space and time. In *IEEE International Conference on Computer Vision*, December 2013.

[60] J. Liebelt, C. Schmid, and K. Schertler. Viewpoint-independent object class detection using 3d feature maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[61] Jörg Liebelt and Cordelia Schmid. Multi-View Object Class Detection with a 3D Geometric Model. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[62] Joseph J. Lim, Ruslan Salakhutdinov, and Antonio Torralba. Transfer learning by borrowing examples for multiclass object detection. In *Advances in Neural Information Processing Systems*, 2011.

[63] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing ikea objects: Fine pose estimation. In *IEEE International Conference on Computer Vision*, 2013.

[64] Joseph J. Lim, C. Lawrence Zitnick, and Piotr Dollar. Sketch tokens: A learned mid-level representation for contour and object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[65] Joseph J. Lim, Aditya Khosla, and Antonio Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *European Conference on Computer Vision*, 2014.

[66] T.Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, 2014.

[67] D.G. Lowe. Fitting parameterized three-dimensional models to images. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 1991.

[68] Tomasz Malisiewicz and Alexei A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *Advances in Neural Information Processing Systems*, December 2009.

[69] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *IEEE International Conference on Computer Vision*, 2011.

[70] J. Marin, D. Vázquez, D. Gerónimo, and A.M. López. Learning appearance in virtual scenarios for pedestrian detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[71] Kevin Matzen and Noah Snavely. Nyc3dcars: A dataset of 3d vehicles in geographic context. In *IEEE International Conference on Computer Vision*, 2013.

[72] Hossein Mobahi, Ce Liu, and William T. Freeman. A compositional model for low-dimensional image set representation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[73] Joseph L. Mundy. Object recognition in the geometric era: A retrospective. In *Toward CategoryLevel Object Recognition, volume 4170 of Lecture Notes in Computer Science*, pages 3–29. Springer, 2006.

[74] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, 2012.

[75] T. Ojala, M. Pietikinen, and T. Menp. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 2002.

[76] Devi Parikh and Kristen Grauman. Relative attributes. In *IEEE International Conference on Computer Vision*, 2011.

[77] Genevieve Patterson and James Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[78] Genevieve Patterson, Chen Xu, Hang Su, and James Hays. The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision*, 108(1-2):59–81, 2014.

[79] Bojan Pepik, Peter Gehler, Michael Stark, and Bernt Schiele. 3d2pm - 3d deformable part models. In *European Conference on Computer Vision*, 2012.

[80] Bojan Pepik, Michael Stark, Peter Gehler, and Bernt Schiele. Teaching 3d geometry to deformable part models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[81] Leonid Pishchulin, Arjun Jain, Christian Wojek, Mykhaylo Andriluka, Thorsten Thormaehlen, and Bernt Schiele. Learning people detection models from few training samples. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[82] J. Ponce, T. Berg, M. Everingham, D. Forsyth, M. Hebert, S. Lazebnik, M. Marszalek, C. Schmid, B. Russell, A. Torralba, et al. Dataset issues in object recognition. *Toward category-level object recognition*, 2006.

[83] Fred Rothganger, Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. *International Journal of Computer Vision*, 66:2006, 2006.

[84] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1), 2008.

[85] Scott Satkin, Jason Lin, and Martial Hebert. Data-driven scene understanding from 3D models. In *British Machine Vision Conference*, 2012.

[86] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 2000.

[87] A. G. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box In the Box: Joint 3D Layout and Object Reasoning from Single Images. In *IEEE International Conference. on Computer Vision*, 2013.

[88] Alex Schwing, Sanja Fidler, Marc Pollefeys, and Raquel Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *IEEE International Conference on Computer Vision*, 2013.

[89] Lavanya Sharan, Ce Liu, Ruth Rosenholtz, and Edward H Adelson. Recognizing materials using perceptually inspired features. *International Journal of Computer Vision*, 103(3):348–371, 2013.

[90] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[91] Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Constrained semi-supervised learning using attributes and comparative attributes. In *European Conference on Computer Vision*, 2012.

[92] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep Fisher networks for large-scale image classification. In *Advances in Neural Information Processing Systems*, 2013.

[93] Erik B. Sudderth, Antonio Torralba, William T. Freeman, and Alan S. Willsky. Describing visual scenes using transformed objects and parts. *International Journal of Computer Vision*, 77(1-3):291–330, May 2008.

[94] Min Sun, Hao Su, Silvio Savarese, and Li Fei-Fei. A multi-view probabilistic model for 3d object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[95] Kevin Tang, Vignesh Ramanathan, Li Fei-fei, and Daphne Koller. Shifting weights: Adapting object detectors from image to video. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 638–646. Curran Associates, Inc., 2012.

[96] Joshua B Tenenbaum and William T Freeman. Separating style and content with bilinear models. *Neural computation*, 12(6):1247–1283, 2000.

[97] Alexander Thomas, Vittorio Ferrar, Bastian Leibe, Tinne Tuytelaars, Bernt Schiel, and Luc Van Gool. Towards multi-view object class detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[98] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[99] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine intelligence*, 29(5), May 2007.

[100] J.R.R. Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 2013.

[101] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 2009.

[102] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, December 2001.

[103] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. Technical Report CNS-TR-201, Caltech, 2010. URL `http://www.vision.caltech.edu/visipedia/CUB-200.html`.

[104] Zhirong Wu, Shuran Song, Aditya Khosla, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition*, Boston, USA, June 2015.

[105] Yu Xiang and Silvio Savarese. Estimating the aspect layout of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[106] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[107] Jianxiong Xiao, Bryan Russell, and Antonio Torralba. Localizing 3d cuboids in single-view images. In *Advances in Neural Information Processing Systems*, 2012.

[108] Bangpeng Yao, Jiayuan Ma, and Li Fei-Fei. Discovering object functionality. In *IEEE International Conference on Computer Vision*, December 2013.

[109] Yibiao Zhao and Song-Chun Zhu. Image parsing via stochastic scene grammar. In *Advances in Neural Information Processing Systems*, 2011.

[110] Yibiao Zhao and Song-Chun Zhu. Scene parsing by integrating function, geometry and appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[111] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning Deep Features for Scene Recognition using Places Database. *Advances in Neural Information Processing Systems*, 2014.

[112] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *International Conference on Learning Representations (ICLR)*, 2015.

[113] Tinghui Zhou, Yong Jae Lee, Stella X. Yu, and Alyosha A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. June 2015.

[114] Xiangxin Zhu, Carl Vondrick, Deva Ramanan, and Charless C. Fowlkes. Do we need more training data or better models for object detection? In *British Machine Vision Conference*, 2012.

[115] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *European Conference on Computer Vision*, 2014.

[116] M.Z. Zia, M. Stark, and K. Schindler. Explicit occlusion modeling for 3d object class representations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2013.