2D and 3D Models for Object and Scene Understanding

by

Min Sun

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering- Systems)
in The University of Michigan
2012

Doctoral Committee:

Assistant Professor Silvio Savarese, Chair
Professor Alfred O. Hero III
Professor Benjamin Kuipers
Associate Professor Fei-Fei Li, Stanford University

DEDICATION

This doctoral dissertation is dedicated to my ever supportive parents, loving wife and daughter.

ACKNOWLEDGEMENTS

I am very fortunate to work closely with both Professor Silvio Savarese and Professor Fei-Fei Li. They are both brilliant scientists and incredibly knowledgeable mentors. Through the course of my PhD, they have helped me evolve from a clueless student to a confident researcher. I truly appreciate their guidances and encouragements. They have also taught my a lot on how to effectively communicate with other researchers: polishing research papers, and making presentation crystal clear and intuitive to follow. Most importantly, I kept on being amazed by their insights on high-impact future research directions, even after 5 years of study with them.

I am also very lucky to study in both Princeton university and university of Michigan. At Princeton university, I have met many great professors. I want to thank them for teaching me the fundamental knowledge of machine learning and computer algorithms that has become essential for my thesis work: professor David Blei, for introducing me the probabilistic graphical models and Bayesian nonparametric methods; professor Robert Schapire, for showing me the power of boosting algorithms and many potential applications; professor Robert Tarjan, for helping me appreciate the beauty of computer algorithms. I also would like to thank my colleagues who I enjoyed hanging out and learned from: Chong Wang, for being a great neighbour and always being able to answer my questions in all fields; Jia Li, for being my role model of an outstanding graduate student and sharing her favorite Sichuan food; Juan Carlos Niebles, for burning the midnight oil with me during my first CVPR deadline and sharing his favorite columbia food; Hao Su, for teaching me how to organize research

codes and working hard with me on my first project; Barry Chai, for helping design the user interface on Amazon Mechanical Turk and always cheer me up with his funny jokes.

At university of Michigan, I have the great honor to have professor Alfred Hero and Benjamin Kuipers on my thesis committee. I want to thank them for being very supportive and giving me insightful suggestions from the perspective of signal progressing and robotics. I also want to thank professor Honglak Lee for sharing his experience on writing machining learning papers and encouraging me to develop general algorithms that could create impacts in many fields. My transition from Princeton to Michigan cannot be smoother thanks to our wonderful student coordinator, Becky Turanski, who gave me tremendous supports and the warmest welcome to me and my family. I also treasure the fellowship with my colleagues who consider me as a true wolverine the day when I arrived. I especially appreciate all the helps and great times spent with all the members in the vision lab. In particular, I would like to thank: Wongun Choi, for sharing his knowledge on processing video and designing sampling algorithms; Byung-soo Kim, for working hard with me on my last project and always answering my questions on how to be a good father and husband; Yingze Bao, for being the best collaborator ever who helped me achieve a very productive first year; Shyam Kumar, for working hard with me on the mobile computing project; Murali Telaprolu, for all the good times working on the branch-and-bound project together.

During my PhD, I was really blessed to have many opportunities to interact with outstanding researchers outside my immediate academic group. I want to thank Dr. Gary Bradski for showing me how influential a person can be by dedicating on open source projects. I also appreciate Dr. Pushmeet Kohli and Jamie Shotton for getting me involved with a fascinating project which has high potential to be used in a future product.

There are many more people I would like to thank who are outside my research

community but gave me significant supports. In particular, I would like to thank my brothers and sisters in the church at Princeton and Ann Arbor. Almost every Friday night, I was looking forward to the fellowship meeting which helps me truly escape from the up and down in school and research. I also would like to thank my fellow Taiwanese student couples in Princeton and Ann Arbor. Thanks for making my five years in the US the best part of my life.

The truth is, I won't be able to accomplish all these without the supports from my family. I sincerely thank my wife's parents for believing in a young man like me for taking care of their precious daughter in a foreign country. I am also truly grateful that my parents give me unconditional supports and embrace me whenever I feel defeated. Most importantly, I want to thank my wife for everything that she has done to make me a better man in the world. Last but not least, I am grateful for god giving me a lovely daughter whose cute giggles simply take away all my pressure.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF ALGORITHMS

Algorithm

ABSTRACT

2D and 3D Models for Object and Scene Understanding

by

Min Sun

Chair: Silvio Savarese

In this thesis, we propose novel 2D and 3D models for object and scene understanding from images. This is an extremely challenging problem in computer vision. Objects change their appearance because of intra-class variability, view point transformations and their inherent deformable nature. Understanding scenes is also challenging. Scenes may comprise large number of objects whose relationships are class specific and depend on the view point and 3D scene geometry. First, we propose object models that are capable of detecting generic rigid objects and simultaneously extracting their viewpoints and 3D shape. The ability of these models to jointly capture appearance and shape in a truly 3D sense makes them robust to intra-class variability, view point changes and occlusions. Second, we have focused on designing models that can effectively capture the appearance and shape variability of deformable objects such as humans or animals. Most importantly, we propose algorithmic solutions that are capable of "taming" the intrinsic complexity of the pose estimation problem while guaranteeing the optimality of the solution. Finally, we have proposed models that can effectively capture the interplay among objects and scene elements so as to simultaneously recognize the scene, detect objects and segment regions accurately and

efficiently.

CHAPTER I

Introduction

One of the key problems in computer vision is to automatically interpret the world from images. Humans do this effortlessly and robustly. By looking at the image in Fig. 1.1(a), we can recognize that there are objects such as cars, scooters, and people; there are generic scene elements such as the road surface and the buildings (Fig. 1.1(b)). Critically, we can do this regardless of: i) whether it is the first time we see a specific car model (e.g., the sedan manufactured by Mitsubishi); ii) whether a scene element is occluded by other objects or not (the buildings are occluded by cars and scooters); iii) which view point an object is observed from; iv) the pose of a person is holding (e.g., sitting on scooter);

Moreover, we can easily determine the key geometrical properties of the scene even when a single image is provided. These include the 3D pose of the objects (the car is observed from a back view; the person is facing front and is holding a "sit" pose); the 3D orientation of the scene elements (the buildings are observed from a 45 degrees view); we can estimate the depth ordering of the objects in the scene (the human outlined by the green polygon is closer to the camera than the car outlined by the red polygon) (Fig. 1.1(c)). Finally, we do not perceive objects as isolated entities in the scene but we rather account of the semantic and geometrical interplay between objects and scene elements (a scooter is roaming on a street) or between humans

Figure 1.1: Illustration of the research goals in this thesis. We would like to recognize all of the objects from an image of a scene such as the one illustrated in panel (a). Examples of objects of interest are indicated by color-coded regions (Panel (b)). They include rigid objects (e.g., cars and scooters), articulated objects (e.g., people), and generic scene elements (e.g., a street and buildings). Moreover, we would like to determine the properties associated to each element (Panel (c)) as well as typical relationships between objects and generic scene elements observed in the real-world (Panel (d)). All of these together enable methods for coherent interpretation of the scene.

and objects (a person is riding a scooter) (Fig. 1.1(d)); Eventually we integrate all these pieces of semantic and geometrical evidence into a coherent interpretation of the scene which can tell us what this image is about: an urban street scene with scooters roaming around.

Figure 1.2: Panel (a,b,c) illustrate the intra-class variability, view point variability, and pose variability, respectively.

1.1 Challenges

Unfortunately, designing a computer vision system that is capable of achieving all the goals above is extremely challenging. We summarize the key challenges below.

Objects change their appearance because of intra-class variability – difference instances that belong to the same object category may present different photometric and shape properties as illustrated in Fig. 1.2(a). Moreover, even when a single object instance is considered, the object appearance is subject to dramatic changes because of view point transformations (Fig. 1.2(b)). Thus, a key challenge in object recognition is to design object models that account for both types of variability –

that is, are capable of: i) generalizing across specific object instances and recognizing object instances that may have not been seen in training before; ii) enabling view-invariant representations and recognizing object instances observed from generic view points.

Recognizing articulated objects such as humans or animals is even more challenging because of the deformable nature of their shape. Humans can change their appearance properties as they hold different poses (e.g., standing, sitting, jumping). Modelling such variability is difficult since the number of possible body part configurations can be extremely large. A model for articulated object should be ideally capable of compactly summarizing such pose variability and enable the recognition of object instances from poses that may have not been seen in training before.

Finally, understanding a scene from an image is also challenging. On top of all the challenges described above, one should also account for the fact that the interplay between objects and scene elements is difficult to model. A scene typically comprises of a large number of elements (humans, cars, side walk, pavement, as illustrated in Fig. 1.1(a)) whose relationships (e.g. "on-top" , "next-to", "behind") vary as function of the scene class and the objects class (i.e., cars lie on top of the street pavement). The interplay between objects can be complex and view dependent (i.e. the depth ordering of objects is typically ambiguous as the 3D-to-2D mapping is not reversible in general) and the number of types of relationships is in general very large. It is desirable to design models for scene understanding that are generic and capable of capturing a hypothetically exponentially large number of configurations of scene elements.

1.2 Previous Work

Unfortunately, after five decades of computer vision research and numerous very successful stories in recognition and reconstruction, most of these challenges are still

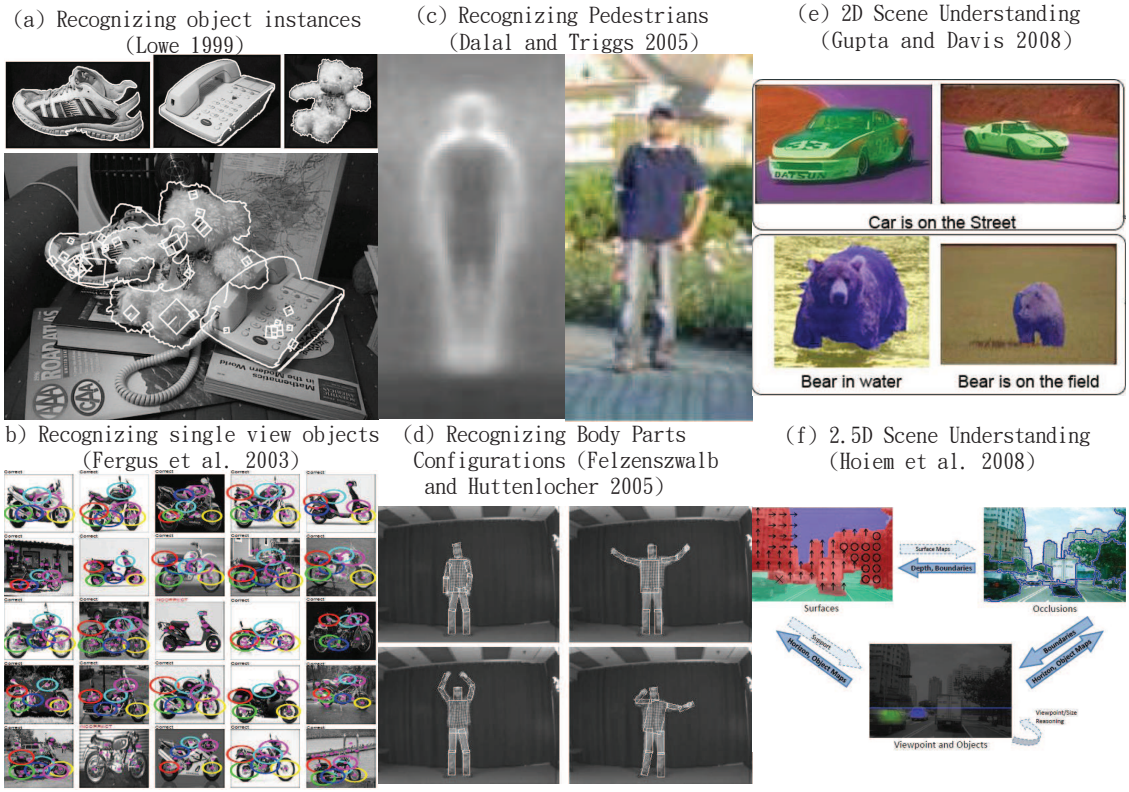


Figure 1.3: Panel (a,b) show methods recognizing object instances (*Lowe*, 1999) and single view objects (*Fergus et al.*, 2003), respectively. Panel (c,d) show methods recognizing pedestrians (*Dalal and Triggs*, 2005) and body parts configurations (*Felzenszwalb and Huttenlocher*, 2005), respectively. Panel (e,f) show methods incorporate 2D (*Gupta and Davis*, 2008a) and 2.5D (*Hoiem et al.*, 2008) relationships for scene understanding, respectively.

unsolved.

Most of the models for object recognition are either intrinsically view point invariant but cannot be generalized so as to recognize categories (object instances that are never seen in training) (*Lowe*, 1999; *Obdrzalek and Matas*, 2002; *Rothganger et al.*, 2003; *Brown and Lowe*, 2005; *Gordon and Lowe*, 2006) (Fig. 1.3(a)), or capture properties of an object category from a handful of viewpoints and cannot generalize well for any generic view point (*Weber et al.*, 2000b; *Borenstein and Ullman*, 2002; *Fergus et al.*, 2003; *Felzenszwalb and Huttenlocher*, 2005; *Grauman and Darrell*, 2005; *Lazebnik et al.*, 2006; *Felzenszwalb et al.*, 2008) (Fig. 1.3(b)).

Most of the models for recognizing articulated objects such as humans either focus

on recognizing a fix number of poses without aiming to localize body parts (*Dalal and Triggs*, 2005; *Felzenszwalb et al.*, 2008) (Fig. 1.3(c)) or assume that object location is given and seek to localize the body parts (*Felzenszwalb and Huttenlocher*, 2005; *Ramanan*, 2006; *Eichner and Ferrari*, 2009; *Sapp et al.*, 2010b) (Fig. 1.3(d)). Moreover, in order to reduce intrinsic complexity of the problem (explore an hypothesis space that is exponentially large) approximations on the inference procedure are typically made (*Jiang and Martin*, 2008; *Tran and Forsyth*, 2010; *Ren et al.*, 2005; *Wang et al.*, 2011).

Most of the models for scene understanding describe the interplay between objects and scene elements by only considering 2D spatial relationships in the image plane (e.g., nearby elements in the pixel domain should share the same class and property) (*Shotton et al.*, 2006; *Heitz and Koller*, 2008; *Gupta and Davis*, 2008a; *Desai et al.*, 2009; *Ladicky et al.*, 2010b; *Kohli et al.*, 2008; *Yao and Fei-Fei*, 2010; *Yao et al.*, 2011) (Fig. 1.3(e)). A few models capture more sophisticated relationships by assuming the a 2D and half reconstruction of the layout of the scene is available (*Hoiem et al.*, 2008; *Gould et al.*, 2009b,a). This is typically obtained by estimating the vanishing lines of the scene. However, these models make assumptions on camera and scene layout (e.g., only one supporting plane and no camera in-plane rotation) (Fig. 1.3(f)).

1.3 Our Contributions

In this thesis we propose a number of novel 2D and 3D models for object and scene understanding that seek to meet the challenges discussed above and address some of the key limitations of previous work.

First, we propose object models that are capable of detecting generic rigid objects and simultaneously extracting their locations and viewpoints (models for 3D object recognition and view point estimation; Sec. 1.3.2 and Chapter III). Our proposed models capture the intrinsic 3D nature of the object category by linking object parts

across views using probabilistic relationships. We introduce a new learning procedure based on variational Expectation Maximization (EM) that learns part configurations within or across view in a semi-supervised fashion. Moreover, we propose a novel probabilistic formulation that allows learning the distribution of object features in 3D automatically and infer the 3D object shape from a single query image (Models for 3D object shape inference; Sec. 1.3.3 and Chapter IV) – an inherently ill posed problem. The ability of these models to jointly capture appearance and shape in a truly 3D sense makes them robust to intra-class variability, view point changes and occlusions.

Second, we have focused on designing models that can effectively capture the appearance and shape variability of articulated objects such as humans or animals. A premium has been put on models that can handle arbitrary object part configurations (poses) and simultaneously detect and estimate the object poses from a single image (Models for articulated objects; Sec. 1.3.4 and Chapter V). Moreover, a key contribution has been to propose algorithmic solutions that are capable of taming the intrinsic complexity of the pose estimation problem (i.e., explore an hypothesis space that is exponentially huge) while guaranteeing the optimality of the solution (efficient inference on loopy models for articulated objects; Sec. 1.3.5 and Chapter VI).

Finally, we have proposed models that can effectively capture the interplay among objects and scene layout (location and orientation of supporting surfaces in 3D) using less restrictive hypotheses than state-of-the-art approaches (Models for capturing interplay between objects and scene layout; Sec. 1.3.6 and Chapter VII). By jointly reasoning about objects, their geometrical properties (pose, shape) and scene elements (e.g., supporting surfaces and background regions) we have introduced probabilistic models that enable coherent scene understanding as well as more accurate object detection and segmentations results than existing methods (Models for coherent scene understanding; Sec. 1.3.7; chapter VIII).

	Goals	Inference	Supervision	Learning	Dataset	Chapter
Models for 3D Objects	Detect objects and estimate viewpoints.	Hough voting	Object bounding boxes	Variational EM and Discriminative learning	3D object (extended) and Pascal dataset	III
	Detect objects, estimate viewpoints, and recover 3D shapes.	Hough voting	Object bounding boxes	Maximum likelihood	Table-Top, ETHZ Shape, and Pascal dataset.	IV
Models for Humans	Detect humans and estimate poses.	Dynamic programming	Body part labels	Discriminative learning	Poselet, Stickmen, and Human Parsing dataset.	V
	Estimate poses on loopy models.	Branch and Bound	Body part labels	Discriminative learning	Stickmen, Buffy, and Video Pose dataset.	VI
Models for Scene Under-standing	Detect objects, segment supporting regions, estimate layout.	Iterative procedure	Segment labels	Independent learning	Table-top, Labelme, and Office dataset.	VII
	Detect objects, segment regions, estimate layout.	Graph cut	Segment labels	Discriminative learning	Stanford and Pascal dataset	VIII

Table 1.1: Summary of themes and our contributions.

1.3.1 Themes

In this thesis, the following themes appear throughout many chapters (see Table 1.1 for summary).

Level of supervision: we present methods requiring different level of supervision. For example, models for 3D object recognition require minimal supervision (no object pose or parts; only object location and category labels), whereas models for human pose require a larger degree of supervision (object location, body part, and category labels); Less supervision allows to easily collect more training data so that the learned models can generalize well on unseen testing images. However, without an appropriate level of supervision, it is harder to train models that are capable of capturing detailed properties such as object pose and parts. In Chapter III, we overcome this challenge and propose a semi-supervised model to automatically align object pose and find object parts that consistently appear across viewpoint for rigid objects. Notice that when detailed properties are difficult to infer automatically (e.g., body parts of articulated objects), larger degree of supervision becomes necessary (Chapter V and VI).

Discriminative vs generative models: The advantages of generative models are its flexibily and modeling power. Moreover, there exist standard mathematical tools to learn models with hierarchy of latent variables. However, generative models are typically less discriminative than discriminative models which can be used to learn the optimal conditional probability distribution such as the decision boundary between positive and negative samples for classification tasks. In Chapter III, we use a generative model to describe the generative process of parts location and appearance under different view points. Using variational inference, we automatically learn latent parts that consistently appeared across viewpoints. During recognition, a discriminative model (i.e., random forest) is used to localize the parts in order to achieve high object detection accuracy. In general, we observed that it is critical to use discriminative models (e.g., random forest (Chapter IV) and Conditional Random Fielf (CRF) (Chapter VIII)) to handle large appearance variation from background clutter. Therefore, in most of our work, discriminative models are used as the fundamental building blocks of our system.

Learning strategies. Mainly two learning strategies are used in this thesis. Firstly, maximum likelihood is used to learn the model parameters and latent variables in Chapter III and IV by maximizing the log probability with different forms. In Chapter III, Sec. 3.2.1.2, all model parameters are jointly learned by maximizing the log marginal probability. In Chapter IV, Sec. 4.2.1.1, the model parameters associated to discriminative tools such as random forest are separately estimated. Finally, max-margin is also used to learn models parameters in Chapter V, VI, and VIII by maximizing the normalized distance between training instances and the separation hyperplane. In general, we found models learned with max-margin objective function tend to achieve higher accuracy.

Model complexity. Many of our models seek to capture interactions between elements (Chapter V and VI). Among them, a tree model is proposed in Chapter V and

a loopy model is proposed in Chapter VI. On the one hand, tree models are more compact (fewer number of parameters) and there exist efficient inference algorithms to find Maximum A Posteriori (MAP) estimation of the tree models compared to loopy models. However, tree models are less descriptive compared to loopy models, and lead to lower accuracy. On the other hand, loopy models have larger number of parameters and they lead to inference problems that are known to be NP-hard. In Chapter VI, we demonstrate inference algorithm for loopy models that are both tractable and lead to more accurate estimation results.

We have also proposed models that captures high order interactions (relations among more than 2 elements) and proposed an efficient inference algorithm to estimate complex interactions such as interactions among an object and multiple regions in chapter VIII. Notice that the high order interactions cannot be trivially decomposed to pair-wise interactions. However, the key for the development of the efficient inference algorithm is to show that, by conditioning on a subspace of variables, the high order interactions can be approximately decomposed into pair-wise interactions.

Optimization strategies. We use different optimization strategies to solve our learning and inference problems. Variational inference is used to approximately solve the learning problem in Chapter III. Dynamic programming is used to solve the MAP inference problem on tree models (Chapter V). A novel branch-and-bound algorithm is proposed to solve MAP inference on loopy models efficiently. A novel iterative inference procedure is applied to approximately solve the scene understanding problem in Chapter VII. An efficient graph-cut algorithm is used to approximately solve the MAP inference problem for coherent understanding of objects and scene elements in Chapter VIII. A general strategy that is shared among most of the above methods is to decompose our learning and inference problems into smaller problems that can be solved by existing optimization strategies very efficiently (e.g., variational inference, dynamic programming, and graph-cut). In some cases, such decomposition is

not possible. For the problem in Chapter VI, we propose a novel branch-and-bound algorithm to solve the MAP inference problem on a loopy model which is known to be NP-hard.

Experimental evaluation. In order to evaluate our models and compare them with state-of-the-art baseline methods, we conduct experiments on public available datasets such as 3D object dataset (*Savarese and Fei-Fei*, 2007), PASCAL dataset (*Everingham et al.*, 2006, 2007), ETHZ Shape dataset (*Ferrari et al.*, 2008a), PASCAL Stickmen dataset (*Eichner and Ferrari*, 2009), Buffy dataset (*Ferrari et al.*, 2008b), Poselet dataset (*Bourdev and Malik*, 2009), Stanford dataset (*Gould et al.*, 2009a), Office scene dataset (*Sudderth et al.*, 2008), and Label-me dataset (*Hoiem et al.*, 2006). When necessary, we collected a number of in-house datasets to demonstrate specific properties or capabilities of our models. For example, in Chapter III, we extended the dataset in (*Savarese and Fei-Fei*, 2007) with several additional object categories to demonstrate that our method can generalize well on a variety of object categories. Moreover, in Chapter IV, we collect one of the first dataset for object categories where images are associated with depth maps.

In the following sections, we describe our contributions in details.

1.3.2 Models for 3D Object Recognition and View Point Estimation

Recognizing object categories and their 3D viewpoints is an important problem in computer vision. In Chapter III, we propose a new 3D object categorical model that is capable of recognizing unseen views, estimating poses, and synthesis new views (Fig. 1.4). We achieve this by using a dense, multi-view representation of the viewing sphere parameterized by a triangular mesh of viewpoints. Each triangle of viewpoints can be morphed to synthesize new viewpoints. By incorporating 3D geometrical constraints, our model establishes explicit correspondences among object parts across viewpoints. We propose an incremental learning algorithm to train the

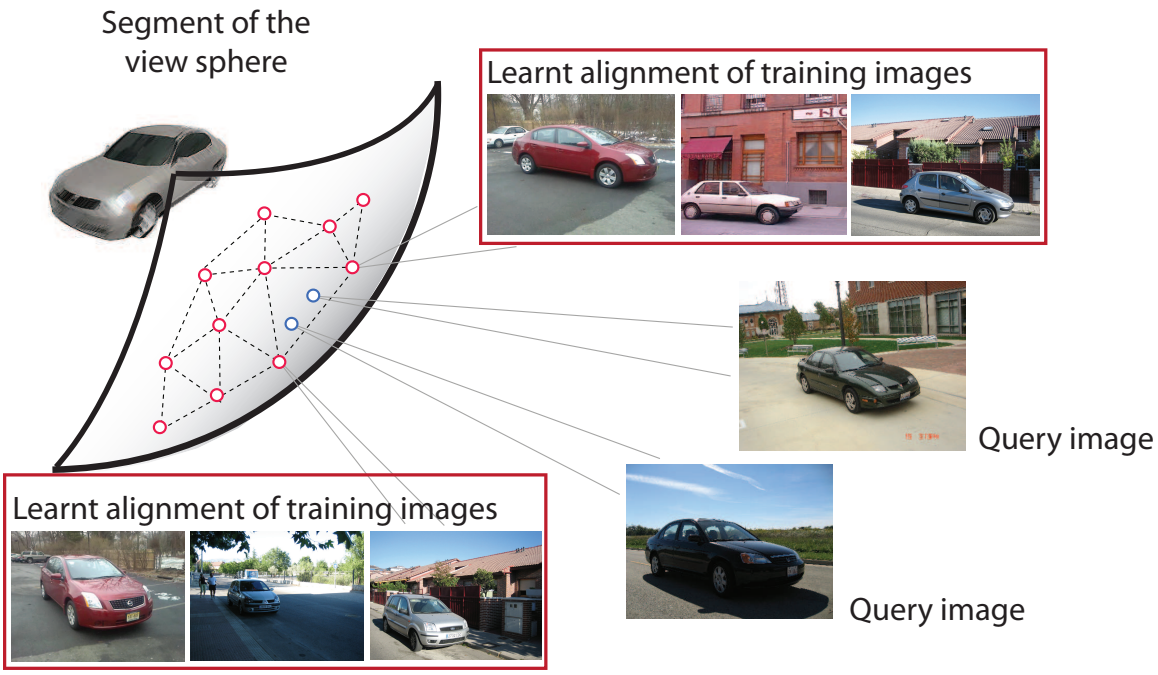


Figure 1.4: In Chapter III, we propose a dense multi-view representation of 3D object categories. Consider a car category as an example. The red circles on the viewsphere indicate the viewpoints of the object category learned by our model. Some sample training images are shown for two of the viewpoints, demonstrating our models ability to automatically align unlabelled poses at training time. Given a query image (dark blue circle) our model is capable of simultaneously categorize the object in the image and estimate its correct viewpoint by synthesizing a novel pose at recognition time.

generative model. After a suitable initialization step, the model is updated by a set of unsorted training images without viewpoint labels. We demonstrate the robustness of our model on object detection, viewpoint classification and synthesis tasks. Our model performs superiorly to and on par with state-of-the-art algorithms on the 3D object dataset (*Savarese and Fei-Fei*, 2007) and PASCAL'06 datasets (*Everingham et al.*, 2006) in object detection. It outperforms all previous work in viewpoint classification and offers promising results in viewpoint synthesis.

1.3.3 Models for 3D Object Shape Inference

Other than detecting objects and estimating their poses, recovering 3D shape information is also a critical problem in many vision and robotics applications. In Chap-

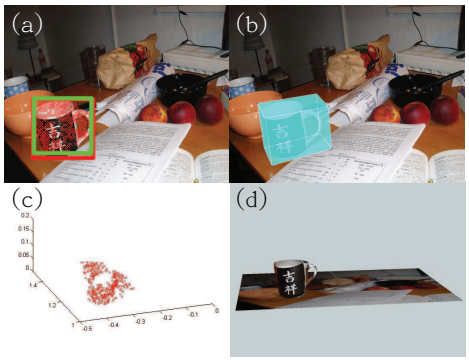


Figure 1.5: Key steps of our Depth-Encoded Hough Voting (DEHV) system (Chapter IV): (a) Detected object (red and green bounding boxes indicate the ground truth and estimated object location); (b) estimated object 3D pose (light blue cube); (c) reconstructed partial surface elements (3D points); (d) reconstructed 3D model.

ter IV, we address the above needs using a two stages approach. In the first stage, we propose a new method called DEHV - Depth-Encoded Hough Voting. DEHV jointly detects objects, infers their categories, estimates their poses, and infers/decodes objects depth maps from either a single image (when no depth maps are available in testing) or a single image augmented with depth map (when this is available in testing) (Fig. 1.5(a,b,c)). Inspired by the generalized Hough voting scheme introduced in *Leibe et al.* (2004), DEHV incorporates depth information into the process of learning distributions of image features (patches) representing an object category. DEHV takes advantage of the interplay between the scale of each object patch in the image and its distance (depth) from the corresponding physical patch attached to the 3D object. Once the depthmap is given, a full reconstruction is achieved in a second (3D modelling) stage, where modified or state-of-the-art 3D shape and texture completion techniques are used to recover the complete 3D model (Fig. 1.5(d)). Extensive quantitative and qualitative experimental analysis on existing datasets (*Everingham et al.*, 2007; *Ferrari et al.*, 2008a; *Savarese and Fei-Fei*, 2007) and a newly proposed 3D table-top object category dataset shows that our DEHV scheme obtains competitive detection and pose estimation results. Finally, the quality of 3D modelling in terms of both shape completion and texture completion is evaluated on a newly proposed 3D modelling dataset containing both in-door and out-door object categories. We demonstrate that our overall algorithm can obtain convincing 3D shape reconstruction from just one single uncalibrated image.

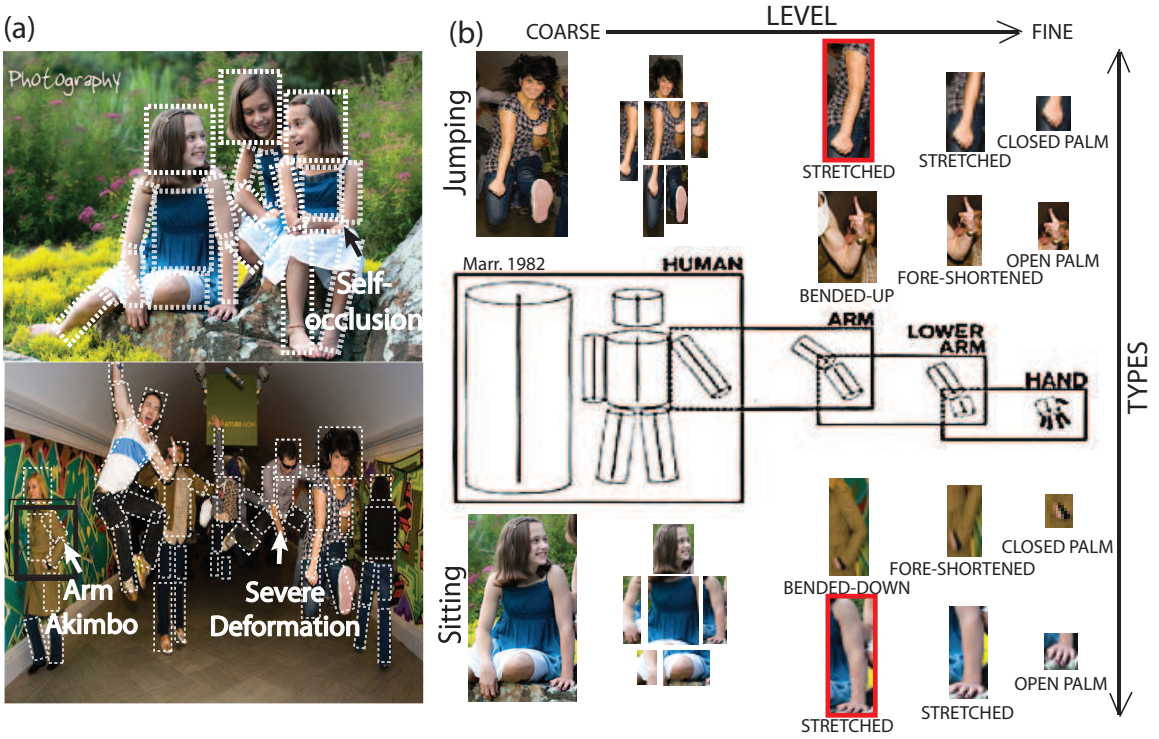


Figure 1.6: Panel (a) shows the ideal human pose estimation results indicated by the white bounding boxes and highlights the challenges including large appearance variation, part deformation, and self-occlusion. In Chapter V, we propose a new model for jointly detecting objects and estimating their pose (Panel (b)). Inspired by Marr (*Marr*, 1982), our model recursively represents the object as a collection of parts from a coarse-to-fine level (e.g., see horizontal dimension) using a parent-child relationship with multiple part-types (e.g., see vertical dimension). We argue that our representation is suitable for "*taming*" large pose and appearance variability.

1.3.4 Models for Articulated Objects

Despite recent successes in generic object detection, articulated objects are still very difficult to detect (Fig. 1.6(a)). Knowledge about the articulated nature of these objects, however, can substantially contribute to the task of finding them in an image. It is somewhat surprising, that these two tasks are usually treated entirely separately. In Chapter V, we propose an Articulated Part-based Model (APM) for jointly detecting objects and estimating their poses. APM recursively represents an object as a collection of parts at multiple levels of detail, from coarse-to-fine, where parts at every level are connected to a coarser level through a parent-child relationship

(a) Tree Model (b) Full Model

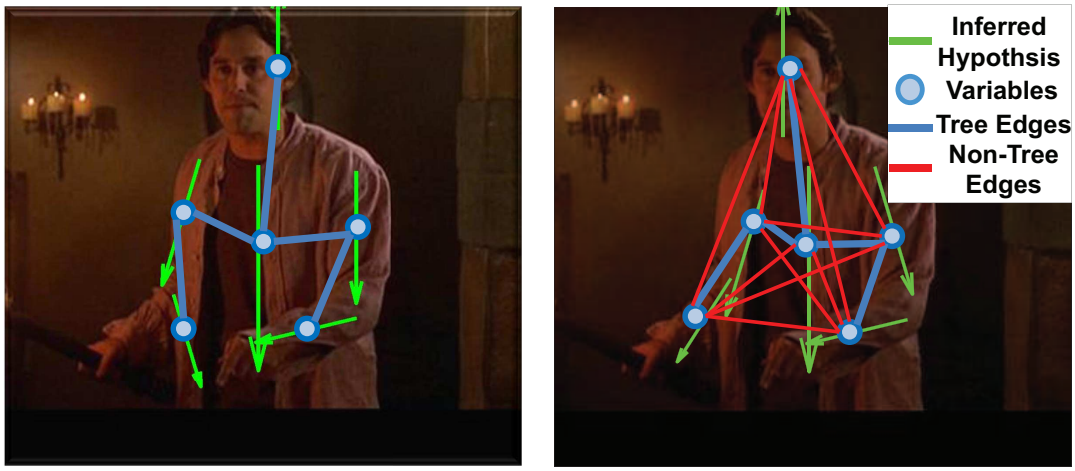


Figure 1.7: Panel (a,b) show the graphical representation of human model for six body parts. Here, circles denote parts, and blue and red edges denote the relationships between pairs of parts in the tree and full model respectively. The best configuration are shown in green arrows, where each arrow indicates the location and orientation of the body part. As we discussed in Chapter VI, the full model allows to detect the correct pose (b), whereas the tree model produces the wrong right-lower-arm (a).

(Fig. 1.6(b)-Horizontal). Parts are further grouped into part-types (e.g., left-facing head, long stretching arm, etc) so as to model appearance variations (Fig. 1.6(b)-Vertical). By having the ability to share appearance models of part types and by decomposing complex poses into parent-child pairwise relationships, APM strikes a good balance between model complexity and model richness by having the relationships between parts to form a tree-structure. Extensive quantitative and qualitative experiment results on public datasets (e.g., Poselet (*Bourdev and Malik*, 2009), Buffy (*Ferrari et al.*, 2008b) and Stickmen (*Eichner and Ferrari*, 2009) datasets) show that APM outperforms state-of-the-art methods. We also show results on PASCAL'07 (*Everingham et al.*, 2007) - cats and dogs - two highly challenging articulated object categories.

1.3.5 Efficient Inference on Loopy Models for Articulated Objects

The ability to capture the human pose with tree models is limited since many informative relationships between parts, such as symmetric relationships between left and right limbs, cannot be captured using such models (Fig. 1.7(a)). A natural extension are loopy models (parts are more densely connected and the model captures the dependency among most of the parts (Fig. 1.7(b))). However, since loopy models make inference intractable, approximated inference methods are usually used. In Chapter VI, we propose a novel Branch-and-Bound (BB) algorithm to solve human pose estimation problem on loopy models. Our analysis of the proposed algorithm on synthetic data shows that, given a limited time budget, our method solves problems that are characterized by a much *larger* number of hypotheses per part when compared to state-of-the-art exact inference algorithms (e.g., *Marinescu and Dechter* (2007); *Sontag et al.* (2008a)) and other baseline BB methods. We show that our method is theoretically and empirically much faster (about two orders of magnitude) than the state-of-the-art exact inference algorithm (*Sontag et al.*, 2008b). By extending a state-of-the-art tree model (*Sapp et al.*, 2010b) to a loopy model, the estimation accuracy can be consistently improved across all parts, especially for lower arms (up to $\sim 5\%$ improvement) on Buffy (*Ferrari et al.*, 2008b) and Stickmen (*Eichner and Ferrari*, 2009) datasets. We further demonstrate that our method is well suited for other problems, such as protein design in *Yanover et al.* (2006), that incorporate pair-wise relationships between elements where the hypothesis space per element is large. In particular, given a time budget of up to 20 minutes, our method consistently solves more protein design problems than *Sontag et al.* (2008b) does.

1.3.6 Models for Capturing Interplay between Objects and Scene Layout

By utilizing the geometrical properties extracted from objects as discussed in Chapter III and IV, we propose a new model, called "Context Feedback Loop Model",

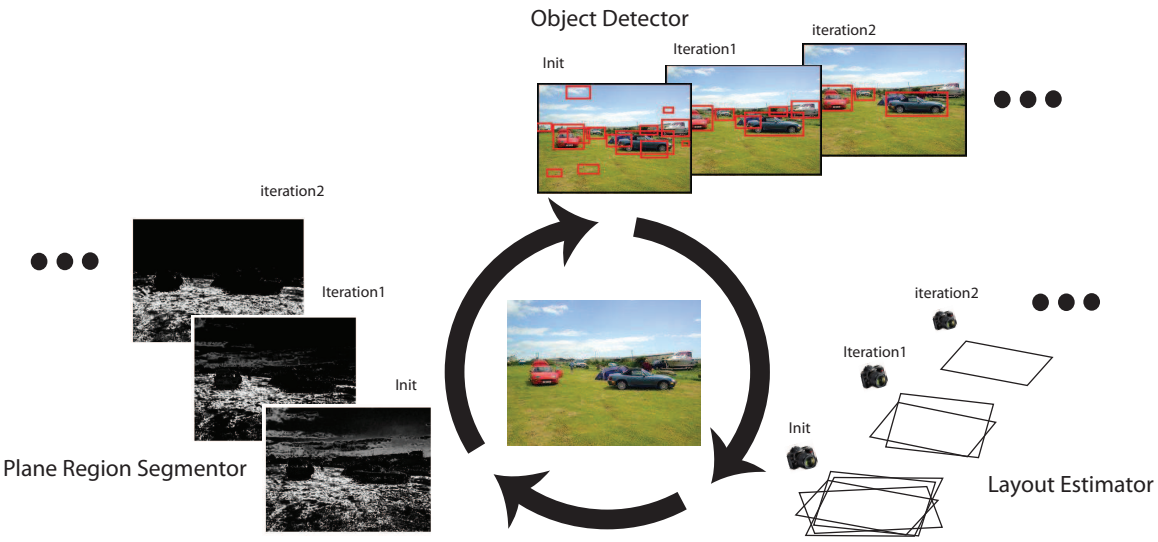


Figure 1.8: The Context Feedback Loop Model. In Chapter VII, we demonstrate that scene layout estimation and object detection can be part of a joint inference process. In this process a supporting region segmentation module (RS) and a scene layout estimation module (LE) provides evidence so as to improve the accuracy of an object detector module (OD). In turn, the object detector module enables a more robust estimation of the scene layout (supporting planes orientation, camera viewing angle) and improves the localization of the supporting regions.

for capturing the interplay between objects, 3D layout, and object supporting regions from a single image (Fig. 1.8) in Chapter IV. Specifically, we study the interaction between three modules: i) object detector; ii) scene 3D layout estimator; iii) object supporting region segmenter. The interactions between such modules capture the contextual geometrical relationship between objects, the physical space including these objects, and the observer. An important property of our algorithm is that the object detector module is capable of adaptively changing its confidence in establishing whether a certain region of interest contains an object (or not) as new evidence is gathered about the scene layout. This enables an iterative estimation procedure where the detector becomes more and more accurate as additional evidence about a specific scene becomes available. Extensive quantitative and qualitative experiments are conducted on the newly proposed table-top dataset (*Sun et al.*, 2010b) and two publicly available datasets (*Sudderth et al.*, 2008; *Hoiem et al.*, 2006), and demon-

strate competitive object detection, 3D layout estimation, and segmentation results.

1.3.7 Models for Coherent Scene Understanding

As opposed to Chapter VII, where only a special class of scene elements is considered (i.e., supporting regions), in Chapter VIII, we propose a framework for scene understanding that models both objects and scene elements (e.g., glass, buildings, etc.) using a common representation while preserving their distinctive nature. This representation allows us to enforce sophisticated geometric and semantic relationships between objects and scene elements in a single graphical model using a list of shared properties (e.g., category, depth, 2D location, etc.). We use the latest advances in the field of discrete optimization to efficiently perform maximum a posteriori (MAP) inference using this model. We evaluate our method on the Stanford dataset (*Gould et al.*, 2009a) by comparing it against state-of-the-art methods for object segmentation and detection. We also show that our method achieves competitive performances on the challenging PASCAL'09 (*Everingham et al.*, 2009) segmentation dataset.

Finally, we conclude in Chapter IX with proposals for future directions in object and scene understanding.

CHAPTER II

Background

In this chapter, we give an overview of the main literature on object recognition (Sec. 2.1), articulated object recognition (Sec. 2.2), and scene understanding (Sec. 2.3).

2.1 Object Recognition

In object recognition, an object model is typically defined so as to capture objects geometrical and appearance properties at the appropriate level of specificity. For instance, an object model can be designed to recognize a generic "car" as opposed to "a specific car model", or vice versa. In the former case, which is often referred as the *object categorization* problem, the main challenge is to design models that are capable of retaining key visual properties for representing an object category, such as a "car", at the appropriate level of abstraction. Such models can be then used to recognize novel object instances from a query image. Moreover, a model must be able to generalize across variations in the objects visual characteristics due to view point and illuminations changes as well as due to occlusions or deformations. Meeting all of these desiderata can be extremely challenging. Researchers have proposed to reduce the complexity of the representation by making assumptions on the type of object specificity or the degree of view point, occlusions and deformation variability. Ulti-

mately, the strategy in designing an object model depends on the relevant application scenario.

2.1.1 Object Instance Recognition

Object models that are designed to recognize an object instance – e.g., "a specific car model" as opposed to "all cars" – are often referred to as *single instance object* models. These models are capable of recognizing a specific object instance while guaranteeing the ability to handle occlusions and a large degree of view point variability (Fig. 1.2(b)). Research on object instance recognition, from early contributions (*Biederman*, 1985; *Binford*, 1971; *Marr*, 1978; *Palmer*, 1975; *P. Winston*, 1975; *Palmer et al.*, 1981; *Tarr and Pinker*, 1989; *Poggio and Edelman*, 1990; *Ullman and Basri*, 1991; *Koenderink and Doorn*, 1979; *Huttenlocher and Ullman*, 1987; *Lowe and Binford*, 1985) to the most recent ones (*Jacobs and Basri*, 1999; *Lowe*, 1999; *Obdrzalek and Matas*, 2002; *Rothganger et al.*, 2003; *Brown and Lowe*, 2005; *Gordon and Lowe*, 2006; *Romea et al.*, 2009; *Ferrari et al.*, 2006; *Hsiao et al.*, 2010; *Xu et al.*, 2009) follows these assumptions. Since single instance object models do not need to accommodate any intra-class variations, they often consist of a rigid collection of visual features associated to a number of 2D or 3D templates. In recognition, by matching features of the query image with those associated to the models, it is possible to identify the object of interest and determine its 3D pose with respect to a common reference system. This matching process is usually subject to a geometrical validation phase that helps verify that the appearance and geometric properties of the query object are consistent with the estimated pose transformation between observation and object model. While critical for ensuring sufficient discrimination power for recognizing single instance objects as well as for enabling large view point variability, tight geometrical constraints become inadequate when shape and appearance intra-class variability must be accounted for. Therefore, these methods are best

suitable for applications where the object instances are known and an accurate 3D pose of the object is required.

2.1.2 Single View Object Category Recognition

Object models that are designed to recognize objects category - e.g., "all cars" as opposed to "a specific car model" are often referred to as *categorical* object models. In early works on object categorization, researchers assume that objects are viewed from a limited number of poses and learn object models that are specialized to identify the object from a specific view point. The ability to generalize across instances in the same category (Fig. 1.2(a)) is critical and is typically achieved by characterizing the object as a collection of features whose appearance and geometrical properties tend to systematically occur in the category of interest. For instance, if the goal is to recognize a car, appearance properties such as the "color of the body" is not adequate to help obtain the right level of generalization (abstraction), whereas the orientation of edges associated to a wheel can capture more general appearance cues across instances. Appearance properties are typically captured by image descriptors such as SIFT (*Lowe*, 1999) or SURF (*Bay et al.*, 2008) associated to interest points that are detected at different locations and scales of the image (*Lowe*, 1999; *Mikolajczyk and Schmid*, 2002). A popular design choice is to describe the object appearance by histograms of vector quantized descriptors (*Csurka et al.*, 2004; *Grauman and Darrell*, 2005; *Fei-Fei et al.*, 2004). The ability of image descriptors such as SIFT (*Lowe*, 1999) to be invariant to affine illumination transformations makes the appearance models robust to variability in illumination conditions. Geometrical properties are captured by retaining the spatial organization of features in the image and including simple characterizations based on the 2D location of either feature points or aggregation of features (e.g. edges, parts, fragments) with respect to a given object reference point (*Fergus et al.*, 2003; *Felzenszwalb and Huttenlocher*, 2005; *Leibe et al.*, 2004; *Lazebnik*

et al., 2006; *Savarese et al.*, 2006b). Object models constructed upon constellation of parts (*Fergus et al.*, 2003; *Felzenszwalb and Huttenlocher*, 2005; *Leibe et al.*, 2004) are suitable to accommodate object variations due to occlusions and simple 2D planar geometrical deformations (isometries or affinities). Suitable machine learning and probabilistic inference techniques such as Expectation Maximization (EM) (*Dempster et al.*, 1977), Structured Support Vector Machine (SSVM) (*Tsochantaridis et al.*, 2004), Markov Random Field (MRF) (*Koller and Friedman*, 2009; *Wainwright and Jordan*, 2008), Conditional Random Field (CRF) (*Lafferty et al.*, 2001), generalized Hough voting (*Ballard*, 1981), and RANdom SAmple Consensus (RANSAC) (*Fischler and Bolles*, 1981) are used to automatically select appearance and geometrical properties so as to reach the appropriate level of generalization and discrimination power.

2.1.3 3D Object Category Recognition

Most of the object models for object categorization (Sec. 2.1.2) mitigate the complexity of the representation by assuming that objects are viewed from a limited number of poses and learn an object model that is specialized to identify the object from a specific view point. These are often referred as view-dependent object models. If similar views in the training set are available, the recognition problem is reduced to match the new query object to one, or a mixture, of the learnt view-dependent object models. This is the approach taken by most of the existing literature (*Li et al.*, 2009b; *Ng and Gong*, 1999; *Schneiderman and Kanade*, 2000; *Weber et al.*, 2000a; *Zhang*, 2004). The drawback of view-dependent object models is that: i) they can accommodate very limited view point variability, mostly changes in scale or 2D rotation transformations; ii) different poses of the same object category results in completely independent models, where neither features or parts are shared across views. Because each single-view models are independent, these methods are often costly to train and

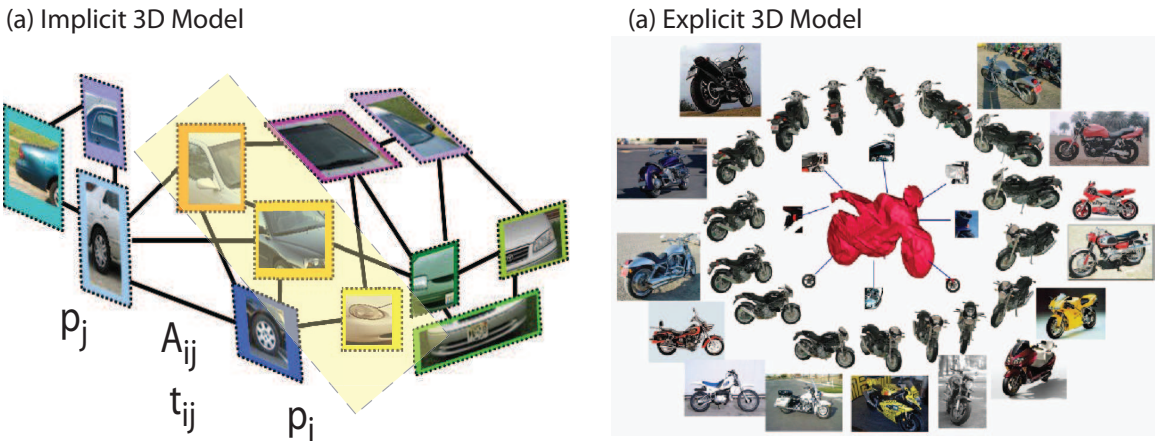


Figure 2.1: Panel (a) shows an example of implicit 3D models as introduced in *Savarese and Fei-Fei* (2007). In the model, object parts are connected to form a graph structure. Each node P_i captures diagnostic appearance of the object part which is assumed to be locally planar. Each edge describes an homographic transformation that captures the view point transformation between parts. The homographic transformation is illustrated by showing that some parts are slanted with respected to others. Panel (b) shows an example of explicit 3D models as introduced in *Pingkun Yan and Shah* (2007). In the model, object elements (interest points) are organized in a common 3D reference frame and form a compact 3D representation of the object category.

prone to false alarms, if several views need to be encoded.

Object models that can accommodate both large view point changes and large intra-class variability (low degree of specificity) overcome the above limitations by introducing a representation that effectively captures the intrinsic three-dimensional nature of the object category. These models are typically divided into two types: implicit 3D models and explicit 3D models. In the implicit 3D models (*Thomas et al.*, 2006; *Kushal et al.*, 2007; *Savarese and Fei-Fei*, 2007; *Farhadi et al.*, 2009b), object diagnostic elements (features, parts, contours) are connected across views to form an unique and coherent implicit 3D model for the object category (Fig. 2.1(a)). Relationships between features or parts capture the way that such elements are transformed as the viewpoint changes. Notice that our own models described in Chapter III are the first that describe such relationships using probabilistic functions and learn part configurations in a semi-supervised fashion. These methods share some key ideas with

pioneering works in 3D object recognition (*Biederman*, 1985; *Binford*, 1971; *Marr*, 1978; *Palmer*, 1975; *P.Winston*, 1975; *Palmer et al.*, 1981; *Tarr and Pinker*, 1989; *Poggio and Edelman*, 1990; *Ullman and Basri*, 1991; *Huttenlocher and Ullman*, 1987; *Lowe and Binford*, 1985) as well as with the theory of aspect graphs (*Koenderink and Doorn*, 1979; *Bowyer and Dyer*, 1990). In the explicit 3D models (*Sun et al.*, 2010b; *Hoeim et al.*, 2007; *Chiu et al.*, 2007; *Pingkun Yan and Shah*, 2007; *Liebelt and Schmid*, 2010; *Stark et al.*, 2010; *Arie-Nachimson and Basri*, 2009; *Xiang and Savarese*, 2012; *Payet and Todorovic*, 2011a) object elements are organized in a common 3D reference frame and form a compact 3D representation of the object category (Fig. 2.1(b)). Such 3D structures of features (parts, edges) can give rise, for instance, to either a 3D generalization of 2D pictorial structures or constellation models or to hybrid models where features (parts or edges) lie on top of 3D object reconstructions or CAD volumes. A comprehensive survey of 3D object detection methods is presented in *Hoeim and Savarese* (2011).

In Chap. III and IV, we introduce a novel dense multi-view representation of 3D object categories and the Depth-Encoded Hough Voting (DEHV) detector to extract 3D object geometrical attributes (e.g., location, pose, and shape) from a single 2D image, respectively.

2.2 Articulated Object Recognition

Recognizing articulated objects (such as humans or animals) is more challenging than recognizing generic rigid objects (such as cars or cups). Not only a model for articulated objects should accommodate appearance variability (people wear different cloths), shape variability (some people are short, other are slim), viewpoint variability (people can be observed from any arbitrary view point), but also pose variability: the configuration of body parts change as the person performs different activities (walk, sit, jump) (Fig. 1.2(c)). This makes existing techniques for rigid object recognition

inadequate. It is important to notice that the problem of estimating the pose of articulated objects – in particular humans – it is tightly related to the one of activity recognition and plays a critical role in modelling the interplay between humans and objects (human-object interaction recognition) (*Yao and Fei-Fei*, 2010; *Yang et al.*, 2010) or among humans (collective activity recognition (*Choi et al.*, 2011; *Lan et al.*, 2010)). In the remainder of the this thesis we will mostly refer to the literature on human detection and body parts estimation since humans are, by far, the most interesting articulated "object" in real world applications.

2.2.1 Human Detection

Most of the early works on human detection focused on recognizing humans as a whole, without attempting to extracting the body part configuration. *Dalal and Triggs* (2005) propose a novel feature called "Histogram of Oriented Gradients" for detecting pedestrians. When the feature is combined with a linear SVM classifier, efficient and accurate detection performance is achieved on pedestrians where the pose variability is very limited (e.g., standing and walking). *Felzenszwalb et al.* (2008) propose a Deformable Part-based Model (DPM) to detect objects and it achieves state-of-the-art performances on the challenging PASCAL dataset. In order to handle deformations, the model consists of a mixture of templates, where each template corresponds to a specific human pose such as standing, sitting, etc. *Bourdev and Malik* (2009) further propose a "Poselet" representation for capturing local human body parts configurations. It achieves better performance than DPM by incorporating a larger number of unique poses compared to the DPM.

2.2.2 Human Pose Estimation

Unlike human detection, the focus of human pose estimation is to correctly estimate the body parts configurations. Pictorial Structure (PS) (*Felzenszwalb and*

Huttenlocher, 2005) is the most common approach for human pose estimation. The PS method essentially represents each part as a variable with a large number of possible locations and each part is related to its connected parts following the kinematic constraints of the human body to form a tree structure. This simple tree structure model can be efficiently learned and have been successfully applied for human pose estimation. Researchers have proposed extensions and generalized the above model following two main directions: i) Improving the robustness of part detectors (*Andriluka et al.*, 2009; *Ramanan*, 2006; *Eichner and Ferrari*, 2009). In particular, *Andriluka et al.* (2009) show that classifiers based on boosting learning strategies can detect parts very robustly, and the detections can be used by the tree models to improve the overall pose estimation accuracy. ii) Improving the discriminative power of the pair-wise relations (*Sapp et al.*, 2010b; *Ramanan*, 2006; *Yang and Ramanan*, 2011; *Sun and Savarese*, 2011). *Sapp et al.* (2010b) propose to use a pair-wise feature that depends on the image appearance (e.g., color, contour, segmentation, etc) to enhance the model discriminative power. *Yang and Ramanan* (2011) and our own method described in Chapter V use the concept of part-type (i.e., parts with specific orientation or foreshortening) to model pair-wise relations of among parts. This way, the pair-wise relations can capture co-occurrence of parts with specific orientation or foreshortening, and can increase the expressiveness and flexibility of the model.

Without making the assumption of restricting the model structure to a tree, many models based on loopy structures have been successfully employed to solve human pose estimation problem. Interactions between pairs of parts have been incorporated by *Jiang and Martin* (2008); *Tran and Forsyth* (2010); *Ren et al.* (2005) in order to encode longer range kinematic constraints as well as encode information such as self-occlusion and color similarity of symmetric parts. *Wang et al.* (2011) propose hierarchical models of parts across multiple scales. In these models, parts at a lower level of the hierarchy are grouped into parts at a higher level of the hierarchy. In

particular, *Wang et al.* (2011) show that parts at a higher level of hierarchy are easier to detect in isolation since they possess very distinctive appearance features (e.g., the whole human body is easier to detect than the hands).

A few works have been proposed to solve the pose estimation problem on loopy models using inference methods based on maximium a posteriory (MAP) maximization. *Tian and Sclaroff* (2010) propose an efficient branch-and-bound algorithm for a tree model augmented with two additional pair-wise relations between left-right legs. The BB search is efficient since it only takes constant time to evaluate the bounds, thus enabling the solution of problems with a large number of states. Notice, however, that the tightness of such bound guarantees efficient search only when the energy originated from the additional pair-wise relations is small (Fig.7 in *Tian and Sclaroff* (2010)). This makes it hard for *Tian and Sclaroff* (2010) to solve a model with a large number of pair-wise interactions. *Bergtholdt et al.* (2010) convert the inference problem over a fully connected model into a shortest path problem and propose an efficient A^* search method for solving it. The main drawback of the A^* search is that the branching factor of the search tree equals the number of states per variable (i.e., number of part location hypothesis). As a result, the method relies on a greedy procedure for pruning part hypotheses to ensure that the search problem is tractable. Cluster pursuit (*Sontag et al.*, 2008b) is an alternative exact inference algorithm which searches for higher-order constraints to tighten the gap between approximated solution and optimal solution. However, since the time complexity of the algorithm is proportional to the number of part hypotheses to the power of the order of the constraints (i.e., number of variables involved in the constraints), the algorithm becomes prohibitively slow for problems with a large number of part hypotheses. In Chapter VI, we introduce a novel and efficient branch-and-bound inference algorithm (*Sun et al.*, 2012c,b) to estimate the optimal configuration of body parts.

2.2.3 Joint Human Detection and Pose Estimation

Recall that most of the existing literature treats object detection and pose estimation as two separate problems. We argue that these two problems are two faces of the same coin and must be solved jointly. The ability to model parts and their relationship allows to identify objects in arbitrary configurations (e.g., jumping and sitting, see Fig. 1.6) as opposed to canonical ones (e.g., walking and standing). In turn, the ability to identify the object in the scene provide strong contextual cues for localizing object parts.

Andriluka et al. (2009) are the first to emphasize the importance of joint detection and pose estimation by demonstrating superior accuracy on both tasks. *Yang and Ramanan* (2011) propose a Flexible Mixtures of Parts (FMP) model for joint detection and pose estimation which achieves improved performance on both tasks. The FMP model demonstrates that a large number (e.g., ~ 26) of small body parts corresponding to human body joints (e.g., elbows, shoulders, etc.) with typical orientation (e.g., 5 or 6 orientations) can be combined to build more discriminative models. Similarly, in Chapter V, we introduce a novel model for joint detection and pose estimation called the Articulated Part-based Model (*Sun and Savarese*, 2011). The model achieves superior performance on both tasks by exploring a coarse-to-fine and multiple part-types representation to handle the pose variability.

2.3 Scene Understanding

As discussed in Chapter I, given a 2D image, we, humans, can easily interpret the underlying 3D scene that generates the image. For instance, we can describe the geometric and semantic properties of the objects within the scene. We can also explain the properties of the scene elements generating the specific 2D patterns in the image. Many theories have been proposed to explain how we do it. These in-

clude Gestalt emergence, Helmholtzian data-driven unconscious inference, etc (*Flock*, 1964). Similarly, a fundamental problem in computer vision is to achieve human-level interpretation of the 3D scene from a 2D image and being able to interpret the 2D images in terms of the various objects and scene elements, and estimating their semantic and geometrical properties.

Early works on scene understanding propose to infer the 3D scene of simple objects or scenes from line drawings. For example, *Kanade* (1981); *Barrow and Tenenbaum* (1981) demonstrate that strong constraints between objects and scene elements designed specifically for the particular setting can be used to achieve this goal. However, generalizing these methods to real world environments has been proven to be problematic. A comprehensive survey of early scene understanding models is presented in *Hoeim and Savarese* (2011).

2.3.1 2D Scene Elements

As a way to simplify the problem, researchers have proposed to model a scene as a collections of 2D image patterns which are used to characterize scene elements. Most methods (*Torralba et al.*, 2003; *Li and Fei-Fei*, 2007; *Li et al.*, 2009a; *Ladicky et al.*, 2010a; *Gonfaus et al.*, 2010; *Rabinovich et al.*, 2007) aim at parsing or segmenting the image into semantically consistent regions and assigning them to categorical labels (e.g., road, building, foreground objects, etc.). Thanks to the recent advance in statistical modelling, pattern recognition, and machine learning, breakthrough in scene understanding has been achieved. In particular, *Ladicky et al.* (2010a); *Gonfaus et al.* (2010); *Rabinovich et al.* (2007) leverage semantic context to capture the typical relationship among object categories co-occurring within each image (e.g., cars and roads are likely to co-occur). *Torralba et al.* (2003); *Li and Fei-Fei* (2007); *Li et al.* (2009a) leverage semantic context between object and scene categories (e.g., cars are likely to occur within an urban scene).

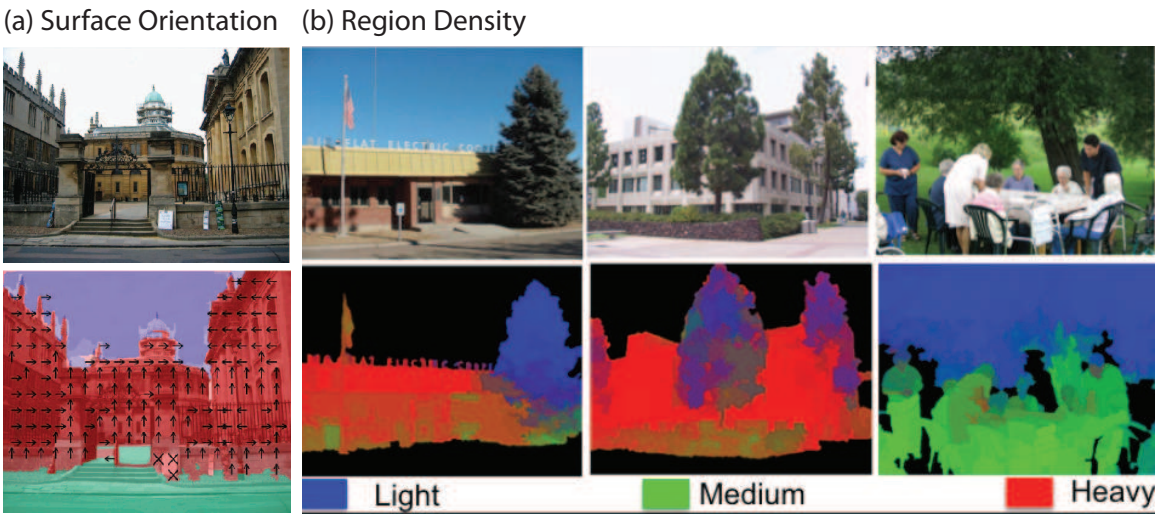


Figure 2.2: Panel (a) shows the surface orientation extracted by *Hoiem et al.* (2005a), where horizontal surface (green), sky (blue), vertical surfaces (red) subdivided into planar orientations (arrows) and non-planar solid (x) and porous (o). Panel (b) shows the region density (e.g., light, medium, and heavy) extracted by *Gupta et al.* (2010).

Other than exploring semantic contexts, methods have leveraged the typical 2D geometrical relationships between objects and scene elements to describe complex scene. These relationships are often referred to as "geometrical context". In particular, *Gupta and Davis* (2008b); *Sudderth et al.* (2008) propose to model 2D spatial relationships among scene elements in the 2D image plane. For example, a car is not only likely to co-occur with a street, but also likely to be "on-top" of the street (Fig. 1.3(e)).

2.3.2 Beyond 2D Scene Elements

While models based on 2D spatial relationships are suitable for scene categorization and for parsing the 2D elements, they cannot account for interplay of objects in the 3D space. *Hoiem et al.* (2006) investigate the possibility of integrating cues from the 3D scene such as vertical and horizontal surfaces (*Hoiem et al.*, 2005a) into the process of jointly detecting objects and estimating the scene layout. Moreover, *Hoiem et al.* (2006) model the interactions between objects and the scene elements

by assuming that objects are supported by ground planes. By further assuming that there is a single ground plane, objects' 2D scales can be used as a critical cue for modelling the interaction between objects as well as determining the distance (depth) of the objects from the camera. *Hoiem et al.* (2008); *Gould et al.* (2009a) further propose an iterative approach wherein additional cues such as the occluding boundaries between objects and the background scene elements in the image are injected into the inference procedure.

Hedau et al. (2009) model the explicit relationships between the scene layout and objects in 3D using a box representation (i.e. a representation that approximates the scene physical space as a 3D box). The 3D box representation provides a good approximation for 3D structures of many indoor scenes, with the floor, walls, and ceiling forming the sides of the box. Most importantly, it provides a more powerful reference frame than the ground plane, since orientation and position of objects can be defined with respect to the walls as well.

Models with grammar-based structures have also been proposed to capture more complex 3D structure of the scene. In *Gupta et al.* (2010), the scene is represented with a series of 3D blocks and planes. In order to establish the relationships between blocks, different region densities such as light density (e.g. trees and bushes), medium density (e.g. humans), and high density (e.g. buildings) (Fig. 2.2(a)) are considered. *Han and Zhu* (2005) parse a scene into rectangles, cubes, and other line-based structures that reflect the underlying 3D structure. Generalization to indoor scenes observed from video sequence have been proposed (*Tsai et al.*, 2011).

Other methods leverage the ability to infer the scene depth maps via probabilistic inference to model spatial relationships among scene elements in 2.5D (*Heitz et al.*, 2008; *Li et al.*, 2010; *Saxena et al.*, 2009; *Payet and Todorovic*, 2011b). On the one hand, depth map is a flexible representation for the 3D structure of the scene so that it is also suitable for outdoor rural scenes like mountains. On the other hand,

it provides less constraints on regularizing the possible 3D structure for indoor and outdoor urban scenes.

Finally, *Bao et al.* (2010b) capture the interaction between the object poses and the 3D supporting surfaces to estimate the 3D layout even when cues from the underlying scene (e.g., vanishing lines or scene surface orientations) are not available. Most importantly, the model is able to handle scene with multiple ground planes. In Chapter VII, we generalize this concept by introducing the ability to identify and segment supporting planes.

2.3.3 Object and Scene Elements

The ability to parse scene elements (Sec. 2.3.1) and the one to reason in 3D (Sec. 2.3.2) are very different to combined in an unique framework. Recently, researchers have proposed methods to jointly detect objects and segment out the scene elements as semantically coherent regions. Many of such methods are very complex and leverage iterative and approximated inference procedures for solving specific tasks such as detection, segmentation, and occlusion reasoning (*Heitz et al.*, 2008; *Hoiem et al.*, 2008; *Sun et al.*, 2010a) (Fig. 1.3(f)). The drawback of these inference procedures is that different objective functions are optimized independently without guaranteeing that a joint solution is reached.

Yao et al. (2012b); *Ladicky et al.* (2010c) and the method described in Chapter VIII utilize Random Field (RF) models to jointly characterize the scene elements, objects, and their 2D relationships. Principled inference algorithms such as message passing and graph-cut algorithms with guaranteed convergence properties are used to solve such complex recognition problems. In Chapter VIII, we present a generalization of such models called Augmented Conditional Random Field (ACRF) model (*Sun et al.*, 2012a), which incorporates scene element-object, object-object, and object-layout 3D relationships within a coherent formulation. Similar to our pro-

posed method, *Ladicky et al.* (2010c) incorporate object-scene element relationships and demonstrate that the information from object detection can be used to improve the segmentation performance consistently across all object categories. However, their model can be considered as a special case of our model when no object-object relationship is incorporated. It is also worthwhile to mention that *Desai et al.* (2009) propose a CRF model capturing object-object relationships and show that object detection performance can be consistently improved for multiple object categories. Their model, however, can also be considered as a special case of our model when no object-scene element and scene element-scene element relationships are incorporated.

CHAPTER III

Models for 3D Object Recognition and View Point Estimation

Visual recognition is a cornerstone task for an artificial intelligence system. In computer vision, object recognition, particularly object categorization, has been one of the most widely researched areas in recent years. Tremendous progress has been made especially in image-level object classification under limited geometric transformations, such as classification of side-view cars, or frontal view faces (e.g., *Ullman and Basri* (1991); *Fergus et al.* (2003). Also relevant is the line of work in object detection in cluttered real-world scenes, such as pedestrian detection, or car detection (*Felzenszwalb et al.*, 2008; *Dalal and Triggs*, 2005). But most of the previous approaches can only handle up to a small degree of viewpoint variations of the 3D objects. As a result, they can hardly be used for robust pose understanding, a crucial functionality for real-world applications where accurate recognition of objects under arbitrary view points and 3D poses are needed. A small but growing number of recent studies have begun to address the problem of object classification in a true multi-view setting (*Thomas et al.*, 2006; *Kushal and Ponce*, 2006; *Hoeim et al.*, 2007; *Pingkun Yan and Shah*, 2007; *Chiu et al.*, 2007; *Savarese and Fei-Fei*, 2008, 2007; *Liebelt et al.*, 2008). While this is an important step forward, the focus is still on object detection without extensive quantitative analysis of 3D viewpoint estimation (the exceptions being

Savarese and Fei-Fei (2008, 2007); *Liebelt et al.* (2008)). In this chapter, we propose a new framework for learning a probabilistic 3D object model that can be used to categorize and detect an object in a cluttered scene, estimate its viewpoints accurately, or synthesize a new viewpoint given a single test image. We focus on overcoming two major challenges in representing and modelling 3D object classes. Firstly, we develop a dense multi-view representation of object classes through an incremental learning algorithm (Sec. 3.1). The probabilistic model construction process is initialized from a video sequence of a single object instance. Our algorithm then builds the object class model from unsorted images without any viewpoint supervision by automatically aligning arbitrary poses at training time (Fig. 1.4). Secondly, our 3D object recognition algorithm is able to recognize objects under arbitrary viewpoints, even if the object instances were not observed during training. If we define the viewing sphere as a collection of viewpoints from which an object can be observed, our algorithm accurately estimates the pose of the object on it (Fig. 1.4). To our knowledge, this is the first probabilistic model that is capable of representing and recognizing unseen object views.

The rest of the chapter is organized as follows. We introduce the model in Sec. 3.1 and describe how it is learned in Sec. 3.2. Finally, we show application of our model in Sec. 3.3 and conclude in Sec. 7.3.

3.1 A Dense Multi-view Representation of 3D object Categories

Given a number of training images of an object category, our goal is to learn a dense, multi-view generative part-based model with minimal supervision. Then, we utilize the learned model to detect objects, estimate their poses, and synthesize new views from a test image. In the training stage, we assume that object bounding boxes

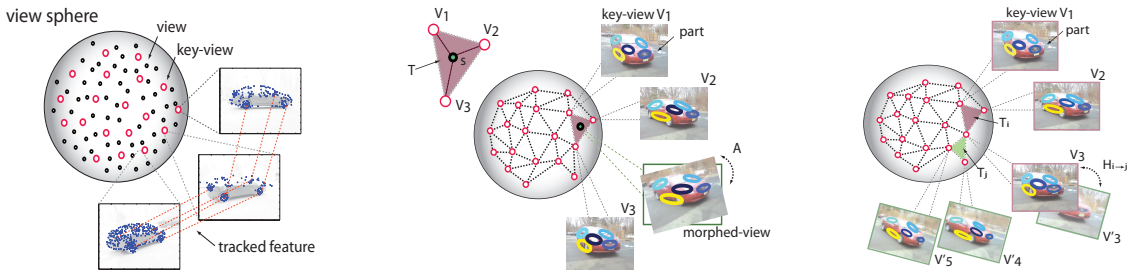


Figure 3.1: Schematic illustration of key concepts of our model. **(a)** View sphere, key views and tracked video frames. Using a cellphone video clip of a single object, we obtain a dense sample of viewpoints (black and red circles) on the view sphere. Features (dots on the car images) are tracked between consecutive video frames using the Lucas-Kanade algorithm (*Lucas and Kanade*, 1981). Some of the tracks are shown in red dotted lines between two pictures. A subset of the viewpoints are selected as key views (red circles) (Sec. 3.1.3. **(b)** The view sphere and all the key views are parameterized as a triangle mesh (Sec. 3.1.3). We formulate the view synthesis problem based on the view morphing technique. A morphing parameter S interpolates and extrapolates the triplet of viewpoints V_i in a given viewpoint triangle T. The post-warping transformation of viewpoint alignment is denoted by A. **(c)** Illustration of a 3D geometrical constraint across triangles on the view sphere. Given any key view on the view sphere, it is shared by two connected triangles on the view sphere. An affine transformation constraint $H_{i\to j}$ is then enforced to ensure consistent part estimate on viewpoints across different triangles on the view sphere.

of training images are given. This assumption is removed during the testing stage, as we would like the recognition scheme to handle not only intra-class and viewpoint variations, but also severe background clutter and occlusion.

In the following sections, we first describe our multi-view generative part-based model for object categories along with our proposed parameterization of the view sphere. Then, we describe our method for incrementally learning such models with weak supervision.

3.1.1 Model

Given a viewpoint, we propose to model an object category as a mixture of parts (Sec 3.1.2). Viewpoints are selected from the view sphere. The view sphere is parameterized as a triangular mesh of viewing regions (Sec 3.1.3). This parameterization

allows to generate (synthesize) the object part distribution at "any" location on the view sphere.

3.1.2 Part-Based Representation

The building blocks of our model are elliptically adapted local image patches. These are found using Hessian-Affine feature detector (*Mikolajczyk and Schmid*, 2002), Speeded Up Robust Features (SURF) detector (*Bay et al.*, 2008), and Maximally Stable Extremal Regions (MSER) (*Matas et al.*, 2002). A codebook obtained using a kmeans algorithm maps the Scale-invariant feature transform (SIFT) descriptors (*Lowe*, 1999) computed over these image patches into discrete codewords. After the pre-processing, each image patch is characterized by its appearance (codeword) Y and location (pixel coordinate) X.

In order to capture intra-class variation and mitigate viewpoint variation, we model the typical distribution of image patches appearance and location across object instances and nearby viewpoints by introducing the concept of part. We define parts as regions within an object that:

- Enclose discriminative features that are frequently observed across different instances of the object category.
- Form a more-or-less planar region on the physical object such that affine transformation becomes a good approximation when the part undergoes viewpoint changes (similarly to *Savarese and Fei-Fei* (2007)).

Similar to *Sudderth et al.* (2008), we propose to model the object as a mixture of parts. However, *Sudderth et al.* (2008) only model images captured from a single viewpoint. On the contrary, our approach models a 3D object category from a set of single viewpoint part-based models which can be combined so as to synthesize object parts at any location on the view sphere as we shall see next.

3.1.3 View Sphere Parameterizations

In most of the previous work, 3D object categorization assumed that a small number of discrete viewpoints for learning the object category (*Thomas et al.*, 2006; *Kushal et al.*, 2007; *Yan et al.*, 2007) were available. The reason for this mostly relies on the difficulty of providing dense viewpoint labels. We argue that a dense parameterization of view sphere yields a more accurate estimation of the object categorical model (Fig. 3.11-Center). Our goal is to learn a dense representation without viewpoint labels, as such human supervision is not only laborious and expensive to obtain, but also prone to errors because humans are not good at quantifying 3D viewpoints (*Palmer*, 1999).

Triangulating the view sphere and defined viewpoint parameters $\{T, S, A\}$. Let's define the set of key views as a finite set of view points on the view sphere. We shall explain later in detail how to obtain such key views (Sec. 3.1.5). Each adjacent triplet of key views defines triangle T inducing a triangular mesh parameterization on the view sphere. This parameterization enables the synthesis of new views within each triangle. Under the assumption that three key views are lying on the same plane (i.e. the views are parallel or rectified), where we denote such a plane as a *view plane*, a new view within T can be synthesized by introducing an interpolating (*morphing*) parameter S and a homography A (*Seitz and Dyer*, 1996; *Xiao and Shah*, 2004), called *post-warping transformation* (Sec. 3.1.5). Morphing parameter S is a 3D vector in the 3D simplex space that regulates the synthesis of the new view from the three key views. Homography A enables the correct alignment (registration) between the synthesized view and a query view. If the assumption of parallel views does not hold (e.g. the key views forming the triangle are not close enough) we can use feature correspondences across the key views to align the key views to the plane formed by the triangle (*Xiao and Shah*, 2004) (Sec. 3.1.5). Note that different triangles may correspond to view planes that are not mutually parallel (Fig. 3.1(c)). In this case, key

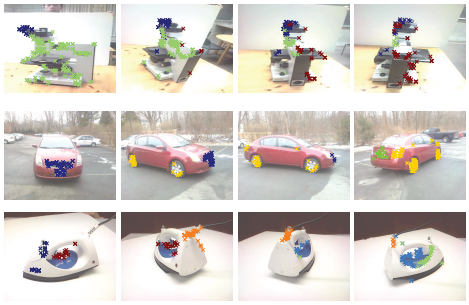


Figure 3.2: Examples of candidate parts of different object classes. Image patch features are denoted by "x". x's of the same color indicate that they belong to the same candidate part.

views may need to be re-aligned and their viewpoints adjusted through a homographic transformation H (*pre-warping transformation*). In conclusion, any viewpoint on the view sphere can be parameterized by a triangle T, interpolating parameter S within T, and post-warping transformation parameter A.

3.1.4 Part-Based Model over the View Sphere

We seek to integrate the part-based model with our parameterization of the views sphere. We propose to follow a probabilistic generative process as described in Sec. 3.1.6 to generate the parts. A key ingredient toward that goal is to automatically establish part correspondences across key views within each triangle T as well as across key views belonging to different triangles (Sec. 3.1.7). This allows us to generate (synthesize) geometrically consistent mixture of object parts within T using S and A. Notice that this mechanism allows us to generate infinite (dense) mixture of parts for all viewpoints on the view sphere from a finite set of G key views. Thus, limiting the complexity of our representation to $O(G)$. In other words, even if we only explicitly model a small set of G views (key views) among all possible views, a dense, multi-view 3D object category model is equivalently established. Notice that, unlike *Hoeim et al.* (2007); *Yan et al.* (2007); *Liebelt et al.* (2008), our model generalizes over the geometrical relationship of parts (the information related to the implicit 3D shape of the object category) across instances.

3.1.5 Key View Generation

The distribution of key views on the view sphere is a function of the object category. Such key views are extracted by acquiring a video sequence of one instance of the object category (Fig. 3.1(a)) using a hand held device such as a cellphone. We use the cellphone camera to take a short video clip by having a camera person walking around the object (while looking inward toward the object) and continuously lifting and lowering the camera. This created a zigzag (sinusoidal) trajectory on the view sphere that was roughly approximating a curve parameterized as $[a\ z] = k\ [a\ \sin(a)]$, where $a = azimuth; z = zenith$ angles on the view sphere; and k is a constant depends on the moving speed. Clearly this trajectory did not cover all angular locations on the view sphere but was sufficient to initialize the algorithm properly.

Given the short clip, we apply a Lucas-Kanade tracker (*Lucas and Kanade*, 1981) to obtain feature-level correspondences between every consecutive frames (Fig. 3.1(a)). Key views are selected sequentially given the feature-level correspondences. At the beginning, the first frame is selected as the initial key view. Then a new key view is selected once the ratio of the number of feature correspondences on the current frame to the number of feature correspondences on the previously selected key views falls below an empirical threshold (in this case 20%). The key view selection is a sequential procedure. The idea is that when this ratio drops below the given threshold, new elements of the object become visible so as to justify the introduction of a new key view. Given a typical video clip, we obtain ~ 100 key views.

3.1.6 Generative Process

In this section, we first describe the generative process of our proposed model, given the viewpoint parameters {T,S,A} (Fig. 8.2). Then we highlight the 3D geometrical constraints encoded in the model. In our approach, the appearance and location of each part K are modeled as a multinomial distribution over codeword Y

with parameter η, and Gaussian distribution over image coordinate X with parameter θ. The part proportion π is generated from a Dirichlet distribution with parameter α. The part-based representation is viewpoint dependent; hence, it is related to the way the view sphere is parameterized.

Generate part parameters (θ, η, π). There are three sets of parameters governing the distribution of each object part K under a specific viewpoint: part position (θ), part appearance (η) and part proportion (π), where each of them depends on the view triangle T and morphing parameter S.

Given a specific viewpoint parameters, $\{T, S\}$, we explore two alternative methods to generate part appearance η and location θ parameters from a set of the part parameters $\{\hat{\eta}, \hat{\theta}\}$ of key views in triangle T

- Nearest Neighbor Method:

  For each part, we set the parameters to be the same as the most similar key view $g^* = \underset{g}{\text{argmax}}\ s_g$ such that

  $$\eta = \eta_T(S) = \hat{\eta}_T^{g^*} \tag{3.1}$$

  $$\Sigma = \Sigma_T(S) = \hat{\Sigma}_T^{g^*} \tag{3.2}$$

  $$m = m_T(S) = \hat{m}_T^{g^*} \tag{3.3}$$

  where $\theta = \{m, \Sigma\}$ contains the mean (part center) and covariance (part shape) of a 2D Gaussian distribution, and $\{\hat{\eta}_T^{g^*}, \hat{\Sigma}_T^{g^*}, \hat{m}_T^{g^*}\}$ are the set of part parameters in key view g^* of the triangle T. A model using nearest neighbor method is referred as *nearest neighbor model*. A preliminary version of the nearest neighbor model is presented in *Sun et al.* (2009).

- View Morphing Method:

  The appearance parameter η and part shape parameter Σ are set in the same

way as in the nearest neighbor method. The part center m is instead generated as

$$m = m_T(S) = \sum_{g=1}^{3} \hat{m}_T^g \cdot S^g \tag{3.4}$$

where m is set to be equivalent to the linear interpolation of part centers $\{\hat{m}_T^1, \hat{m}_T^2, \hat{m}_T^3\}$ in the key views of triangle T.

According to view morphing technique (*Seitz and Dyer*, 1996; *Xiao and Shah*, 2004), the generated part center m is a valid synthesis of a new view, under the assumption that three key views are lying on the same plane (i.e. the views are parallel or rectified). Notice that different triangles may correspond to view planes that are not mutually parallel (Fig. 3.1(c)). In this case the key views may need to be re-aligned and their viewpoints adjusted approximately through a affine transformation H (*pre-warping transformation*) (in Sec. 3.1.7). A model using view morphing method is referred as *view morphing model*.

The part proportion parameter π is generated from a Dirichlet distribution $Dir(\alpha_T)$. π governs the likelihood of the different parts that will appear under this view, such that object part assignments Ks are sampled according to the distribution $Mult(\pi)$. For example, for a car model, π should be large for the wheel part in the side view but small in a frontal view.

Generate image features Given m, Σ, the position of each image feature $\hat{X}$ on the view plane is generated according to the Gaussian distribution $\mathcal{N}(m, \Sigma)$, where m and Σ denote the mean and covariance respectively. Recall the post-warping affine transformation A is introduced to align (register) the image plane and the view plane. The position of each image feature $\hat{X}$ on the view plane is equivalent to AX, where X is the position of each image feature on the image plane. The appearance of each image feature Y is generated from the Multinomial distribution $Mult(\eta)$, where η is

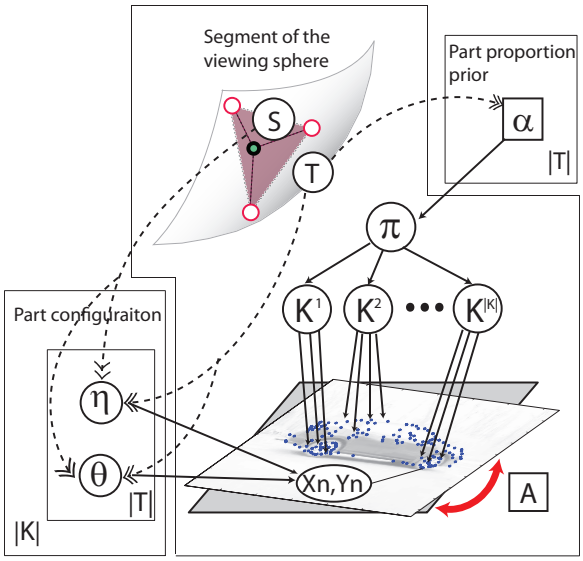


Figure 3.3: A schematic representation of our 3D object model. Each circular node represents a random variable, whereas each rectangular node represents a parameter. Solid arrows indicate conditional probability relationship between a pair of variables. Dashed arrows indicate the influence of the viewpoint triangle T and morphing parameter S on the variables. $\{X_n, Y_n\}$ indicate the position and appearance of an image patch feature. $K^1, K^2, \cdots$ are part types assigned to image features. π is the part proportion parameter governed by the Dirichlet parameter α. θ and η are the part position and part appearance parameters describing each image feature $\{X_n, Y_n\}$. Finally A is the post-warping affine transformation parameter. Note that the graphical model does not depict the 3D geometrical constraints used by this model (Sec. 3.1.7).

the multinomial distribution parameter that governs the proportion of the codewords.

Putting all the observable variables (X, Y, T, S) and latent variables (K, π) together with their corresponding parameters, we write down the joint probability of the model.

$$p(X, Y, T, S, K, \pi) = p(T)p(\pi|\alpha_T)p(S)$$
$$\prod_n^N \{p(X_n|\hat{\theta}, K_n, T, S, A)p(Y_n|\hat{\eta}, K_n, T, S)p(K_n|\pi)\} \tag{3.5}$$

where X_n, Y_n, K_n are the position, appearance, and part assignment of the n_{th} feature, and N is the number of features.

3.1.7 3D Geometric Constraints

A fundamental difference between our model and a typical mixture of parts model is the explicit correspondence among parts across different views. This correspondence guarantees that the synthesis (morphing) of parts at any view within triangle T is consistent under viewpoint transformation. To that end, we apply two types of 3D geometric constraints on the model.

A. Within triangle T constraints. Part configurations of key views within triangle T should be consistent with each other by an affine transformation $M^T_{i\to j}$, such that $M^T_{i\to j}\hat{m}^i_T = \hat{m}^j_T$. This information is encoded as a penalty term (C in Eq.3.26) in our Variational EM algorithm described in Sec. 3.2.1.2. Thus, our probabilistic model learning process favors those part configurations that have consistent geometrical relationships across different views. In practice, we use the tracks (feature correspondences) obtained by Lucas-Kanade algorithm between key views (in Sec. 3.1.5) V_i and V_j in a triangle T to estimate the affine transformation M.

B. Across triangle T constraints. As is shown in Fig. 3.1(c), each key view is shared by neighboring triangles $\{T_i, T_j, \ldots\}$. It is therefore important to make the correct correspondences between parts across the triangles. We estimate a transformation from T_i to T_j using an affine transformation operation $H_{i\to j}$ and enforce that $H_{i\to j}\hat{m}^{g_i}_{T_i} = \hat{m}^{g_j}_{T_j}$ for a specific part, where $\hat{m}^{g_i}_{T_i}$ is the part center of the key view g_i in triangle T_i. Intuitively, this means that we can establish part correspondences by enforcing the centers of the same part in different planes defined by neighboring triangles to share the same topological configuration in the coordinate system of the key view. Similarly, this constraint is encoded as a hard constraint (F in Eq.3.28) in the variational EM algorithm described in Sec. 3.2.1.2.

3.2 Learning

We have described the model in detail and can now describe how to learn its parameters and infer the latent variables. In detail, the learning procedure is set up as follows. Given images with object category labels and object bounding boxes, and a short video clip, the goal is to predict the viewpoint parameters $\{T, S, A\}$, infer the latent parts assignments Ks for features on each training image, and learn the part parameters $\{\hat{\eta}, \hat{\theta}\}$ for the model. In order to learn such a complex model, a good initialization is desirable. Thus, it ensures the learning algorithm to be effective. We use the video clip as a set of images (Sec. 3.2.1) to initialize the model. After initialization, the model is incrementally updated by sequentially applying variational EM algorithm on each training image of different object instances.

3.2.1 Initialization with A Video Clip

We initialize the learning process using a video sequence portraying a single instance of an object category from different viewpoints. We use the same video sequence as introduced in Sec. 3.1.5. We see two advantages. First it enables us to obtain a robust initial model of parts and viewpoints by using a single object instance, making it easier for the model to learn and adjust to the intra-class variations in the training stage as more training images become available. Second this method allows the algorithm to learn a 3D object categorical model with little human supervision, compared to all existing methods.

3.2.1.1 View Matching Exemplars:

As a first step of learning, we initialize viewpoint parameters $\{T, S, A\}$ of each frame by solving the following optimization problem

$$\{T, S, A\} = \underset{T,S,A}{\operatorname{argmax}} \sum_{i \in (\text{tracks in } T)} \|A\overline{X}_i - \overline{X}_{iT}S\|^2 \tag{3.6}$$

where $\overline{X}_{iT} \in R^{2\times 3}$ is the i_{th} feature correspondences shared among the three key views in triangle T, and $\overline{X}_i \in R^2$ is the corresponding feature in the frame. By solving the problem, the system finds the best parameters to synthesize the features in the frame using feature correspondences on the key views. Hence, in the following section, the viewpoint parameters $\{T, S, A\}$ of each frame are treated as observed variables in Fig.8.2. And all frames act like the exemplars in our view matching algorithm described in Sec. 3.2.2.1.

3.2.1.2 Part-Based Model:

We are now ready to estimate the hidden variables $\{K, \pi\}$ and part parameters $\{\theta, \eta, \alpha\}$ by maximizing the log marginal probability $\ln p(X, Y, T, S)$. Notice that $\ln p(X, Y, T, S)$ can be decomposed into

$$\ln p(X, Y, T, S) = \mathcal{L}_q + KL(q||p) \tag{3.7}$$

where $\mathcal{L}_q$ is the lower bound of log marginal probability $\ln p(X, Y, T, S)$ (Eq. 3.8), $q(.)$ is an arbitrary distribution over $\{K, \pi\}$. Notice that when $q(K, \pi) = p(K, \pi|X, Y, T, S)$,

$KL(q||p) = 0$ and $\ln p(X, Y, T, S) = \mathcal{L}_q$. The lower bound $\mathcal{L}_q$ can be written as

$$\begin{aligned}
\mathcal{L}_q &= \sum_K \int\int q(K, \pi) \\
&\quad \ln\{\frac{p(X, Y, T, S, K, \pi)}{q(K, \pi)}\}\, d\pi \\
&= E[\ln p(X, Y, T, S, K, \pi)] \\
&\quad -E[\ln q(K, \pi)] \\
&= E[\ln p(X|T, S, K)] + E[\ln p(Y|T, S, K)] + \\
&\quad E[\ln p(K|\pi)] + E[\ln p(\pi)] - \\
&\quad E[\ln q(K)] - E[\ln q(\pi)] + const. \qquad (3.8)
\end{aligned}$$

We use a mean-field variational distribution to approximate the true posterior $p(K, \pi|X, Y, T, S)$ as follows:

$$q(K, \pi) = q(\pi|\gamma) \prod_n^N q(K_n|\rho_n) \qquad (3.9)$$

where γ denote the variational parameter of the Dirichlet distribution which governs part proportion π, and $\rho_n \in R^{|K|}$ denotes the variational parameter for the part assignment variable K_n, which represents the probability that the n_{th} feature belongs to different parts.

Combing the lower bound L_q of the log marginal probability objective with the 3D geometric constraints introduced in Sec. 3.1.7, we formulate the learning problem as an optimization problem and solve it using a variational EM algorithm (*Blei*, 2004).

$$\text{maximize}_u \quad \lambda\mathcal{L}_q(u) - (1 - \lambda)\, C(u) \;\; \text{s.t.} \;\; F(u) = 0, \qquad (3.10)$$

where C is the within triangle constraint function (Sec. 3.1.7(A)), F is the across triangle constraint function (Sec. 3.1.7(B)), u denotes all the model and variational

parameters $\{\hat{\eta}, \hat{\theta}, \gamma, \rho_n\}$, and λ is the weight to balance the importance of the within T constraints C vs the lower bound $\mathcal{L}_q$. This optimal problem approximately finds the parameters which maximize the log marginal probability $\ln p(X, Y, T, S)$ and satisfy the constraints C, F.

The problem is solved using a variational EM algorithm (*Blei*, 2004) by iterating between the following M- and E-steps for model parameter and variational parameter updates, respectively.

M-Step: 1. Part appearance parameter $\hat{\eta}$ update.

$$\hat{\eta}_{TK}^{gw} = \frac{N_{TK}^{gw}}{N_{TK}^{g}} \tag{3.11}$$

$$N_{TK}^{gw} = \sum_{j \in (T_j = T,\ g^* = g)} \sum_{n \in (y_{nj} = w)} \rho_{nj}^{K} \tag{3.12}$$

$$g^* = \underset{g}{\text{argmax}}\ (s_g) \tag{3.13}$$

$$N_{TK}^{g} = \sum_{w} N_{TK}^{gw} \tag{3.14}$$

where $\hat{\eta}_{TK}^{gw}$ is the probability that codeword w appears for part K on key view g in triangle T, N_{TK}^{gw} is the sufficient statistics of the Multinomial distribution $Mult(\hat{\eta}_{TK}^{g})$, and ρ_{nj}^{K} is the probability that the feature n in image j belongs to part K.

2. Part center parameter $\hat{m}$ updates. The terms related to $\hat{m}$ in Eq. 3.10 are the expectation of the part location term $E[\ln p(X|T, S, K)]$, the within triangle constraints $C(u)$, and the across triangle constraints $F(u)$. We describe each term in detail below.

The expectation of $\ln p(X|T, S, K)$ taken over part assignment variation distribu-

tion $q(K|\rho)$ can be written as:

$$\begin{aligned} E[\ln p(X|T,S,K)] = \frac{1}{2}\sum_{K,j} N_{Kj}\{-\ln(\Sigma_{T_jK}(S_j)) \\ -(A_j\overline{x}_{Kj} - b_j - m_{T_jK}(S_j))^T \\ (\Sigma_{T_jK}(S_j))^{-1}(A_j\overline{x}_{Kj} - b_j - m_{T_jK}(S_j)) \\ -tr((\Sigma_{T_jK}(S_j))^{-1}A_jU_{Kj}A_j^T)\} + const \end{aligned} \tag{3.15}$$

where $\{m_{T_jK}(S_j), \Sigma_{T_jK}(S_j)\}$ are the generate part center (mean) and shape (covariance matrix) of the K path in the j_{th} object respectively, $\{A_j, b_j\}$ is the alignment transformation, N_{Kj}, $\overline{x}_{Kj}$, and U_{Kj} are the soft count of features belonging to part K in image j, the expected mean, and the expected covariance of image patches from part K in image j, which are defined as the following respectively:

$$N_{Kj} = \textstyle\sum_{n=1}^{N_j} \rho_{nj}^K \tag{3.16}$$

$$\overline{x}_{Kj} = \tfrac{1}{N_{Kj}} \textstyle\sum_{n=1}^{N_j} \rho_{nj}^K x_{nj} \tag{3.17}$$

$$U_{Kj} = \tfrac{1}{N_{Kj}} \textstyle\sum_{n=1}^{N_j} \rho_{nj}^K \left(X_{nj} - \overline{x}_{Kj}\right)\left(X_{nj} - \overline{x}_{Kj}\right)^T \tag{3.18}$$

$E[\ln p(X|T,S,K)]$ could be rewritten as the following:

$$\begin{aligned} E[\ln p(X|T,S,K)] = \frac{1}{2}\sum_{K,j} N_{Kj}\{-\ln(\hat{\Sigma}_{T_jK}^{g^*})- \\ tr((\hat{\Sigma}_{T_jK}^{g^*})^{-1}[A_j\ b_j\ \hat{\mathbf{m}}_{T_jK}]B_{Kj}[A_j\ b_j\ \hat{\mathbf{m}}_{T_jK}]^T) \\ -tr((\hat{\Sigma}_{T_jK}^{g^*})^{-1}A_jU_{Kj}A_j^T)\} + const \end{aligned} \tag{3.19}$$

$$g^* = \underset{g}{\text{argmax}}\ (s_g) \tag{3.20}$$

$$B_{Kj} = [\overline{x}_{Kj}; -1; -S_j][\overline{x}_{Kj}; -1; -S_j]^T \tag{3.21}$$

where $\hat{\mathbf{m}}_{TK} = [\hat{m}_{TK}^1 \hat{m}_{TK}^2 \hat{m}_{TK}^3]$, $\hat{m}_{TK}^g$ and $\hat{\Sigma}_{TK}^g$ are the mean and covariance of part K in key view g of the triangle T, respectively.

Finally, $E[\ln p(X|T,S,K)]$ can be rewritten as the following form using a set of compact statistics $N^g_{TK}, \hat{B}^g_{TK}, \hat{U}^g_{TK}$.

$$\begin{aligned} E[\ln p(X|T,S,K)] &= \frac{1}{2}\sum_{T,K,g} N^g_{TK}\left(-\ln(\hat{\Sigma}^g_{TK})\right) \\ &-tr((\hat{\Sigma}^g_{TK})^{-1}[I\ 0\ \hat{\mathbf{m}}_{TK}]\hat{B}^g_{TK}[I\ 0\ \hat{\mathbf{m}}_{TK}]^T) \\ &-tr((\hat{\Sigma}^g_{TK})^{-1}\hat{U}^g_{TK}) + const \end{aligned} \tag{3.22}$$

where $\hat{\Sigma}^g_{TK}$ are the mean (part center) and covariance (part shape) of part K in key view g of the triangle T, and $\{N^g_{TK}, \hat{B}^g_{T,K}, \hat{U}^g_{T,K}\}$ are a set of compact statistics, which are described below.

$$N^g_{TK} = \sum_{j\in\{j|T_j=T,\ g^*=g\}} \sum_n \rho^K_{nj} \tag{3.23}$$

$$\hat{B}^g_{TK} = \sum_{j\in\{j|T_j=T,\ g^*=g\}} N_{Kj} Z_j B_{Kj} Z_j^T \tag{3.24}$$

$$\hat{U}^g_{TK} = \sum_{j\in\{j|T_j=T,\ g^*=g\}} N_{Kj} A_j U_{Kj} A_j^T \tag{3.25}$$

where

$$Z_j = \begin{pmatrix} A_j & -b_j & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix}.$$

$\{N_{Kj}, B_{Kj}, U_{Kj}\}$ are the statistics related to empirical part weight, part center and part shape defined in Eq. 3.16, 3.21, and 3.18, respectively.

Notice that, given the part location parameters $\{\hat{m}, \hat{\Sigma}\}$, having the compact statistics $\{N^g_{TK}, \hat{B}^g_{TK}, \hat{U}^g_{TK}\}$ are sufficient to evaluate $E[\ln p(X|T,S,K)]$.

As is described in Sec. 3.1.7 (A), $C(u)$ is only related to the part center $\hat{m}$. And

the within triangle constraints contribute to the objective function as the following:

$$C(\hat{\mathbf{m}}) = \sum_{T,K} \sum_{(i,j)\in Q(T)} \|M^T_{i\to j}\hat{m}^i_{TK} - \hat{m}^j_{TK}\| \tag{3.26}$$

where $Q(T)$ is a set of pair of key views in triangle T

As described in Sec. 3.1.7 (B), $F(u)$ is also only related to the part center $\hat{m}$. And the across triangle constraints contribute to the constraints of the optimization problem as the following:

$$F(\hat{\mathbf{m}}) = \sum_{K} \sum_{(T_i,T_j)\in O} \sum_{(g_i,g_j)\in V(T_i,T_j)} \tag{3.27}$$

$$\|H_{i\to j}\hat{m}^{g_i}_{T_iK} - \hat{m}^{g_j}_{T_jK}\| \tag{3.28}$$

where O is a set of pairs of neighboring triangles, and $V(T_i, T_j)$ contains pairs of local index of key views in triangle T_i and T_j which share the same key view.

As a result, part center $\hat{m}$s are updated by solving the following optimization problem.

$$\begin{aligned} \text{maximize} \quad & \frac{1}{2}\lambda \sum_{K,T,g} \{-tr(\hat{\Sigma}^g_{TK})^{-1} \\ & [I\ 0\ \hat{\mathbf{m}}_{TK}]\, \hat{B}^g_{T,K}\, [I\ 0\ \hat{\mathbf{m}}_{TK}]^T\} + (\lambda - 1) \\ & \sum_{T,K} \sum_{(i,j)\in Q(T)} \|M^T_{i\to j}\hat{m}^i_{TK} - \hat{m}^j_{TK}\| \\ & \text{s.t. } F(\hat{m}) = 0 \end{aligned} \tag{3.29}$$

Since the objective is in a quadratic form of $\hat{\mathbf{m}}$ and the constraint $F(\hat{\mathbf{m}})$ is a linear equality constraint of $\hat{\mathbf{m}}$, $\hat{\mathbf{m}}$ could be estimated by solving a quadratic programming (QP) problem with linear equality constraints. In our implementation, we solve the QP problem using OOQP *Gertz and Wright* (2001) efficiently.

3. Part shape $\hat{\Sigma}$ updates Part shape parameter $\hat{\Sigma}$ is updated as follow. The

terms in Eq. 3.10 related to $\hat{\Sigma}^g_{TK}$ are

$$-N^g_{TK} \ln(\hat{\Sigma}^g_{TK}) \tag{3.30}$$

$$-tr((\hat{\Sigma}^g_{TK})^{-1}[I\ 0\ \hat{\mathbf{m}}_{TK}]\hat{B}^g_{TK}[I\ 0\ \hat{\mathbf{m}}_{TK}]^T) \tag{3.31}$$

$$-tr((\hat{\Sigma}^g_{TK})^{-1}\hat{U}^g_{TK}) \tag{3.32}$$

This objective function could be formulated as a semi-definite programming problem over the precision matrix $W^g_{TK} = (\hat{\Sigma}^g_{TK})^{-1}$. A closed form solution exists as below:

$$\hat{\Sigma}^g_{TK} = \left([I\ 0\ \hat{\mathbf{m}}_{TK}]\hat{B}^g_{TK}[I\ 0\ \hat{\mathbf{m}}_{TK}]^T + \hat{U}^g_{TK}\right)/N^g_{TK} \tag{3.33}$$

where $\{N^g_{TK}, \hat{B}^g_{TK}, \hat{U}^g_{TK}\}$ are compact sufficient statistics as define in Eq. 3.23-3.25.

E-Step: 1. Part proportion π update. For image j, we update the variational distribution $q(\pi_j|\gamma_j)$ of part proportion π_j by

$$\gamma_j = \alpha_{T_j} + N_j \tag{3.34}$$

where $N_j = [N_{1j}N_{2j}\ldots N_{Kj}\ldots N_{|K|j}]$. Note $N_j \in R^{|K|}$ is the sufficient statistics of the Dirichlet distribution $Dir(\gamma)$, where $|K|$ is the number of parts in our model.

2. Part assignment K update. We approximately model the distribution of the hidden variable K_n of each image feature using the variational distribution $q(K_n|\rho_n)$. In each step of the variational EM, we update ρ_n by maximizes $\mathcal{L}_q$. The terms in Eq. 3.10 related to ρ_n are $E[\ln p(X|T,S,K)]$, $E[\ln p(Y|T,S,K)]$, $E[\ln p(K|\pi)]$, and

$E[\ln q(K)]$. We have

$$\begin{aligned}\ln \rho_{nj}^{K} \quad \propto \quad & \frac{1}{2}\{-D\ln(2\pi) - \ln(\Sigma_{T_jK}(S_j)) \\ & -(A_jX_{nj} - b_j - m_{T_jK}(S_j))^T \\ & (\Sigma_{T_jK}(S_j))^{-1}(A_jX_{nj} - b_j - m_{T_jK}(S_j))\} \\ & + \ln \eta_{T_jK}^{Y_{nj}} + (\psi(\gamma_j^K) - \psi(\sum_{l=1}^{K}\gamma_j^l)) \end{aligned} \tag{3.35}$$

where $\eta_{T_jK} \in R^{|W|}$ is the part appearance parameter of $p(Y|\eta_{T_jK})$, $\gamma_j \in R^{|K|}$ is the variational parameter of $q(\pi_j|\gamma_j)$, $\sum_{K=1}^{|K|}\rho_{nj}^K = 1$, $|W|$ is the codebook size, and $|K|$ is the number of parts.

3.2.1.3 Obtaining Candidate Parts:

Since variational EM is an approximate inference algorithm, its solution is sensitive to the initial values of the parameters. We use a modified J-Linkage clustering algorithm (*Toldo and Fusiello*, 2008) to obtain candidate parts to initialize the variational parameter ρ^K of part assignment K. In *Toldo and Fusiello* (2008), given feature correspondences, the algorithm segments them into a number of planar regions so that the corresponding regions can be fitted by an unique affine transformation. Notice that tracks between every pair of views are equivalent to feature correspondences. We can therefore apply the J-Linkage algorithm to segment the tracks into planar regions for each pair of several neighboring key views. We then apply an agglomerative clustering algorithm to finalize the grouping of tracks to planar regions. Fig. 3.2 shows some sample results of this step.

3.2.2 Incremental learning with Unsorted Images

Having initialized our 3D object categorical model with a video clip of a single object instance, we can now complete model learning using a set of unsorted images

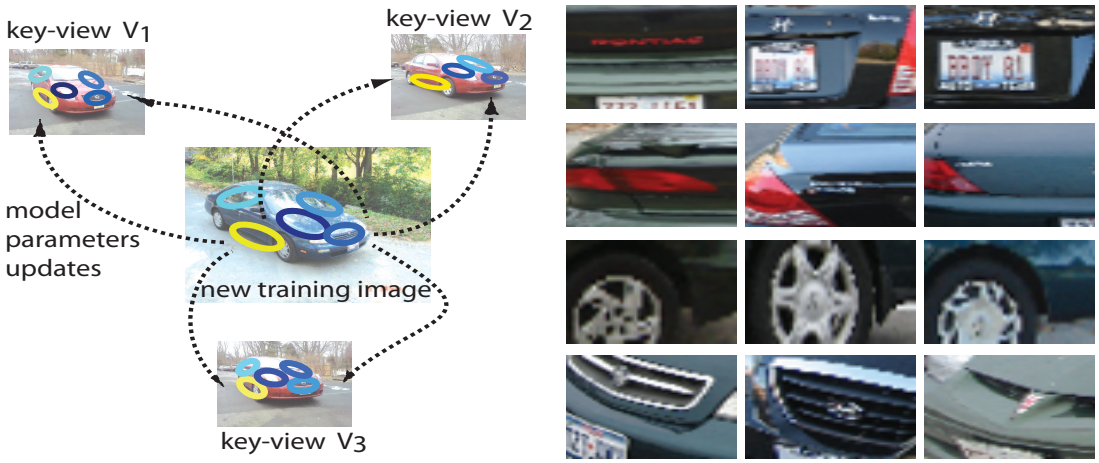


Figure 3.4: **Left.** Illustration of the updates for position parameter θ during incremental learning. As a new training image is assigned to the triangle T, new evidence on the sufficient statistics is produced which results in updating relevant model parameters of the key-views. **Right.** Examples of object parts are collected by automatically cropped image regions from different training images. Three examples are shown at each row for each part.

downloaded from the Internet. We propose an incremental learning procedure in the following two steps.

3.2.2.1 Viewpoint Estimation

Treating the video frames of the initial object instances as well as all the training images seen so far as view matching exemplars, we obtain the viewpoint $\{T, S, A\}$ of a new training image j by matching it to the closest exemplar according to a spatial pyramid matching algorithm. This is done in a similar image re-ranking scheme as proposed by *Li et al.* (2007)[1].

3.2.2.2 Part-Based Model Incremental Update

Given $\{T_j, A_j, S_j\}$ of the new training image j, we can now run the updates of part $\{\eta, m, \Sigma\}$ and variational $\{\pi, K\}$ parameters by applying variational EM (Sec. 3.2.1.2).

[1]Instead of the classification model used in *Li et al.* (2007), we use a KNN classifier with parzen window to do the matching.

- The part appearance parameter $\hat{\eta}_{TK}^{g^*w}$ can be updated according to Eq.3.11 by using the updated sufficient statistics $N_{T_jK}^{g^*w} \leftarrow N_{T_jK}^{g^*w} + \sum_{n\in(y_{nj}=w)} \rho_{nj}^{K}$, where w is the codeword index.

- The part center parameter $\hat{m}_{TK}^{g}$ and part shape parameter $\hat{\Sigma_{TK}^{g^*}}$ can be updated by solving Eq. 3.29 and Eq. 3.33, respectively using the updated sufficient statistics

$$N_{T_jK}^{g^*} \leftarrow N_{T_jK}^{g^*} + N_{Kj} \tag{3.36}$$

$$\hat{B}_{T_jK}^{g^*} \leftarrow \hat{B}_{T_jK}^{g^*} + N_{Kj} Z_j B_{Kj} Z_j^T \tag{3.37}$$

$$\hat{U}_{T_jK}^{g^*} \leftarrow \hat{U}_{T_jK}^{g^*} + N_{Kj} A_j U_{Kj} A_j^T \tag{3.38}$$

  Where N_{Kj}, B_{Kj}, and U_{Kj} are calculated using Eq. 3.16,3.21, and 3.18 respectively. Notice that the part centers of three key views in triangle T_j will be affected by the updated N_{Kj}, B_{Kj}.

- The variational part proportion parameter γ and variational part assignment parameter ρ are updated according to Eq. 3.34 and 3.35 respectively.

As a result, object parts for training images are extracted sequentially (some examples are shown in Fig.3.4-Right), and the part-based model is incrementally updated. Fig. 3.4-Left is a schematic illustration of how this is done.

3.2.3 Learning Summary

The overall learning algorithm combing variational EM algorithm with an incremental learning framework is described in Alg. 1

Algorithm 1 One iteration of the variational EM algorithm are shown below. The algorithm infers the part assignment and part proportion variational parameter ρ, γ of image j. Meanwhile, the model parameters η, m, Σ are updated given the new evidence from image j. The algorithm ends, when the lower bound of the likelihood (in Eq. 3.8) converges.

Given a new image j with matched viewpoint parameter $\{T_j, A_j, S_j\}$, feature location and appearance $\{X_{nj}, Y_{nj} | n \in 1 \sim N_j\}$, variational part assignment $\{\rho_{nj} | n \in 1 \sim N_j\}$ and proportion $\{\gamma_j\}$ parameters. We apply the following variational EM algorithm.

- Update statistics.

$$(N^{g^* w}_{T_j K})_j \leftarrow (N^{g^* w}_{T_j K})_{(j-1)} + \sum_{n \in (y_{nj}=w)} \rho^K_{nj} \tag{3.39}$$

$$(N^{g^*}_{T_j K})_j \leftarrow (N^{g^*}_{T_j K})_{(j-1)} + N_{Kj} \tag{3.40}$$

$$(\hat{B}^{g_*}_{T_j K})_j \leftarrow (\hat{B}^{g^*}_{T_j K})_{(j-1)} + N_{Kj} Z_j B_{Kj} Z_j^T \tag{3.41}$$

$$(\hat{U}^{g^*}_{T_j K})_j \leftarrow (\hat{U}^{g^*}_{T_j K})_{(j-1)} + N_{Kj} A_j U_{Kj} A_j^T \tag{3.42}$$

- M-step:

$$\hat{\eta}^{gw}_{TK} \sim \textit{Eq. 3.11} \tag{3.43}$$

$$\hat{m}^{g}_{TK} \sim \textit{Eq. 3.29} \tag{3.44}$$

$$\hat{\Sigma}^{g}_{TK} \sim \textit{Eq. 3.33} \tag{3.45}$$

- E-step

$$\gamma_j \sim \textit{Eq. 3.34} \tag{3.46}$$

$$\rho^K_{nj} \sim \textit{Eq. 3.35} \tag{3.47}$$

3.2.4 Comparison with previous methods

Our proposed method requires only object category labels and object bounding boxes supervision for the training images, where the degree of supervision is lower than most of the previous work (*Thomas et al.*, 2006; *Kushal et al.*, 2007; *Yan et al.*, 2007; *Chiu et al.*, 2007). Moreover, our model incorporates the view morphing technique so that it could represent theoretically an infinite number of viewpoints using a small set of key views. Therefore, our model could cover the view sphere more densely comparing to other 2D linkage-based methods (*Thomas et al.*, 2006; *Kushal et al.*, 2007; *Savarese and Fei-Fei*, 2007, 2008). Finally, the proposed incremental learning framework enables efficient model learning, where the time complexity is linear proportion to the number of training images.

(a) Car (b) Bicycle

Figure 3.5: Example images of car and bicycle categories from the 3D object category dataset. For each category, 8 images from different angles with randomly selected heights and scales are shown.

(a) Car (b) Bicycle

Figure 3.6: Example images of car and bicycle categories from the challenging Pascal VOC 2006 dataset. In each image, a yellow bounding box with object category annotation indicates the existence of the object.

(a)Binocular Microscope (b) Watch (c) Sewing Machine (d) Calculator (e) Swivel Chair (f) Teapot (g) Flashlight

Figure 3.7: Example images from household item dataset. Each row shows a different object category.

3.3 Applications

We have introduced a new probabilistic multi-view representation for 3D object categories. With minimal supervision, our algorithm is capable of learning the 3D structures of this part-based model across viewpoints. We test now how our model can be used to perform three challenging recognition tasks: object detection in cluttered background, viewpoint classification upon detection, and new viewpoint synthesis given a single test image.

Imagery from the real-world contains a mixture of both viewpoint and intra-class variation, unlike Caltech 101 and 256 where objects categories are captured from limited number of viewpoints. Recently proposed datasets are designed to capture both types of the variation. The 3D objects dataset (*Savarese and Fei-Fei*, 2007) contains 10 object instances for each object categories. Each object instance consists of images observed from 8 angles, 3 or 2 heights, and 3 scales. The dataset focuses on addressing viewpoint and intra-class variation. Hence, each image typically contains one object instance without occlusion (Fig 3.5). The pascal VOC 2006 dataset (*Everingham et al.*, 2006) not only captures viewpoints and intra-class variation, but also contains severe background clutter and object-object occlusion (Fig 3.6). In our experiments, we follow the experiment setting in *Savarese and Fei-Fei* (2007) and use 5 object instances from the 3D objects dataset to train the model and evaluate on the remaining object instances. For Pascal VOC 2006 dataset, we use the official

training and validation set to train the model and evaluate on the official testing set. Furthermore, we collect a set of images of 7 household object categories (e.g., watch, swing machine, microscope, swivel chair, calculator, flashlight, and teapot) from the ImageNet (*Deng et al.*, 2009) in order to evaluate our method on more diverse object categories. We use 5 object instances from such dataset to train the model and evaluate on the remaining object instances. All three datasets are used to evaluate object detection, viewpoint classification, and new viewpoint synthesis performance of our method.

3.3.1 Object class detection

A robust visual recognition system needs to detect and categorize real-world objects under arbitrary viewpoints. Having trained a part-based 3D object categorical model, we could now use this model to build a robust object category detector. Three datasets are used for evaluating this task: 8 object categories in the 3D Object dataset (*Savarese and Fei-Fei*, 2007), the car and bicycle categories in Pascal VOC 2006 dataset (*Everingham et al.*, 2006), and 7 categories in the household object dataset. We first describe briefly an object detector based on the learned object parts and 3D structure.

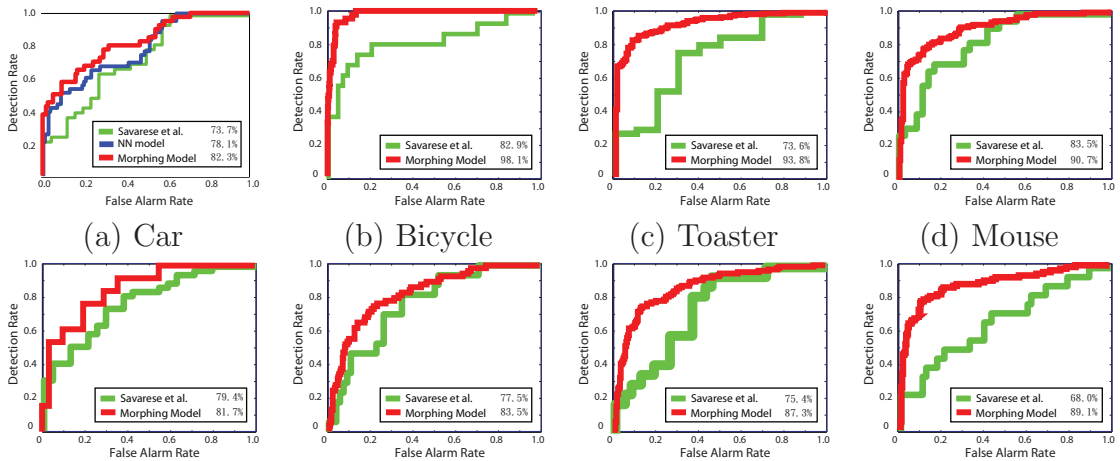
- **Pre-processing Training data:** Given all training images, we first obtain the image regions of the corresponding parts and viewpoint by the method described in Sec. 3.2.2. Each object bounding box is then re-scaled to have the same width. Within the bounding box, a set of patches are densely sampled with equal spacing, each of which has labels of the part and viewpoint that it belongs to. We also sample patches in regions outside the bounding box, each of which is assigned a background label.

- **Training Random Forest:** Given the patches with labels, we are able to train a random forest classifier, which discriminatively clusters patches having

similar labels into the same leaf node. In other words, each leaf node contains mostly patches from the same part observed from similar viewpoint, or mostly patches from background regions. Since leaf nodes correspond to different parts observed from different viewpoints, we use the learned model to lookup the relative position of the parts with respect to the object center. Notice these relative position information is used to vote for object center in the detection stage.

- **Fast Detection by a Generalized Hough Voting:** At the detection stage, patches are uniformly sampled from each image, and then passes down all the trees of the forest to find the codewords. Similar to the ISM (*Leibe et al.*, 2004), the saved relative position information for all codewords are use to cast votes for candidate object centers. Recall that the saved relative position is looked up from our learned generative model. Hence, Generalized Hough Voting technique is a fast way to evaluate the learned Gaussian part location model. To handle scale invariance, we also scale the test image to form a pyramid. The voting space is parameterized by object center location, object scale, and a coarse discretization of the view sphere (i.e., 8 different azimuth angles are used in our experiment). Finally, we obtain local maximum in the voting space as candidate object detections and apply non-maximum suppression using the 50% overlapping ratio criteria to remove redundant detections.

- **Verification Classifier:** In order to fuse evidences of part level together with object level information, an object verification classifier is trained using a linear SVM. An object is represented as a spatial pyramid of codewords (*Schmid*, 2006). The candidate object detections obtained from the part-based detector is reclassified by assigning the detection score as a linear combination of the original voting score and the object verification score.



(a) Car (b) Bicycle (c) Toaster (d) Mouse

(e) Iron (f) cellphone (g) Stapler (h) Shoe

Figure 3.8: Object detection results using the 3D objects dataset (*Savarese and Fei-Fei*, 2007). We use ROC curves to show the detection results. In each panel, view morphing model (red line) shows a performance of 82.3% measured by area under the curve (AUC), compared to 73.7% by using *Savarese and Fei-Fei* (2007) (green line), and 78.1% by using the nearest neighbor model (NN model)(*Sun et al.*, 2009) (blue line). The AUC of other 7 object classes are compared between view morphing model (red line) and *Savarese and Fei-Fei* (2007) (green line).

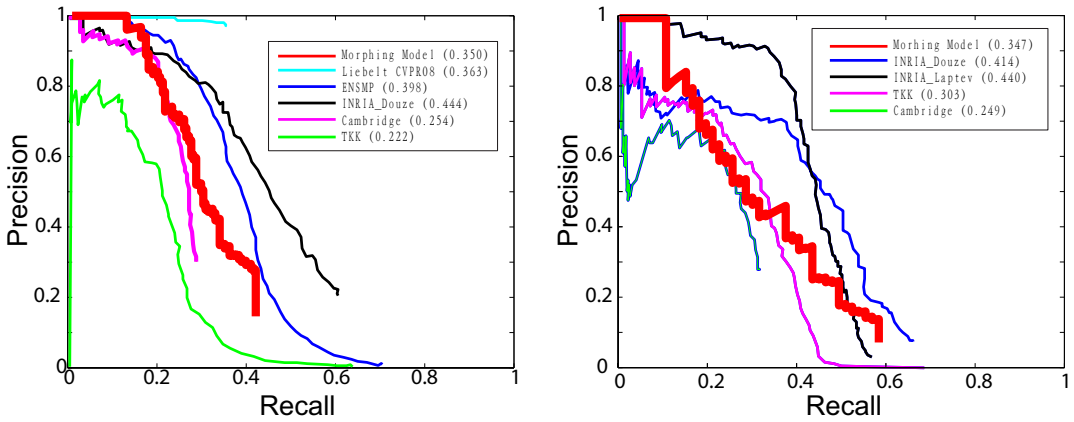
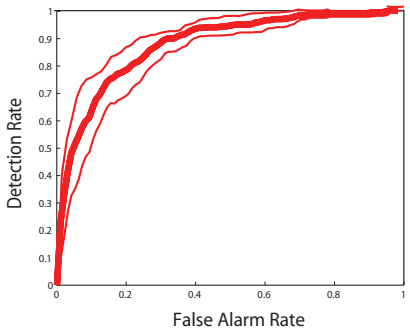


Figure 3.9: Object detection results using the Pascal VOC06 dataset (*Everingham et al.*, 2006). **Left.** Object detection using the Pascal VOC06 car dataset. **Right.** Object detection using the Pascal VOC06 bicycle dataset. We follow the protocol of Pascal VOC object detection challenge (50% overlap criteria) and use precision-recall curves to show the results of our view morphing model (red line) compared with *Liebelt et al.* (2008) and the detection result of the 2006 challenges (*Everingham et al.*, 2006)-INRIA_Douze , (*Everingham et al.*, 2006)-INRIA_Laptev, (*Everingham et al.*, 2006)-TKK, (*Everingham et al.*, 2006)-Cambridge, and (*Everingham et al.*, 2006)-ENSMP. Average precision (AP) scores are shown in the legends.



Object Category	AUC
Watch	84.9
Sewing Machine	98.1
Microscope	87.9
Swivel Chair	91.2
Calculator	97.2
Flashlight	87.1
Teapot	86.4

Figure 3.10: **Left.** Object detection results using the 7 household object categories dataset. The thick red line shows the average ROC curve, whereas the thin red lines show the standard deviation over 7 classes. Average Area Under the Curve (AUC) score is 90.1%. **Right.** AUC score for each of the household object category is shown at the 2nd column of each row.

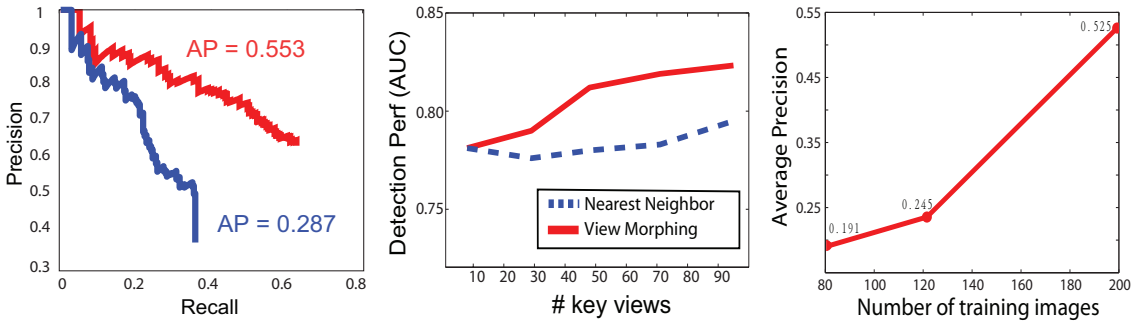


Figure 3.11: **Left.** Object detection with or without using the object parts. We use a car detection task (3D objects dataset (*Savarese and Fei-Fei*, 2007)) to show the performance difference between an object detector using the object parts learned by the 3D model (red line) and an object detector built without the object parts (blue line). **Center.** Effect of view synthesis on improving detection accuracy at different number of key views via learning with the morphing parameter S. We demonstrate this by showing a binary detection task result (measured by AUC) versus the number of key views used by the model. View morphing model (red solid line) is compared with a nearest neighbor model (blue dashed line). **Right.** Effect of incremental learning on improving detection accuracy at different number of training images.

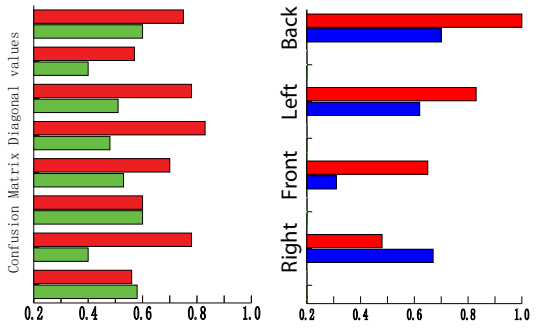
Fig. 3.8 compares the Receiver Operating Characteristic (ROC) detection results of our method with other state-of-the-art algorithms on the 3D objects dataset (*Savarese and Fei-Fei*, 2007). View morphing model consistently outperforms *Savarese and Fei-Fei* (2007) on all 8 object categories of the 3D objects dataset. Fig. 3.9 compares the precision-recall detection results of our view morphing model with other state-of-the-art algorithms on car and bicycle of Pascal VOC06 dataset (*Everingham*

et al., 2006). View morphing model shows comparable results to most of the state-of-the-art methods on the Pascal VOC06 dataset. Fig. 3.10 shows detection results on the new 7 household object categories dataset. These items have significantly different image features compared to the often used car and bicycle datasets. We show very promising detection results in all seven object categories. Some example detection results are shown in Fig. 3.13.

An important contribution of our work is to propose a method that is able to learn parts and associate viewpoint on the view sphere automatically and in turn use them for building an object category detector. In an object detection experiment using the 3D objects dataset (car category), we show that an object detector built by using these proposed object parts and viewpoint significantly outperforms an object detector that does not use the parts and viewpoint information (Fig. 3.11-Left).

In a separate experiment, we examine the effect of using dense viewpoint representation and view morphing framework for building 3D object models. Fig. 3.11-Center shows a detection experiment on car category of the 3D object dataset measured by Area Under the ROC Curve (AUC). View morphing model (red curve) is compared with a nearest neighbor model (blue curve). We observe two trends. For both of these models, as the number of viewpoints increases during training, the detection performance increases. But view morphing model performs consistently better than nearest neighbor model even given the same number of viewpoints. Since the view morphing model can better capture the relative part location observed under arbitrary viewpoint on the view sphere using the morphing parameter S than the nearest neighbor model. In other words, even if the view morphing model maintains a limited number of key-views, it is capable of representing intermediate views to mitigate the unavoidable discrepancies existing in the discretized representation.

Finally, we evaluate the effect of incremental learning by using detection of car in the 3D objects dataset. Fig.3.11-Right shows that as the number of training images



Object Category	Avg. Acc
Watch	61.9
Sewing Machine	71.4
Microscope	63.9
Swivel Chair	58.6
Calculator	69.2
Flashlight	68.4
Teapot	60

Figure 3.12: Viewpoint classification results: **Left.** 8-view classification of the 3D objects car dataset. We compare our view morphing model (red bar) with (*Savarese and Fei-Fei*, 2008) (green bar). **Center.** 4-view classification of the Pascal VOC06 car dataset. Our view morphing model (red bar) is compared with *Sun et al.* (2009) (blue bar). **Right.** Viewpoint classification accuracy for the household objects dataset.

increases, our view morphing model is capable of taking advantage of the additional data and continues to achieve higher performances.

3.3.2 Object viewpoint classification

After the detection stage, the candidate object detections are associated with a coarse location on the view sphere. Our view morphing model is capable of predicting a more precise viewpoint of a query object by estimating $\{T, S, A\}$ using the viewpoint estimation method described in Sec. 3.2.2. We evaluate our view morphing model using three datasets: car category in the 3D objects dataset (*Savarese and Fei-Fei*, 2007), car category in Pascal VOC06 dataset (*Everingham et al.*, 2006) and seven categories in the household objects dataset.

The results of the first two are shown in Fig. 3.12-Left&Center. Whereas results of household objects are shown in Fig.3.12-Right. Notice that the numbers of viewpoints that we evaluated are dataset dependent. For 3D objects, the ground truth labels provide 8 viewing angles, 3 scales and 2 heights, and the number of images is evenly distributed on the view sphere. For 7 household object categories dataset, the number of images is also evenly distributed on the view sphere. For Pascal VOC06, there are more diverse viewpoints, but the number of images is not evenly distributed on the

(a) Bicycle (b) Swivel Chair (c) microscope

(d) Car (e) Watch (f) Iron

(g) Teapot (h) Flashlight (i) Calculator

Figure 3.13: Examples of viewpoint estimation for bicycle (*Savarese and Fei-Fei*, 2007; *Everingham et al.*, 2006), swivel chair, microscope, car (*Savarese and Fei-Fei*, 2007; *Everingham et al.*, 2006), watch, iron, teapot, flashlight, and calculator. Blue arrows indicate the viewpoint T for the detected object (in red bounding box). Green bounding box indicates correct detections of the objects, but in a different viewpoint.

view sphere and the ground truth labels only contain 4 (i.e., front, back, left, and right.) viewpoints. For evaluation convenience, we discretize all views into 8 canonical views when evaluating the performances on 3D object and household object dataset, and all views into 4 canonical views when evaluating the performances on Pascal VOC 2006 dataset. On the 3D objects dataset, our view morphing model significantly outperforms *Savarese and Fei-Fei* (2008), largely due to its richer representation

Figure 3.14: New views can be synthesized given a single test image. The right column of each row indicates the original test image. The left two columns are two synthesized views.

(Fig. 3.12-Right). On the Pascal dataset, our view morphing model also achieves better classification accuracy than the nearest neighbor model (Fig. 3.12-Center). Finally, we obtain reasonably accuracy classification accuracy on the household object dataset (Fig. 3.12-Right). We show examples of the viewpoint classification results in Fig.3.13.

3.3.3 Viewpoint synthesis

After both detection and viewpoint classification stage, a candidate object detection is associated with the part locations and viewpoint parameters $\{T, S, A\}$. Recall that the view morphing model can generate the part configuration of any view specified by the viewpoint parameters $\{T, S, A\}$. Therefore, given the extracted parts and estimated viewpoint parameters $\{T, S, A\}$, we are able to synthesize any new view by calculating the part configuration of a specific viewpoint parameters $\{T, S, A\}$. We show in Fig. 3.14 several synthesized views of test images. Notice, the model can not generate detail appearance of unobserved parts. Therefore, all examples in Fig. 3.14 are views which contain only the observed parts in the original testing images.

3.4 Conclusion

In this chapter, we have proposed a 3D object categorical model based on a dense, multi-view representation of the view sphere. A morphing parameter S is introduced to allow our model to recognize and synthesize unseen views. Our experiments show promising results in object detection, viewpoint classification and synthesis tasks. For future work, we would like to incorporate a more discriminative learning process into the model building step, as well as to combine viewpoint synthesis into incremental learning framework to generate virtual training images so as to maximize the usage of the data.

CHAPTER IV

Models for 3D Object Shape Inference

Detecting objects and estimating their geometric properties are crucial problems in many application domains such as robotics, autonomous navigation, high-level visual scene understanding, surveillance, gaming, object modelling, and augmented reality. For instance, if one wants to design a robotic system for grasping and manipulating objects, it is of paramount importance to encode the ability to accurately estimate object orientation (pose) from the camera view point as well as recover structural properties such as its 3D shape. This information will help the robotic arm grasp the object at the right location and successfully interact with it. Moreover, if one wants to augment the observation of an environment with virtual objects, the ability to reconstruct visually pleasing 3D models for object categories is very important.

This chapter addresses the above needs, and tackles the following challenges: i) Learn models of object categories by combining view specific depth maps along with the associated 2D image of object instances of the same class from different vantage points. Depth maps with registered RGB images can be easily collected using sensors such as Kinect Sensor (*Microsoft Corp. Redmond WA*, 2010). We demonstrate that combining imagery with 3D information helps build richer models of object categories that can in turn make detection and pose estimation more accurate. ii) Design a coherent and principled scheme for detecting objects and estimating their poses from

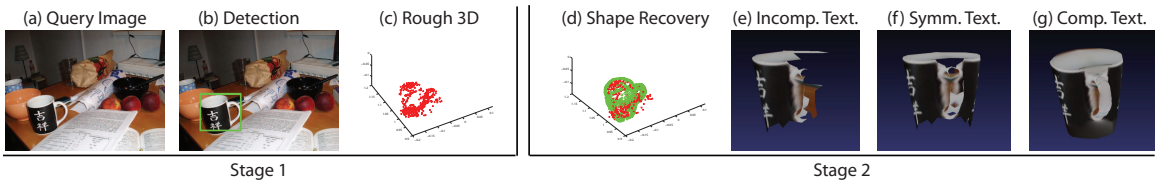


Figure 4.1: Key steps of our reconstruction algorithm: (a) Single query 2D image; (b) Detected object; the bounding box indicates the location where the object has been estimated in the image; Our proposed Depth Encoded Hough Voting (DEHV) detector can be used to recognize object class label, roughly estimate the object pose (i.e., object orientation in the camera reference system), and automatically reconstructs surface elements (3D points) in the camera reference system (c). As figure shows, the reconstruction is clearly partial and incomplete; (d) Shape recovery: by using the estimated object class label and pose, we propose a novel 2D+3D ICP algorithm to register the reconstructed surface elements with one of the 3D models that is available in training; this allows to infer the object 3D structure in regions that are not visible from the query image. (e) Texture mapping: after performing 3D shape registration, we texture map image texture to the 3D shape model; again, the object texture is incomplete as we cannot map image texture to occluded surface elements; (f) Texture completion: we use the fact that some object categories are symmetric to transfer image texture to the occluded regions; (g) Remaining un-textured surfaces elements are completed using image compositing methods inspired by *Tao et al.* (2010).

either just a single image (when no depth maps are available in testing) (Fig. 4.1(b)), or a single image augmented with depth maps (when these are available in testing). In the latter case, 3D information can be conveniently used by the detection scheme to make detection and pose estimation more robust than in the single image case. iii) Have our detection scheme reconstruct the 3D model of the object from just a single uncalibrated image (when no 3D depth maps are available in testing) (Fig. 4.1(c-g)) and without having seen the object instance during training.

In this chapter, we propose a two stages approach to address the above challenges (Fig. 4.2). In the first stage, our approach seeks to i) detect the object in the image, ii) estimate its pose, and iii) recover a rough estimate of the object 3D structure (if no depth maps are available in testing). This is achieved by introducing a new formulation of the Implicit Shape Model (ISM) (*Leibe et al.*, 2004) and generalized Hough voting scheme (*Ballard*, 1981). In our formulation, depth information is in-

corporated into the process of learning distributions of object image patches that are compatible with the underlying object location (shape) in the image plane. We call our scheme *DEHV - Depth-Encoded Hough Voting scheme* (Sec. 4.2.1). DEHV addresses the intrinsic weaknesses of existing Hough voting schemes (*Leibe et al.*, 2004; *Gall and Lempitsky*, 2009; *Maji and Malik*, 2009; *Ommer and Malik*, 2009) where errors in estimating the scale of each image object patch directly affects the ability of the algorithm to cast consistent votes for the object existence. To resolve this ambiguity, we take advantage of the interplay between the scale of each object patch in the image and its distance (depth) from the corresponding physical patch attached to the 3D object, and specifically use the fact that objects (or object parts) that are closer to the camera result in image patches with larger scales. Depth is encoded in training by using available depth maps of the object from a number of view points. At recognition time, DEHV is applied to detect objects (Fig. 4.1(b)), estimate their pose, and simultaneously infer the their 3D structure given hypotheses of detected objects (Fig. 4.1(c)). The object 3D structure is inferred at recognition time by estimating (decoding) the depth (distance) of each image patch involved in the voting from the camera center. Critically, depth decoding can be achieved even if just a single test image is provided. If depth maps are available in testing, the additional information can be used to further validate if a given detection hypothesis is correct or not. We summarize the inferred quantities in Table 4.1 and the required supervision in Table 4.2. Notice that the inferred object 3D structure from stage one is partial (it does not account for the portions of the object that are not visible from the query image) and sparse (it only recovers depth for each voting patch).

The goal of the second stage is to obtain a full 3D object model where both 3D structure and albedo properties (texture) are also recovered. In the second stage, the information inferred from stage one (object location in the image, scale, pose, and rough 3D structure) is used to obtain a full 3D model of the object. Specifically,

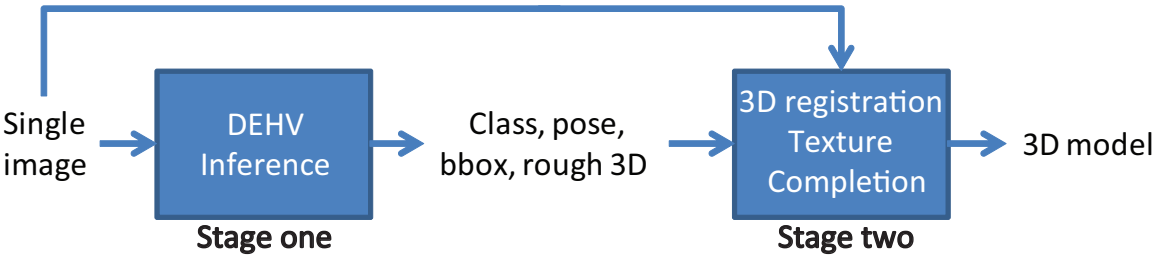


Figure 4.2: Flow chart showing the process of our proposed system.

Single Image		
	Depth in testing	No depth in testing
Inferred quantities	object class, location, scale, pose	object class, location, scale, pose, depth map

Table 4.1: Estimated quantities in Stage 1.

Stage 1	Stage 2
-Images of object from multiple views -Depth maps of object from multiple views -Bounding boxes and pose annotation	-List of CAD Models

Table 4.2: Required degree of supervision in training for each stage.

we consider an 3D modelling stage where a full 3D model of the object is obtained by 3D shape recovery and texture completion (Sec. 4.2.2). We carry out 3D shape recovery (i.e., infer shape from the unseen regions) by: i) utilizing 3D shape exemplars from a database of 3D CAD models which can be collected from *Shilane et al.* (2004) and other online 3D warehouses, or obtained by shape from silhouette (*Laurentini*, 1994); ii) applying a novel 2D+3D iterative closest point (ICP) matching algorithm which jointly registers the best 3D CAD model to the inferred 3D shape and the occlusion boundaries of back projected 3D CAD model to object contours in the image. By choosing the best fit, our system obtains a plausible full reconstruction

of the object 3D shape (Sec. 4.2.3) (Fig. 4.1(d)). Object appearance is rendered by texture mapping the object image into the 3D shape. Such texture is clearly incomplete as non-visible object surface areas cannot be texture mapped (Fig. 4.1(e)). Thus, we perform texture completion by: i) transferring texture to such non-visible object surface areas by taking advantage of the fact that some object categories are symmetric (when possible) (Fig. 4.1(f)); ii) using an error-tolerant image compositing technique inspired by *Tao et al.* (2010) to fill the un-textured regions (i.e., holes) (Sec. 4.2.4) (Fig. 4.1(g)). We summarize the required supervision in Table 4.2.

Extensive experimental analysis on a number of public datasets (including car Pascal VOC07 (*Everingham et al.*, 2007), mug ETHZ Shape (*Ferrari et al.*, 2008a), mouse and stapler 3D object dataset (*Savarese and Fei-Fei*, 2007)), an two in-house datasets (comprising at most 5 object categories), where ground truth 3D information is available, are used to validate our claims (Sec. 6.4). Experiments with the in-house datasets demonstrate that our DEHV scheme: i) achieves better detection rates (compared to the traditional Hough voting scheme); further improvement is observed when depth maps are available in testing; ii) produces convincing 3D reconstructions from single images; the accuracy of such reconstructions have been qualitatively assessed with respect to ground truth depth maps; iii) achieves accurate 3D shape recovery and visually pleasing texture completion results. Experiments with public datasets demonstrate that our DEHV successfully scales to different types of categories and works in challenging conditions (severe background clutter, occlusions). DEHV achieves state of the art detection results on several categories in ETHZ Shape dataset (*Ferrari et al.*, 2008a), and competitive pose estimation results on 3D object dataset (*Savarese and Fei-Fei*, 2007). We also evaluate the accuracy of shape completion and quality of the texture completion on the 3D modelling dataset (Sec. 4.2.2). Finally, we show typical results demonstrating that DEHV is capable to produce convincing 3D reconstructions from single uncalibrated images using Pas-

cal VOC07 dataset (*Everingham et al.*, 2007), ETHZ Shape dataset (*Ferrari et al.*, 2008a), and 3D object dataset (*Savarese and Fei-Fei*, 2007) in Fig. 4.15 and 4.13.

The rest of the chapter is organized as follows. We first give brief overview of works on 3D object modelling in Sec. 4.1. Then, we introduce our method in Sec. 4.2. Finally, we report experimental results in Sec. 6.4.

4.1 Related Works on 3D Object Modelling

In this section, we give a brief overview of recent research on 3D object and scene modelling from images. For an overview of methods for object detection and view point estimation, please refer to Chapter II, Sec. 2.1. Approaches for 3D object or scene modelling are often referred to as image-based modelling techniques (IBM). Starting from early work by *Debevec et al.* (1996); *Pollefeys et al.* (2004), IBM techniques have been recently employed for successfully modelling large scale environments such as city environments from large collection of images on the internet (*Snavely et al.*, 2006; *Agarwal et al.*, 2009). IBM techniques often require different degrees of human intervention (*Debevec et al.*, 1996) or the assumptions that special equipments are available and/or cameras are calibrated (*Dick et al.*, 2004; *Teller et al.*, 2003).

Even if outstanding results have been produced, many of these methods make the basic assumption that several images (portraying the object in the scene from different view points) are available. However, this is not always the case. Recovering scene geometry from a single view has been initially explored under the assumption of having users guiding the reconstruction (*Horry et al.*, 1997; *Liebowitz et al.*, 1999) or augmenting the photograph with additional 3D data (*Kopf et al.*, 2008). Recently, researchers have proposed to apply machine learning methodologies for resolving the 3D-2D mapping ambiguity and obtaining convincing reconstructions of outdoor (*Saxena et al.*, 2009; *Hoiem et al.*, 2005b) and indoor scenes (*Lee et al.*, 2009; *Wang et al.*,

2010; *Hedau et al.*, 2010; *Schwing et al.*, 2012) from just one single image.

Alternative techniques have been proposed for modelling specific 3D objects (rather than scenes or environments). Again, depending on the application and the level of accuracy that one aims to achieve, researchers have proposed methods employing either external lighting sources such as lamps (*Bouguet and Perona*, 1995; *Savarese et al.*, 2006a), projectors (*Rusinkiewicz et al.*, 2002), lasers (*Levoy et al.*, 2000), or a number of calibrated (*Kutulakos and Seitz*, 2000) or uncalibrated views obtained using external devices such as turntables (*Mendonça et al.*, 2000). A recent survey nicely summarizes most relevant works (*Seitz et al.*, 2006) from an almost endless literature on this topic. Recently, *Prasad et al.* (2010) have proposed a method to reconstruct deformable object classes from multiple and unordered images. Due to the absence of reliable point correspondences across deformable object instances, class-specific curve correspondences need to be manually selected.

The reconstruction of an underlying 3D shape model is not always a necessary step if one wants to render the environment appearance from just images. These methods fall under the name of image based rendering approaches (IBR). Works by *McMillan and Bishop* (1995); *Levoy and Hanrahan* (1996); *Aliaga et al.* (2003); *Zitnick et al.* (2004) are among the most notable examples. The lack of the underlying 3D shape model, however, makes it harder for these techniques to be used in applications where virtual worlds are to be augmented with the reconstructed models.

As opposed to indoor or outdoor scenes where cues such as vanishing lines or texture foreshortening are available, fewer methods have been proposed for recovering 3D models of objects from a single image. Researchers mostly focused on recovering 3D shape models from object contours (silhouettes) extracted or identified on a single image either automatically (*Prasad and Fitzgibbon*, 2006; *Colombo et al.*, 2005) or through some level of user intervention (*Chen et al.*, 2008; *Karpenko and Hughes*, 2006; *Jiang et al.*, 2009). These methods, however, often assume topological proper-

ties of objects such as smoothness, convexity, or cylindrical symmetry or heavily relies on user intervention. In our work, we do not want our query objects to be subject to these constraints. Rather, similar to *Saxena et al.* (2009); *Hoiem et al.* (2005b), we advocate the usage of machine learning for solving the daunting task of single view object reconstruction with arbitrary topology and minimal user intervention. Very recently, *Thomas et al.* (2007); *Sun et al.* (2010b); *Arie-Nachimson and Basri* (2009); *Oswald et al.* (2009) have shown the ability to reconstruct sparse/partial 3D object points from a single image. However, none of these methods have been extensively tested so as to demonstrate that realistic 3D models of objects can be obtained.

4.2 Our Method

To summarize, our method can be roughly decomposed in a recognition/reconstruction stage and a 3D modelling stage.

In the recognition/reconstruction stage, Depth-Encoded-Hough-Voting detectors (DEHV) (*Sun et al.*, 2010b), trained with both object 3D shape and local diagnostic appearance information, identifies object' locations and classes, and recovers approximate and partial 3D structure information from a single query image (Sec. 4.2.1) (Fig. 4.1(a-c)).

Because we obtain only a partial reconstruction - object surface that is not visible from the query image cannot be reconstructed at this stage. Thus, we consider a 3D modelling stage where a full 3D model of the object is obtained by 3D shape recovery and texture completion (Sec. 4.2.2) (Fig. 4.1(d-g)).

4.2.1 Stage 1: Depth-Encoded Hough Voting

In recognition techniques based on hough voting (*Ballard*, 1981) the main idea is to represented the object as a collection of parts (patches) and have each part to cast votes in a discrete voting-space. Each vote corresponds to a hypothesis of object

location x and class O. The object is identified by the conglomeration of votes in the voting space $V(O, x)$. $V(O, x)$ is typically defined as the sum of independent votes $p(O, x, b_j, s_j, l_j)$ from each part j, where l_j is the location of the part, s_j is the scale of the part, and b_j is the part appearance.

Previously proposed methods (*Leibe et al.*, 2004; *Gall and Lempitsky*, 2009; *Maji and Malik*, 2009; *Ommer and Malik*, 2009) differ mainly by the mechanism for selecting good parts. For example, parts may be either selected by an interest point detector (*Leibe et al.*, 2004; *Maji and Malik*, 2009), or densely sampled across many scales and locations (*Gall and Lempitsky*, 2009); and the quality of the part can be learned by estimating the probability (*Leibe et al.*, 2004) that the part is good or discriminatively trained using different types of classifiers (*Maji and Malik*, 2009; *Gall and Lempitsky*, 2009). In this chapter, we propose a novel method that uses 3D depth information to guide the part selection process. As a result, our constructed voting space $V(O, x|D)$, which accumulates votes for different object classes O at location x, depends on the corresponding depth information D of the image. Intuitively, any part that is selected at a wrong scale can be pruned out by using depth information. This allows us to select parts which are consistent with the object physical scale. It is clear that depending on whether object is closer or further, or depending on the actual 3D object shape, the way how each patch votes will change (Fig. 4.3).

In detail, we define $V(O, x|D)$ as the sum of individual probabilities over all observed images patches at location l_j and for all possible scales s_j, i.e,

$$\begin{aligned} V(O, x|D) &= \sum_j \int p(O, x, b_j, s_j, l_j|d_j)\, ds_j \\ &= \sum_j \int p(O, x|b_j, s_j, l_j, d_j) p(b_j|s_j, l_j, d_j) \\ &\quad p(s_j|l_j, d_j) P(l_j|d_j)\, ds_j \end{aligned} \tag{4.1}$$

where the summation over j aggregates the evidence from individual patch location,

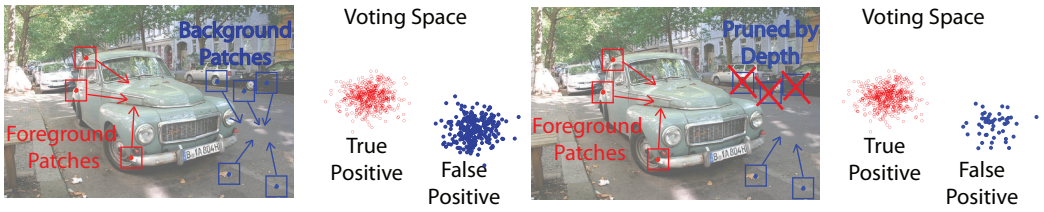


Figure 4.3: Top panel shows that patches associated to the actual object parts (red boxes) will vote for the correct object hypothesis (red dots) in the voting space on the right. However, parts from the background or other instances (cyan boxes) will cast votes that may create a false object hypothesis (green dots) in the voting space. Bottom panel shows that given depth information, the patches selected at a wrong scale can be easily pruned. As a result, the false positive hypothesis will be supported by less votes.

and the integral over s_j marginalizes out the uncertainty in scale for each image patch. Since b_j is calculated deterministically from observation at location l_j with scale s_j, and we assume $p(l_j|d_j)$ is uniformly distributed given depth, we obtain:

$$\begin{aligned} V(O,x|D) &\propto \sum_j \int p(O,x|b_j,s_j,l_j,d_j)p(s_j|l_j.d_j)ds_j \\ &= \sum_{j,i} \int p(O,x|C_i,s_j,l_j,d_j)p(C_i|b_j) \\ &\quad p(s_j|l_j,d_j)ds_j \end{aligned} \tag{4.2}$$

Here we introduce codebook entry C_j, matched by feature b_j, into the framework, so that the quality of a patch selected will be related to which codeword it is matched to. Noting that C_j is calculated only using b_j and not the location l_j, scale s_j, and depth d_j, we simplify $p(C_j|b_j,s_j,l_j,d_j)$ into $p(C_j|b_j)$. And by assuming that $p(O,x|.)$ does not depend on b_j given C_j, we simplify $p(O,x|C_j,b_j,s_j,l_j,d_j)$ into $p(O,x|C_j,s_j,l_j,d_j)$.

Finally, we decompose $p(O,x|.)$ into $p(O|.)$ and $p(x|.)$ as follows:

$$\begin{aligned} V(O,x|D) &\propto \sum_{j,i} \int p(x|O,C_i,s_j,l_j,d_j)p(O|C_i,s_j,l_j,d_j) \\ &\quad p(C_i|b_j)p(s_j|l_j,d_j)\,ds_j \end{aligned} \tag{4.3}$$

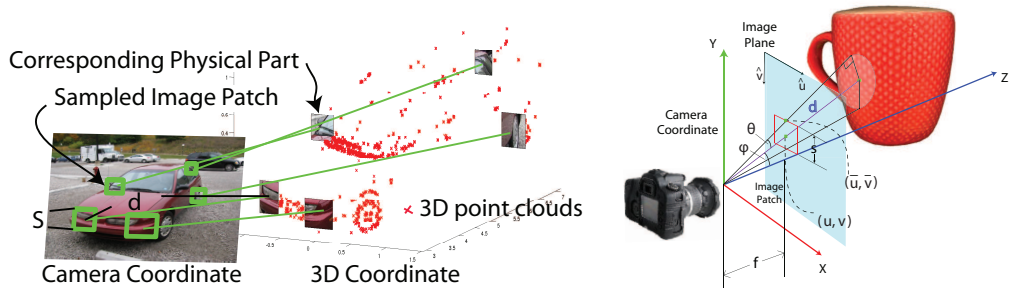


Figure 4.4: Illustration of interplay between scale and depth (depth to scale mapping). Top panel illustrates the interplay between scale and depth. We make the assumption that an image patch (green box) tightly encloses the physical 3D part with a fix size. During training, our method deterministically selects patches given the patch center l, 3D information of the image, and focal length f. During testing, given the selected image patches on the object, our method directly infers the location of the corresponding physical parts and obtains the 3D shape of the object. Bottom Panel illustrates the physical interpretation of Eq. 4.4. Under the assumption that image patch (red bounding box) tightly encloses the 3D sphere with radius r, the patch scale s is directly related to the depth d given camera focal length f and the center $l = (u, v)$ of the image patch. Notice that this is a simplified illustration where the patch center is on the yz plane. This figure is best viewed in color.

Interplay between scale and depth. We design our method so as to specifically selects image patches that tightly enclose a sphere with a fix radius r in 3D during training. As a result, our model enforces a 1-to-1 mapping m between scale s and depth d. This way, given the 3D information, our method deterministically select the scale of the patch at each location l, and given the selected patches, our method can infer the underlying 3D information (Fig.4.4). In detail, given the camera focal length f, the corresponding scale s at location $l = (u, v)$ can be computed as $s = m(d, l)$ and the depth d can be inferred from $d = m^{-1}(s, l)$. The mapping m obeys the following relations:

$$
\begin{aligned}
& s = 2(\overline{v} - v); \;\; \overline{v} = \tan(\theta + \phi) f \\
& \theta = \arcsin(\frac{r}{d_{yz}}); \;\; \phi = \arctan(\frac{v}{f}) \\
& d_{yz} = \frac{d\sqrt{f^2 + v^2}}{\sqrt{u^2 + v^2 + f^2}} : \text{ d projected onto yz plane}
\end{aligned} \tag{4.4}
$$

Hence, $p(s|l,d) = \delta(s - m(d,l))$. Moreover, using the fact that there is a 1-to-1 mapping between s and d, probabilities $p(x|.)$ and $p(O|.)$ are independent to d given s. As a result, only scale s is directly influenced by depth.

In the case when depth is unknown, $p(s|l,d)$ becomes a uniform distribution over all possible scales. Our model needs to search through the scale space to find patches with correct scales. This will be used to detect the object and simultaneously infer the depth $d = m^{-1}(s,l)$. Hence, the underlying 3D shape of the object will be recovered.

Random forest codebook. In order to utilize dense depth map or infer dense reconstruction of an object, we use random forest to efficiently map features b into codeword C (similar to *Gall and Lempitsky* (2009)) so that we can evaluate patches densely distributed over the object. Moreover, random forest is discriminatively trained to select salient parts. Since feature b deterministically maps to C^i given the i_{th} random tree, the voting score $V(O.x|D)$ becomes:

$$V(O,x|D) \propto \sum_{j,i} \int p(x|O, C^i(b_j), s_j, l_j) p(O|C^i(b_j)) \; p(s_j|l_j, d_j)\, ds_j \tag{4.5}$$

where the summation over i aggregates the discriminative strength of different trees. In section 4.2.1.1, we describe how the distributions of $p(x|O, C^i(b_j), s_j, l_j)$ and $p(O|C^i(b_j))$ are learned given training data, so that each patch j knows where to vast votes during recognition.

4.2.1.1 Training the model

We assume that for a number of training object instances, the 3D reconstruction D of the object is available. This corresponds to having available the distance (depth) of each image object patch from its physical location in 3D. Our goal is to learn the distributions of location $p(x|.)$ and object class $p(O|.)$, and the mapping of $C^i(b)$.

Here we define location x of an object as a bounding box with center position q, height h, and aspect ratio a. We sample each image patch centered at location l and select the scale $s = m(l, d)$. Then the feature b is extracted from the patch (l, s). When the image patch comes from a foreground object, we cache: 1) the information of the relative voting direction b as $\frac{q-l}{s}$; 2) the relative object-height/patch-scale ratio w as $\frac{h}{s}$; 3) the object aspect ratio a. Then, we use both the foreground patches (positive examples) and background patches (negative examples) to train a random forest to obtain the mapping $C^i(b)$. $p(O|C)$ is estimated by counting the frequency that patches of O falls in the codebook entry C. $p(x|O, C, s, l)$ can be evaluated given the cached information $\{v, w, a\}$ as follows:

$$p(x|O, C, s, l) \propto \sum_{j \in g(O,C)} \delta(q - b_j \cdot s + l, h - w_j \cdot s, a - a_j)$$

where $g(O, C)$ is a set of patches from O mapped to codebook entry C.

4.2.1.2 Recognition and 3D reconstruction

Recognition when depth is available. It is straightforward to use the model when 3D information is observed during recognition. Since the uncertainty of scale is removed, Eq. 4.5 becomes

$$V(O, x|D) \quad \propto \quad \sum_{j,i} p(x|O, C^i(b_j), m(l_j, d_j), l_j) p(O|C^i(b_j))$$

Since $s_j = m(l_j, d_j)$ is a single value at each location j, the system can detect objects more efficiently by computing less features and counting less votes. Moreover, patches selected using local appearance at a wrong scale can be pruned out to reduce hallucination of objects (Fig. 4.3).

Recognition when depth is not available. When no 3D information is available during recognition, $p(s_j|l_j, d_j)$ becomes a uniform distribution over the entire

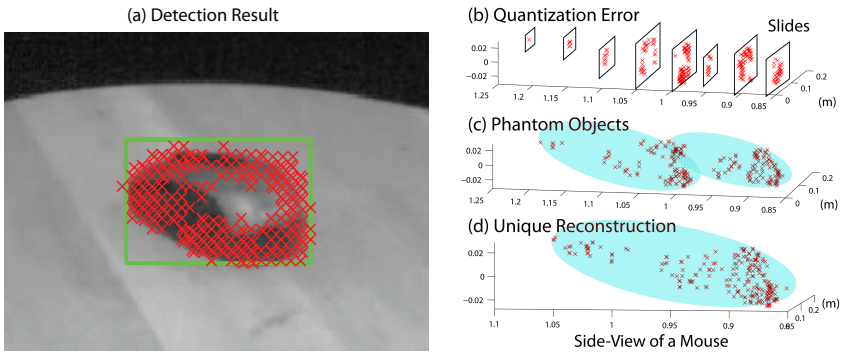


Figure 4.5: A typical detection result in (a) shows object hypothesis bounding box (green box) and patches (red crosses) vote for the hypothesis. A naive reconstruction suffers from quantization error (b) and phantom objects (c). Our algorithm overcomes these issues and obtains (d)

scale space. Since there is no closed form solution of integral over s_j, we propose to discretize the space into a finite number of scales S so that Eq. 4.5 can be approximated by

$$V(O, x|D) \propto \sum_{j,i} \sum_{s_j \in S} p(x|O, C^i(b_j), s_j, l_j) p(O|C^i(b_j)) \ .$$

Decoding 3D information. Once we obtain a detection hypothesis (x, O) (green box in Fig. 4.5(a)) corresponding to a peak in the voting space V, the patches that have cast votes for a given hypothesis can be identified (red cross in Fig. 4.5(a)). Since the depth information is encoded by the scale s and position l of each image patch, we apply Eq. 4.4 in a reverse fashion to infer/decode depths from scales. The reconstruction, however, is affected by a number of issues: i) **Quantization error**: The fact that scale space is discretized into a finite set of scales, implies that the depths d that we obtained are also discretized. As a result, we observe the reconstructed point clouds as slices of the true object (See Fig. 4.5(b)). We propose to use the height of the object hypothesis h and the specific object-height/patch-scale ratio w to recover the continuous scale $\hat{s} = h/w$. Notice that since w is not discretized, $\hat{s}$ is also not discretized. Hence, we recover the reconstruction of an object as a continuum of 3D points (See Fig. 4.5(c)). ii) **Phantom objects**: The strength and

robustness of our voting-based method comes from the ability to aggregate pieces of information from different training instances. As a result, the reconstruction may contain multiple phantom objects since image patches could resemble those coming from different training instances with slightly different intrinsic scales. Notice that the phantom objects phenomenon reflects the uncertainty of the scale of the object in an object categorical model. In order to construct a unique shape of the detected object instance, we calculate the relative object height in 3D with respect to a selected reference instance to normalize the inferred depth. Using this method, we infer a unique 3D structure of the visible surface of the detected object.

4.2.2 Stage 2: 3D Modelling

The goal of 3D modelling is to obtain the full 3D shape and texture of an (unknown) object from a single images portraying the object observed from an (unknown) viewpoint. We can achieve this by using the inferred depths from the image (Sec. 4.2.1), which is a partial (view point limited) 3D point cloud (Partial Shape) of the object (Fig. 4.1(c)). Here we discuss details on how to complete the partial reconstruction.

4.2.3 3D shape recovery

We adopt the idea of using 3D shape exemplars to help recover the missing portions of object 3D surface. The idea (similar to *Pauly et al.* (2005)) is to find a 3D shape exemplar from a given database of 3D shape that can be aligned to the existing incomplete 3D structure. As a result of this alignment, the incomplete elements of the surface can be filled (replaced) with those of the aligned 3D exemplar. The challenges are: i) how to search efficiently in the database of 3D shape exemplars until the most suitable shape is found. ii) perform accurate alignment so as to enable accurate replacement. The first challenge is addressed by leveraging the DEHV detector's

ability to return object class and pose labels. This greatly reduces the search space and allows to extract from the dataset a subset of exemplars that are likely to be very similar to the one we seek to reconstruct.

We carry out accurate alignment between the reconstructed 3D shape and the exemplar 3D shape using a novel ICP algorithm. This novel ICP performs alignment jointly in 3D shape as well as in image space. The alignment in 3D shape is carried out between vertices of a 3D exemplar model and the reconstructed 3D points. The alignment in image space is carried out between the projected occluding boundaries of the 3D exemplar model and object 2D contour. In the image, 2D contours are obtained by applying grabcut foreground segmentation algorithm (*Rother et al.*, 2004) within the detection window. This joint alignment process is obtained by minimizing the following cost function,

$$C(T) = \sum_i C_3(q_i, T(v_i)) + \lambda \sum_j C_2(e_j, Proj(T(v_{o_j}))) \tag{4.6}$$

The first term, $C_3(q_i, T(v_i))$ evaluates the 3D distance between an inferred 3D point q_i and the transformed corresponding vertex $T(v_i)$, where $T(.)$ applies a 3D affine transform on a vertex v_i. The second term,

$$C_2(e_j, Proj(T(v_{o_j}))) \ ,$$

evaluates the 2D distance between a pixel at the object's 2D contour e_j and the 2D projection of the transformed corresponding vertex at the occlusion boundary $(Proj(T(v_{o_j})))$. The parameter λ strikes the balance between two terms and it is chosen empirically. Since the ground truth 3D and 2D correspondences are unknown, the ICP algorithm alternates between 1) finding the transformation T which minimizes the cost $C(T)$ and 2) finding the correspondences which are the closest 3D point $T(v_i)$ to q_i and the closest 2D point $Proj(T(v_{o_j}))$ to e_j, till convergence. By choos-

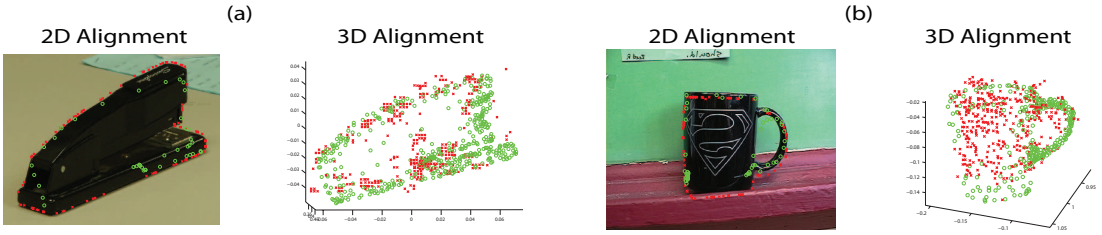


Figure 4.6: Two examples of 3D+2D ICP fitting. In **(a,b)** (Left), the 2D contour alignment results are shown, where a subset of points on the 2D object contour are indicated by red crosses, and projected vertices lying on the occluding boundary of the 3D CAD model are indicated by green dots. In **(a,b)** (Right), the 3D points alignment results are shown, where the partial/sparse inferred point clouds (by DEHV) are indicated by red crosses, and the vertices of the 3D CAD model are indicated by green dots. Notice that these two alignments are jointly enforced by Eq. 4.6.

ing the model corresponding to the smallest cost, we automatically complete the 3D shape which best represents the query object in both 2D and 3D (See Fig. 4.6). Notice that both terms in Eq. 4.6 are critical for achieving robust alignment. For instance, the alignment of projected 3D CAD model with the 2D object contour (second term of Eq. 4.6) can give rise to erroneous solutions that can be easily fixed if the first term of Eq. 4.6 is also considered. On the other hand, second term of Eq. 4.6 is useful to fix small registration errors in 3D which may correspond to large retrojection errors.

4.2.4 Texture Completion

After shape alignment (Fig. 4.1 (d)), we can directly map the texture from the image inside the 2D object contour onto the 3D model. This simple approach gives us a model with incomplete texture (See Fig. 4.1 (e)), where occluded object regions will not be assigned to any texture. In order to obtain a model with complete texture, we propose the following two approaches to infer the texture of the occluded regions of the 3D model.

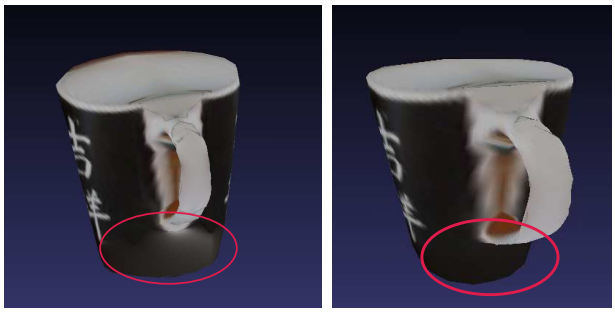


Figure 4.7: Hole filling results using (Left) classic Poisson compositing, and (Right) our error-tolerant compositing technique. Notice that red circles highlight regions where the bleeding artifact is fixed by the error-tolerant technique.

.

4.2.4.1 Symmetric Property

We use the property that object categories have often symmetric topology to transfer the texture from the visible regions to the invisible ones (See Fig. 4.1 (f)). Specifically, we assume that the object shape of the categories of interest are approximately bilateral symmetric (that is, they are symmetric with respect to a plane of reflection). Most common man-made objects satisfy this property. The identification of the bilateral symmetry is carried automatically by applying the symmetry detection algorithm by *Mitra et al.* (2006) to the registered CAD model. This algorithm allows to detect the plane of reflection. After the plane of reflection is detected, we identify the pairs of faces which are in symmetric correspondence across the plane of reflection. By knowing symmetric pairs of faces, we transfer the texture from the visible surface areas (group of faces) to the invisible ones as follows: i) Since faces are either on the left or right side of the plane of reflection, we decide which group (left or right of the plane of reflection) are most visible. The texture coordinates of the vertices composing the faces in the less visible group are removed. ii) The remaining texture coordinates are transferred to their symmetric correspondences.

4.2.4.2 Hole Filling

The property of symmetry discussed above does not guarantee that all surface elements are filled or assigned to object texture. Typically, the resulting models will still have small holes on the surface (See Fig. 4.1 (f)). A rich line of work (*Criminisi et al.*, 2003; *Shamir and Avidan*, 2009; *Hays and Efros*, 2007; *Efros and Leung*, 1999) have studied the problem of image completion or hole filling only on the 2D domain. In this chapter, we apply an error-tolerant image compositing technique (inspired by *Tao et al.* (2010)) to the un-textured region (holes in Fig. 4.1 (f)). Instead of solving the classic poisson equation (*Pérez et al.*, 2003), we solve the following weighted equation:

$$div(W(\nabla I - v)) = 0 \tag{4.7}$$

where I is the unknown image, v is the gradient field to guide the texture completion process, and W is the weight capturing the importance of the gradient field. W is introduced in *Tao et al.* (2010) so that the error between the image ∇I and the gradient field v is not evenly distributed which causes the typical bleeding artifacts (Fig. 4.7). In our implementation, we extract the boundary RGB value from the image and simply assume a uniform gradient field v within region (hole). Most importantly, we set W such that all interior pixels correspond to a constant weight, except for pixels lying on the edges between pairs of faces with very different surface normals corresponds to zero weights. The weights corresponding to boundary pixels are set such that if a boundary color is very different from the median color of its neighboring boundary pixels, its corresponding weight is low, and vice versa. In order to fill all the holes, we first group the faces without texture to a set of disjoint groups, where faces in different groups do not share vertices. For each group, we find the hole boundary which shares vertices with the faces with texture, and extract the RGB value from

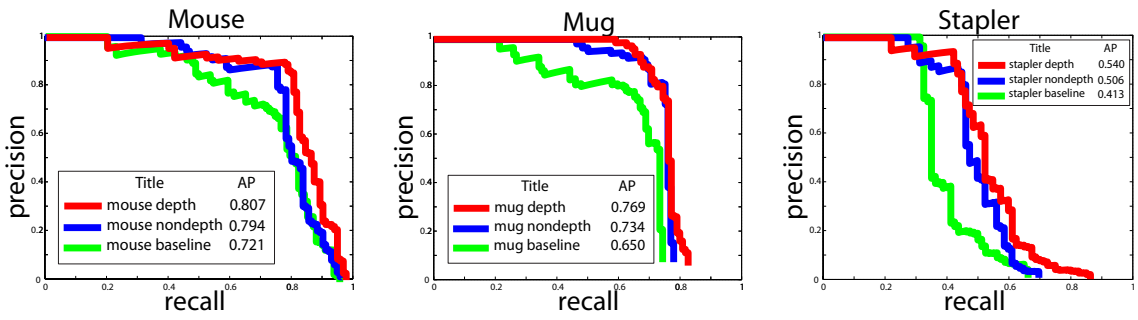


Figure 4.8: Object localization results are shown as precision recall curves evaluated using PASCAL VOC protocol. (Green curve) Result using standard ISM model (baseline). (Blue curve) Result using DEHV with no depth information during testing. (Red curve) Result using DEHV with partial depth information during testing. Notice the consistent improvement of average precision (AP) compared to the baseline hough voting.

the faces with texture along the hole boundary. We then project the group of faces without texture onto an image plane with a most frontal view and solve I in Eq. 4.7 to fill the image RGB value within the projected hole boundary.

4.3 Experiment

We conduct experiments to evaluate the object detection and shape recovery performance of our DEHV algorithm in Sec. 4.3.1, and the quality of 3D modelling in terms of both shape recovery and texture completion in Sec. 4.3.2.

4.3.1 Evaluation of DEHV

We evaluated our DEHV algorithm on several datasets: ETHZ Shape dataset (*Ferrari et al.*, 2008a), 3D object dataset (*Savarese and Fei-Fei*, 2007), and Pascal VOC07 dataset (*Everingham et al.*, 2007). The training settings were as follows. For each training image, we randomly sample 100 image patches from object instances and 500 image patches from background regions. The scale of the patch size from the corresponding object instance is determined by its (known) depth (Fig. 4.4). At the end, 10 random trees (Sec. 4.2.1.1) are trained using the sampled foreground and

background patches for each dataset. For each experiment, we use a Hog-like feature introduced in *Gall and Lempitsky* (2009). During detection, our method treats each discrete viewpoint as a different class O.

4.3.1.1 Exp.I: System analysis on a novel 3D table-top object dataset

Due to the lack of datasets comprising both images and 3D depth maps of set of generic object categories, we propose a new 3D table-top object category dataset collected on a robot platform. The dataset contains three common table-top object categories: mice, mugs, and staplers, each with 10 object instances. We arrange these objects in two different sets for the purpose of object localization and pose estimation evaluation. The object localization dataset (Table-Top-Local) contains 200 images with the number of object ranging from 2 to 6 object instances per image in a clutter office environment. The object pose estimation dataset (Table-Top-Pose) contains 480 images where each object instance is captured under 16 different poses (8 angles and 2 heights). For both settings, each image comes with depth information collected using a structure-light stereo camera. Please see the author's project page (`http://www.eecs.umich.edu/~sunmin`) for more information about the dataset.

We evaluate our method under 3 different training and testing conditions, which are 1) standard ISM model trained and tested without depth, 2) DEHV trained with depth but tested without depth, and 3) DEHV trained and tested with depth. We show that the knowledge of 3D information helps in terms of object localization (Fig. 7.6), and pose estimation (Fig. 4.9). Moreover, we evaluate our method's ability to infer depth from just a single 2D image. Given the ground truth focal length of the camera, we evaluate the absolute depth error for the inferred partial point clouds in Table 4.3-Left Column. Notice that our errors are always lower than the

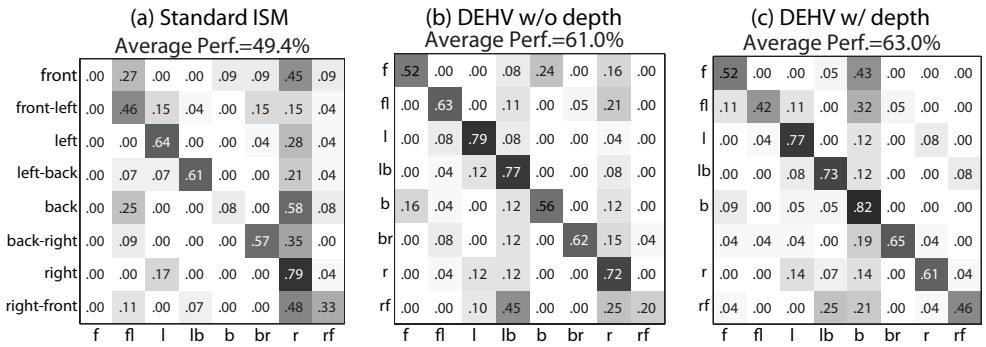


Figure 4.9: Pose estimation results averaged across three categories. The average accuracy increases when more 3D information is available. Notice that, when depth is available in both training and testing, the best performances are achieved.

	Absolute Depth in (m) (known focal length)	Relative Depth (unknown focal length)
	Sparse/Baseline	Sparse/Baseline
Mouse	0.0145/0.0255	0.0173/0.0308
Mug	0.0176/0.0228	0.0201/0.0263
Stapler	0.0094/0.0240	0.0114/0.0298

Table 4.3: Depth recovery error.

baseline errors[1]. We also evaluate the relative depth errors[2] reported in Table 4.3-Right Column when the exact focal length is unknown. Object detection examples and inferred 3D point clouds are shown in Fig. 4.10.

DEHV stapler	DEHV mouse	*Savarese and Fei-Fei* (2008)	*Farhadi et al.* (2009b)
75.0%	73.5%	64.78%	78.16%

Table 4.4: Pose estimation performance on 3D object dataset (*Savarese and Fei-Fei*, 2007)

4.3.1.2 Exp.II:Comparision on three challenging datasets

In order to demonstrate that DEHV generalizes well on other publicly available datasets, we compare our results with state-of-the-art object detectors on a subset of

[1]It is computed assuming each depth is equal to the median of the depths of the inferred partial point clouds.

[2] $\frac{\|d-\hat{d}\|}{d}$ where d is the ground truth depth, and $\hat{d}$ is the estimated depth. $\hat{d}$ is scaled so that d and $\hat{d}$ have the same median.

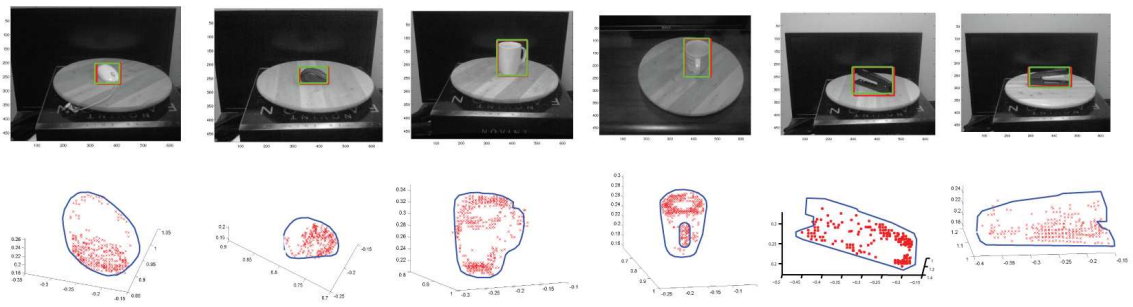


Figure 4.10: Example of object detections (Top) and inferred 3D point clouds (Bottom). The inferred point clouds preserve the detailed structure of the objects, like the handle of mug. Object contours are overlaid on top of the image to improve the readers understanding. Please refer to the author's project page for a better visualization.

(a)

Hough Detector recall @ 1.0 FPPI			
DEHV	ISM	M^2HT	PMKrank
87.1	35.5	55.0	74.2

Non-Hough Detector recall @ 0.3/0.4 FPPI	
DEHV	KAS
77.4/80.6	67.8/77.4

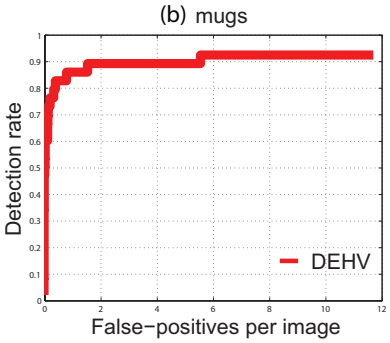


Figure 4.11: Performance on the mug category of ETHZ shape dataset (*Ferrari et al.*, 2008a). (a-Top) Performance comparison with other pure Hough voting methods: ISM (*Leibe et al.*, 2004), M^2HT (*Maji and Malik*, 2009), and PMKrank (*Ommer and Malik*, 2009). (a-Bottom) Performance comparison between state-of-the-art non-hough voting methods (KAS (*Ferrari et al.*, 2008a)). (b) Detection Rate vs. FPPI of DEHV.

object categories from the ETHZ shape dataset, 3D object dataset, and Pascal 2007 dataset. Notice that all of these datasets contain 2D images only. Therefore, training of DEHV is performed using the 2D images from these public available dataset and the depth maps available from the 3D table-top dataset and our own set of 3D reconstruction of cars[3].

ETHZ Shape Dataset. We test our method on the Mug category of the ETHZ Shape dataset. It contains 48 positive images with mugs and 207 negative images with

[3]Notice that our own dataset is only used to provide depth information.

a mixture of apple logos, bottles, giraffes, mugs, and swans. Following the experiment setup in *Ferrari et al.* (2008a), we use 24 positive images and an equal number of negative images for training. We further match the 24 mugs with the mugs in 3D table-top object dataset to transfer the depth maps to the matched object instances so that we obtain augmented depth for positive training images. All the remaining 207 images in the ETHZ Shape dataset are used for testing.

The table in Fig. 4.11(a)-top shows the comparison of our method with the standard ISM and two state-of-the-art pure voting-based methods at 1.0 False-Positive-Per-Image (FPPI). Our DEHV method (recall 83.0 at 1 FPPI) significantly outperforms Max-Margin Hough Voting (M^2HT) (*Maji and Malik*, 2009) (recall 55 at 1 FPPI) and pyramid match kernel ranking (PMK ranking) (*Ommer and Malik*, 2009) (recall 74.2 at 1 FPPI). The table in Fig. 4.11(a)-bottom shows that our method is superior than state-of-the-art non-voting-based method KAS (*Ferrari et al.*, 2008a). Note that these results are not including a second stage verification step which would naturally boost up performance. The recall vs (FPPI) curve of our method is shown in Fig. 4.11(b).

3D object dataset. We test our method on the mouse and stapler categories of the 3D object dataset (*Savarese and Fei-Fei*, 2007, 2008), where each category contains 10 object instances observed under 8 angles, 3 heights, and 2 scales. We adapt the same experimental settings as *Savarese and Fei-Fei* (2007, 2008) with additional depth information from the first 5 instances of the 3D table-top object dataset to train our DEHV models. The pose estimation performance of our method is shown in Table 4.3.1.1. It is superior than *Savarese and Fei-Fei* (2008) and comparable to *Farhadi et al.* (2009b) (which primarily focuses on pose estimation only).

Pascal VOC 2007 Dataset. We tested our method on the car category of the Pascal VOC 2007 challenge dataset (*Everingham et al.*, 2007), and report the localization performance. Unfortunately PASCAL does not contain depth maps. Thus,

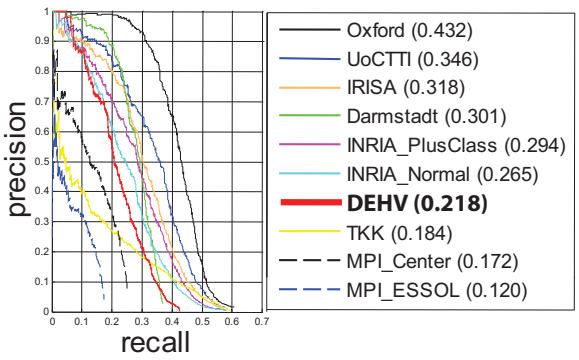


Figure 4.12: Object localization result using PASCAL VOC07 dataset. The precision-recall generated by our method (red) is compared with the results of 2007 challenge (*Everingham et al.*, 2007)-Oxford, (*Everingham et al.*, 2007)-UoCTTI, (*Everingham et al.*, 2007)-IRISA, (*Everingham et al.*, 2007)-Darmstadt , (*Everingham et al.*, 2007)-INRIAPlusClass, (*Everingham et al.*, 2007)-INRIANormal, (*Everingham et al.*, 2007)-TKK, (*Everingham et al.*, 2007)-MPICenter, (*Everingham et al.*, 2007)-MPIESSOL.

in order to train DEHV with 3D information, we collect a 3D car dataset containing 5 car instances observed from 8 viewpoints, and use Bundler (*Snavely et al.*, 2006) to obtain its 3D reconstruction. We match 254 car instances [4] in the training set of Pascal 2007 dataset to the instances in 3D car dataset and associate depth maps to these 254 Pascal training images. This way the 254 positive images can be associated to a rough depth value. Finally, both 254 positive Pascal training images and the remaining 4250 negative images are used to train our DEHV detector. We obtain reasonably good detection performance (Average Precision 0.218) even though we trained with fewer positive images (Fig. 4.12). Detection examples and inferred objects 3D shape are shown in Fig. 4.13.

4.3.2 Evaluation of 3D Modelling

We conduct experiments to evaluate quantitatively and qualitatively the 3D modelling stage of our system (Stage 2 Sec. 4.2.2). At that end, we collect a dataset which comprises 3D reconstructions of 5 object categories: mice, mugs, staplers, cars, and

[4] 254 cars is a subset of the 1261 positive images in the PASCAL training set. The subset is selected if they are easy to match with the 3D car dataset.

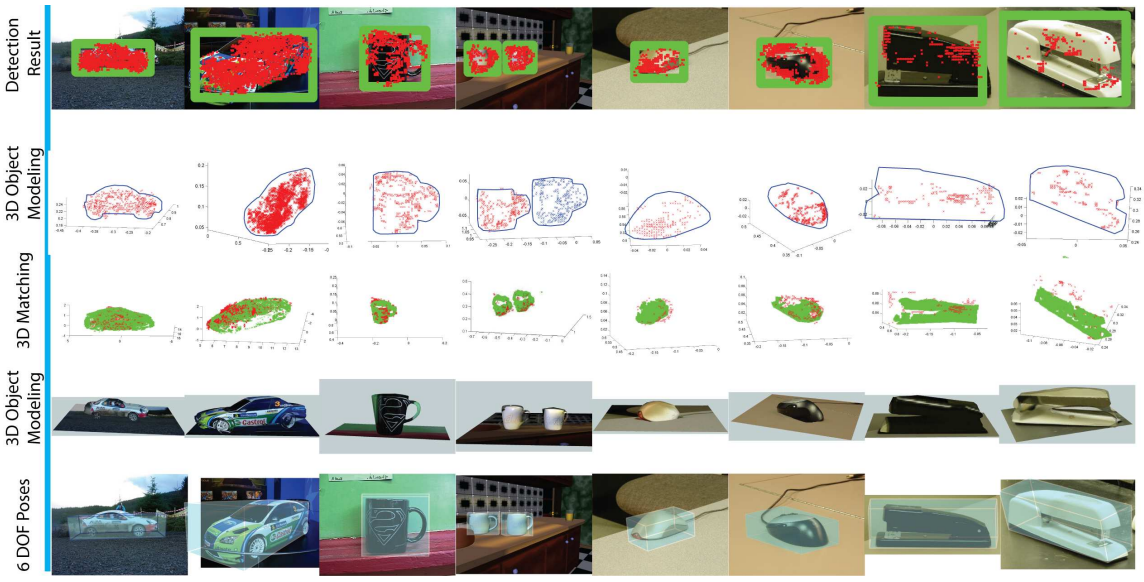


Figure 4.13: Examples of the complete 3D object inference process using the testing images from Pascal VOC07 (*Everingham et al.*, 2007), ETHZ Shape (*Ferrari et al.*, 2008a), and 3D object dataset (*Savarese and Fei-Fei*, 2007). This figure should be viewed in color. **Row 1** Detection results (green box) overlaid with image patch centers (red cross) which cast the votes. **Row 2** Inferred 3D point clouds (red dots), given the detection results. **Row 3** 3D registration results, where red indicates the inferred partial point clouds and green indicates the visible parts of the 3D CAD model. **Row 4** 3D Object modelling using the 3D CAD models and estimated 3D pose of the objects. Notice that the supporting plane in 3D object modelling are manually added. **Row 5** Visualizations of the estimated 6 DOF poses. (See author's project page for 3D visualization.)

bicycles. For each category, the dataset includes about 3 object instances and each instance contains images of the object from camera poses evenly sampled across multiple azimuth angles. The corresponding depth information of each image is either collected from a structured-light stereo camera or a structure from motion method.

We evaluate our method's ability to recover the full 3D shape from an inferred rough 3D structure (output of stage 1). Relative depth errors between ground truth depths and recovered depths (i.e. these obtained after both just 3D ICP (Top-Row) and joint 2D+3D ICP (Bottom-Row) CAD model alignment) are shown in Table 4.5. Baseline errors are computed assuming the depths are all equal to the median of the inferred depths. Notice that the errors of 2D+3D ICP are always smaller than

	Relative Depth Error (Dense/Baseline)				
	Mouse	Mug	Stapler	Car	Bicycle
3D ICP	0.0140/0.0216	0.0287/0.0252	0.0271/0.0283	0.0770/0.1038	0.0630/0.0631
2D+3D ICP	0.0113/0.0209	0.0227/0.0295	0.0260/0.0360	0.0900/0.1189	0.0563/0.0607

Table 4.5: This table shows the median of the relative depth errors for inferred depths obtained after both just 3D ICP (Top-Row) and joint 2D+3D ICP (Bottom-Row) CAD model alignment. Notice that relative depth error is defined as $\frac{\|d-\hat{d}\|}{d}$, where d is the ground truth depth, and $\hat{d}$ is the estimated depth. Notice that $\hat{d}$s for each object instance are scaled so that ds and $\hat{d}$s have the same median so that inconsistent differences between median depths will not influence the evaluation of 3D shape reconstruction.

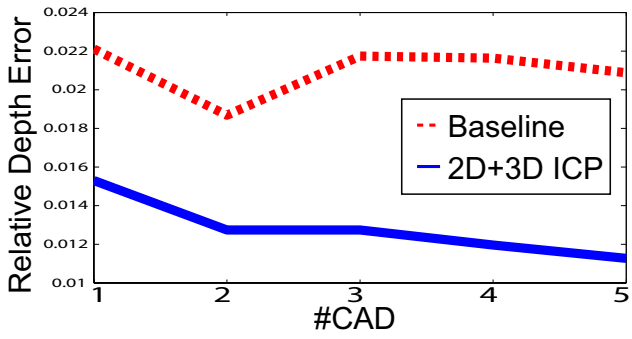


Figure 4.14: Relative depth errors using different number of CAD models for 2D+3D ICP.

the baseline errors, and the errors of 2D+3D ICP are always smaller or similar than the errors of 3D ICP. In our experiments, the inferred 3D and 2D information are matched with about 5 different 3D CAD models selected from the database with the correct object category and pose. The database of 3D CAD models is either collected from *Shilane et al.* (2004) and other online 3D warehouses, or obtained by shape from silhouette (*Laurentini*, 1994). Fig. 4.14 shows a plot of the relative depth errors of 2D+3D ICP versus the number of CAD models of mouse being used. The plot suggests that the more CAD models are used in 2D+3D ICP, the smaller the error in registration is.

We have further used the ETHZ Shape mug dataset (*Ferrari et al.*, 2008a) and 3D object dataset (*Savarese and Fei-Fei*, 2007) to generate typical examples of 3d reconstructions from a single view. Figure 4.15 shows qualitative results of our full

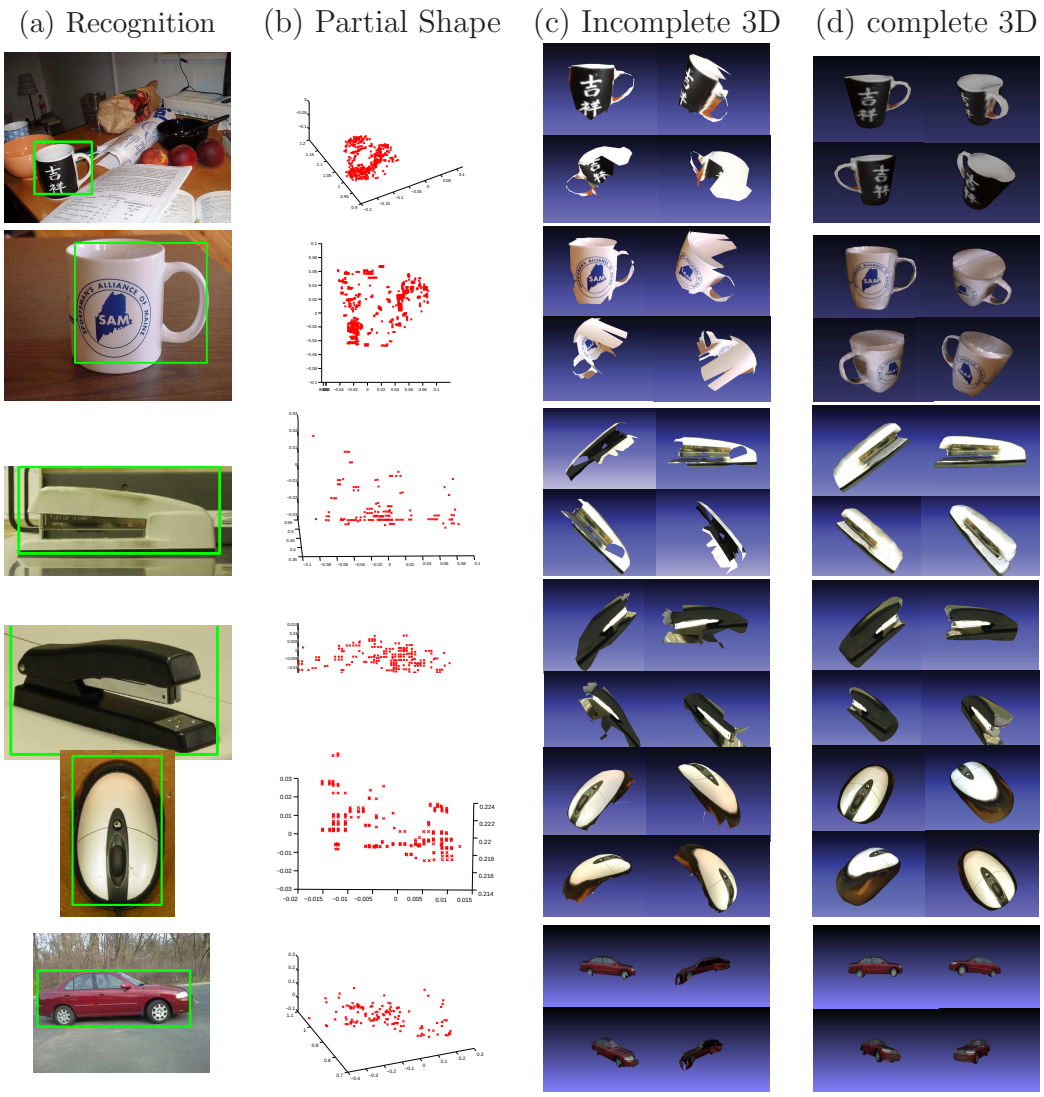


Figure 4.15: Examples of semi-automatic 3D object modelling process on a number of query images. This figure is best viewed in color. **Col. (a)** Sample detection results (green bounding box). **Col. (b)** Partial/Sparse reconstruction of the detected object, where the inferred point clouds in red. **Col. (c)** Incomplete object 3D models using only the visible part of the registered 3D CAD model. **Col. (d)** Complete 3D model after texture completion using symmetric properties and hole filling.

algorithm on several images from 3D object dataset, ETHZ Shape mug dataset, 3D table-top object dataset, and 3D modelling dataset.

4.4 Conclusion

In this chapter, we proposed a new detection scheme called DEHV which can successfully detect objects, estimate their pose from either a single 2D image or a 2D image combined with depth information. Moreover, we demonstrated that DEHV is capable of recover the 3D shape of object categories from just one single uncalibrated image. Given such a partial 3D Shape of the object, we show that novel 3D shape recovery and texture completions techniques can be applied to fully reconstruct the 3D model of the object with both complete shape and texture. As future work, we envision the possibility of integrating more sophisticated texture or 3D shape completion techniques for further improving the quality of the overall 3D model on a large scale of object categories.

CHAPTER V

Models for Articulated Objects

Detecting and estimating the pose (i.e., detecting the location of every body parts) of articulated objects (e.g., people, cats, etc.) has drawn much attention recently. This is primarily the result of an increasing demand for an automated understanding of the actions and intentions of objects in images. For example, person detection and pose estimation algorithms have been applied to the fields of automotive safety, surveillance, video indexing, and even gaming. Most of the existing literature treats object detection and pose estimation as two separate problems. On the one hand, most of the state-of-the-art object detectors (*Felzenszwalb et al.*, 2010; *Leibe et al.*, 2004; *Viola and Jones*, 2004; *Bouchard and Triggs*, 2005) do not focus on localizing articulated parts (e.g., location of heads, arms, etc.). Such methods have shown excellent results on rigid vehicle-type objects (e.g., cars, motorbikes, etc) but less so on the articulated ones (e.g., human or animals) (*Everingham et al.*, 2010). On the other hand, most pose estimators (*Lan and Huttenlocher*, 2005; *Wang and Mori*, 2008; *Felzenszwalb and Huttenlocher*, 2005; *Eichner and Ferrari*, 2009; *Sapp et al.*, 2010b; *Ramanan*, 2006; *Ionescu et al.*, 2009) assume that either the object locations, the object scales, or both are predetermined by either a specific object detector, or given manually. We argue that these two problems are two faces of the same coin and must be solved jointly. The ability to model parts and their relationship allows

to identify objects in arbitrary configurations (e.g., jumping and sitting, see Fig. 1.6) as opposed to canonical ones (e.g., walking and standing). In turn, the ability to identify the object in the scene provide strong contextual cues for localizing object parts.

Some recent works partially attempt to solve the problems in a joint fashion. *Andriluka et al.* (2009) combine a tree-model with discriminative part detectors to achieve good pose estimation and object detection performance. However, good detection performance is only demonstrated on the TUD-UprightPeople and TUD-Pedestrians datasets (*Andriluka et al.*, 2009), which have fairly restricted poses. Alternatively, *Bourdev and Malik* (2009); *Bourdev et al.* (2010) propose a holistic representation of human body using a large number of overlapping parts, called poselets, and achieve the best performance on PASCAL 2007∼2010 person category. However, poselet can only generate a distribution of possible locations for each part's end points independently, which make it difficult to infer the best joint configuration of parts for the entire object.

Our Model. In this chapter, we present a new model for jointly detecting articulated objects and estimating their part configurations (Fig. 1.6(a)). Since the building blocks of this model are object parts and their spatial relationship in the image, we call it the Articulated Part-based Model (APM). Our approach based on APM seeks to satisfy the following properties.

Hierarchical (coarse-to-fine) Representation. Inspired by the articulated body model in the 1980s (*Marr*, 1982) which recursively represents objects as generalized cylinders at different coarse to fine levels (Fig. 1.6(b)), our model jointly models the 2D appearance and relative spatial locations of 2D parts (Fig. 1.6(b)) recursively at different Coarse-to-Fine (CF) levels. We argue that a coarse-to-fine representation is valuable because distinctive features at different levels can be used to jointly improve detection performance. For example, the whole body appearance features are very

useful to prune out false positive detection from the background, whereas detail hand appearance features can be used to further reinforce or lower the confidence of the detection.

Robustness to Pose Variability by Part Sharing. Articulated objects exhibit tremendous appearance changes because of variability in: i) view point location (e.g. frontal view, side view, etc); ii) object part arrangement (e.g. sitting, standing, jumping, etc); iii) self-occlusions among object parts (Fig. 1.6(a)). We refer to the combination of these effects as to the *pose* of the object. Methods such as *Felzenszwalb et al.* (2010); *Zhu et al.* (2010) capture such appearance variations by introducing a number of fully independent models where each model is specialized to detect the object observed under a specific pose. Clearly such representation is extremely redundant as appearance and spatial relationship of parts are likely to be shared across different poses (e.g., a "stretched arm" is observed in both a sitting (Top) and standing (Bottom) person as Fig. 1.6(b) highlights in red). While this representation may be suitable for rigid objects (for which appearance changes are mostly dictated by the view point location of the observer), it may be less so for articulated objects. In order to obtain a more parsimonious representation while keeping the ability to capture rich pose variability, we introduce the concept of "*part-type*". A part-type allows to characterize each part with attributes associated to semantic or geometrical properties of the part. For example a human arm can be characterized by part-types such as "stretched" or "fore-shortened" at a given level of the hierarchy. The introduction of part-types lets parts be shared across object poses if they can be associated to the same part-type. By having the APM to share parts, we seeks to strike a good balance between model richness (i.e., the number of distinct poses) and model complexity (i.e., the number of part-types) (Sec. 5.1.3).

Efficient Exact Inference & learning Following the recursive structure of an APM, we use efficient dynamic programming algorithms to jointly (and exactly) infer

the best object location and estimate their pose (Sec. 5.1.1, 5.1.2). We learn the parameters regulating part appearance and their relationships across coarse-to-fine levels by using a Structured Support Vector Machine (SSVM) (*Tsochantaridis et al.*, 2004) with a loss function penalizing incorrect pose estimation (Sec. 8.3).

Novel Evaluation metric. Because the detection and pose estimation are often performed separately, no standard method exists for evaluating algorithms that address both problems. The popular Percentage of Correctly estimated body Parts (PCP) metric measures the percentage of correctly detected parts for the objects that have been correctly detected. This is problematic in that PCP can be high while detection accuracy is low. To fix this, we propose to directly compare the recall vs False Positive Per Image (FPPI) curves of the whole object and all parts. Using this new measure as well as standard evaluation metrics, we show that APM outperforms state-of-the-art methods. We also show, for the first time, promising pose estimation results on two very challenging categories of PASCAL: cats and dogs.

The rest of the chapter is organized as follows. Model representation, recognition, learning, and implementation details are discussed in Sec.5.1, 8.3, and 7.1.4 respectively. Experimental results are given in Sec. 8.4.

5.1 Articulated Object Representation

Given a still image containing many articulated objects (e.g., persons in Fig. 1.6(a)), our goal is to jointly localize the objects and estimate their poses (i.e., localize articulated parts such as arms, legs, etc).

We introduce a new model called Articulated Part-based Model (APM) to achieve this goal. In designing the APM model, we seek to meet the desiderata discussed in the Sec. 5.1.3 and propose a representation that is hierarchical, robust to pose variability and parsimonious. An APM for an object category (object-APM) is a hierarchical structure constructed by recursively combining primary elements called atomic APM

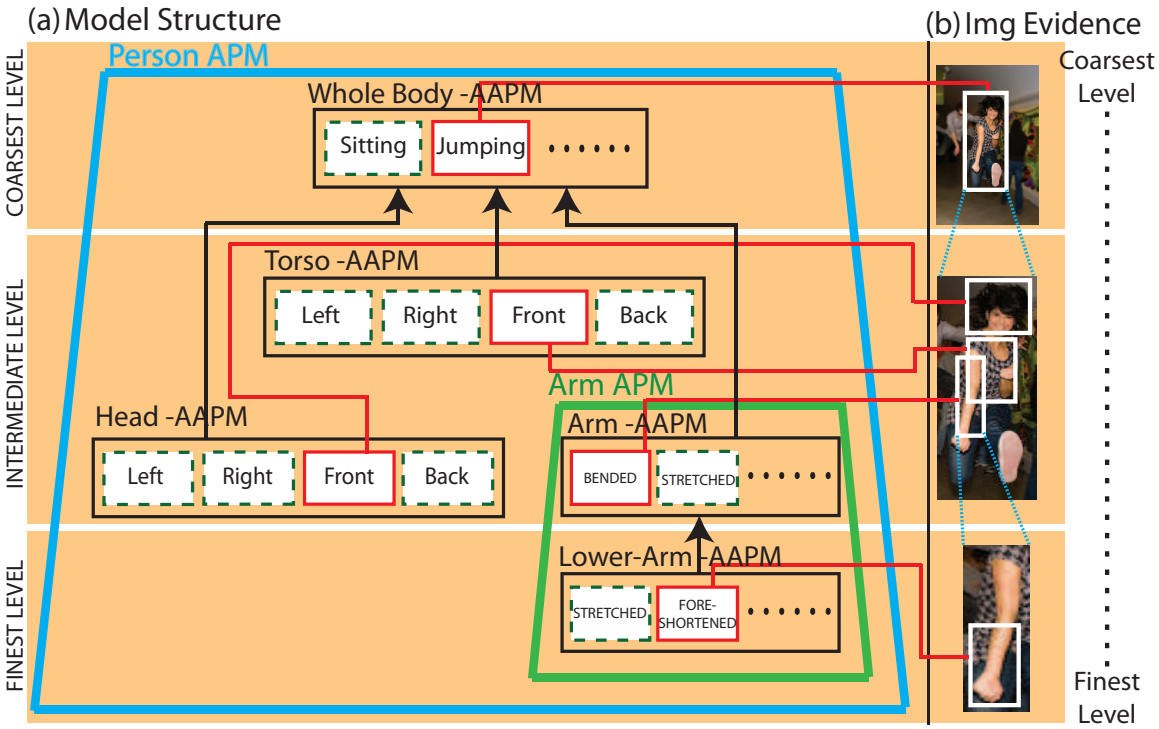


Figure 5.1: Graphical illustration of the recursive coarse-to-fine structure of APM. Panel (a)-Top: An APM (blue trapezoid) can be obtained by recursively combining atomic APMs (AAPM) (black boxes) such as arm-AAPM, lower-arm-AAPM, etc, into higher-level part-APM such as the arm-APM (green trapezoid). Panel (b) shows examples of selected part locations (white windows) at the different levels. The selected part-types are highlighted by red boxes in panel (a).

(AAPM). An AAPM is used to represent an object part at each level of the object representation (e.g., an AAPM can represent a lower arm, the torso, or the whole body, etc.). An AAPM just models the appearance of a part and it is characterized by a number of part types (e.g., an head-AAPM is characterized by types such as left, front, etc.) (Fig. 5.1). AAPMs can be recursively combined into APMs by following a parent-child relationship. E.g., an arm-AAPM and a lower-arm-AAPM are subject to a parent-child relationship (a lower-arm is part of an arm) and are combined into an APM called arm-APM. As an other example, children APMs such as the arm-APM, or head-APM can be combined with their parent (the body-AAPM) and form the person-APM (Fig. 5.1). An APM models the part appearance of both parent and children as well as the 2D geometrical relationships between parent and children.

Since each AAPM can be characterized by several part types, and since AAPM or APMs can be *reused* toward constructing new APMs, an object-APM has the nice property of being able of capturing an exponentially large number of pose configurations by just using a few AAPMs. For instance, suppose that a person is described by 5 parts (head, torso, arm, lower-arm) (thus 5 AAPMs) and that each part is characterized by 4 types. A person-APM model can then encode up to 4^5 different poses in total by only using the 5 AAPMS. This way, the APM allows us to strike a good balance between model richness (i.e., the number of distinct object poses that the model can capture) and model complexity (i.e., the number of model parameters) (See Sec. 5.1.3(i)).

The structure of the APM model (i.e., number of parts, part-types, and parents-child relationships) may be pre-defined following the kinematic construction of the object. Given such a structure, the goal of learning is to jointly learn the appearance model for every part-type and parent-child geometric relationships so that the importance of different part-types and the discriminative power of different parent-child relationships can be automatically discovered. During recognition (inference), the goal is to determine the most likely configuration of object parts and part-types that is compatible with the observation and the learnt APM model. The next section describes how to utilize the recursive structure of APM to efficiently estimate the most likely configuration.

5.1.1 Recognition

Finding the best part configuration (i.e., in our case, both the part locations and types) for arbitrary part-based models corresponding to the highest likelihood or score is in general computationally intractable since the configuration space grows exponentially with the number of parts. By leveraging the recursive structure of an APM, we show that an efficient top-down search strategy for exploring pose configuration

hypotheses in the image is possible. Then we show how to compute a matching score for each hypothesis with a time that is at most quadratic with the number of hypothesis per part by using a bottom-up score assignment scheme. This matching scores are used to guide the top-down search to reach the best pose configuration hypothesis. The result is an efficient inference algorithm that reaches the optimal solution in at most quadratic time.

Top-down Search strategy. The image is explored at different levels of decompositions (from coarse-to-fine) using a recursive strategy. At each level, the image is decomposed into regions (windows) and each region is associated with a part type. Based on the selected part type and the parent-child relationship, each image region is further processed and the next level of decomposition is initiated. The example below clarifies this process.

Let us consider an APM for the object *person* (Fig. 5.1). At the first (coarsest) level of decomposition only a single part is considered. This corresponds to the whole object (*person*). Part types are different human poses (sitting, jumping, standing, etc). The image is explored at different locations (i.e., a score is assigned at different locations following a sliding window approach) and a part type (hypothesis) is associated to each window location. E.g., the white window in Fig. 5.1(b) is associated with the part type jumping. Following the structure of the APM, jumping is a parent of a number of child parts (head, torso, left arm, etc), and the goal is to identify each of these child parts within the current working window. Now the next level of decomposition is initiated. Let us consider the child left-arm part as an APM. The area within the current working window is explored at different locations and each of these are associated to a left-arm part type (hypothesis). At this level, part types are, for instance, stretched or foreshortened. E.g., the white window in Fig. 5.1(b) is associated with the part type stretched. Following the structure of the APM, left-arm is a parent of a number of child parts (upper-arm, lower arm), and the goal is to

identify each of these child parts within the current working window. This initiates the next level of decomposition. The process terminates when all the image windows are explored, all parts are processed and no additional decompositions are allowed. In the Fig. 5.1, the active part types across levels are highlighted by red edges. Notice that the levels of recursion depends on the structure design of the model.

Bottom-up matching score assignment. While the best hypothesis is found using a top-down strategy, the process of assigning a matching score to each hypothesis follows a bottom-up procedure. The benefit of such procedure is that all the scores can be computed in time at most quadratic to the number of hypothesis per part. Notice that special forms of geometric relationship can even be computed in linear time as in *Felzenszwalb and Huttenlocher* (2004b). In details, each matching score is computed by combining an appearance score and a deformation score. The appearance score is obtained by matching the evidence within the working image window against the learned part type appearance model. The deformation score is obtained by: i) computing the parent-child geometrical configuration - that is, the location and orientation (angle) of a part within its parent reference frame; ii) matching this configuration with the learnt parent-child geometrical configuration. These scores are collected and combined bottom-up so as to obtain a final score that indicates the confidence that an image window (at the coarsest level) contains a person with a certain pose and part configuration. Details are explained in Sec. 5.1.2.

5.1.2 Matching Scores

Let us first introduce the parameterization of a part hypothesis in an APM. A part hypothesis is described by the location $h = (x, y, l, \theta)$ and type s of the part, where (x, y) is the part reference position (e.g.,the top-left corner of the part), (l, θ) are the part scale (coarse-to-fine) and 2D orientation, respectively. The task of joint object detection and pose estimation is equivalent to finding a set of part hypotheses

$H = \{(h_0, s_0), \ldots, (h_k, s_k), \ldots\}$ such that the location $h = (x, y, l, \theta)$ and type s is specified for all parts.

As previously introduced, the matching scores can be divided into two classes: *appearance* and *deformation* scores. The appearance score of a specific part-type is obtained by matching the feature $\psi_a(h, I)$ extracted from the image within the window specified by the part location h against the learned appearance model A, and the score is defined as

$$f^A(h; I) = A^T \psi_a(h, I) \tag{5.1}$$

The deformation score is obtained by: i) computing the parent-child geometrical relationship - that is, the difference $\psi_d(h, \hat{h}) = (\Delta x, \Delta y, \Delta\theta)$ of position and orientation between the expected child hypothesis $\hat{h}$ and the actually child hypothesis h at the child reference scale; ii) matching this relationship with the learnt parent-child deformation model d. The score is defined as,

$$\begin{aligned} f^D(h, \hat{h}) = -d^T \psi_d(h, \hat{h}) = -(d_1 \cdot (\Delta x)^2 + d_2 \cdot (\Delta x) \\ + d_3 \cdot (\Delta y)^2 + d_4 \cdot (\Delta y) + d_5 \cdot (\Delta\theta)^2 + d_6 \cdot (\Delta\theta)) \end{aligned} \tag{5.2}$$

where $d = (d_1, d_2, d_3, d_4, d_5, d_6)$ is the model parameter for parent-child deformation.

The final score for each person hypothesis is recursively calculated by collecting and combining scores associated to AAPMs into scores associated to APMs from bottom to upper levels. In details, the score $f_{i,s_i}(h_i, I)$ for an APM with index i and type s_i, is obtained by aggregating: *i*) its own appearance score $f^A_{i,s_i}(h, I)$; *ii*) the scores from each child APM $f_{c,s_c}(h_c, I)$; *iii*) the deformation score $f^D(h_c, \hat{h}_c)$ calculated with respect to its child APM as defined in Eq. 5.2.

This process of estimating the score $f_{i,s_i}(h_i, I)$ by aggregating the scores from its child APMs is achieved by performing the following three steps: i) Child Location

Selection step. Given an expected child part hypothesis $\hat{h}_c$ with index c and part type s_c, we select among all the location hypotheses h_c for this part the one associated to the largest score. The score associated to part c of type s_c is then: $f_{c,s_c}(\hat{h}_c, I) = \max_{h_c}\Big(f_{c,s_c}(h_c, I) + f^D(h_c, \hat{h}_c)\Big)$.

ii) Child Alignment step: we need to align score contributed from each part child. Let us indicate by s_c the type of c^{th} child part. Then, the expected location of the child part c is given by $T(h_i, t_{i,c}^{s_i,s_c})$, such that $T(h,t) = h - t = (x - t_x, y - t_y, l - t_l, \theta - t_\theta)$, where $t_{i,c}^{s_i,s_c}$ is the expected displacement between type s_i of part i and type s_c of part c. iii) Child Type Selection step: For each child part, we need to select the part type corresponding to the highest score as follows:

$$f_c(h_i, I) = \max_{s_c \in \mathcal{S}^c} \left(f_{c,s_c}(T(h_i, t_{i,c}^{s_i,s_c}), I) + b_{i,c}^{s_i,s_c}\right) \tag{5.3}$$

where S^c is the set of types for part c, $b_{i,c}^{s_i,s_c}$ is the bias between type s_i of i^{th} part and type s_c of c^{th} part. Such biases capture the property that some types may be more descriptive than other and therefore they can affect the relevant score function differently. We learn such biases during the learning procedure (Sec. 8.3).

Finally, the score $f_{i,s_i}(h_i, I)$ is obtained as $f_{i,s_i}(h_i, I) = f_{i,s_i}^A(h_i, I) + \sum_{c \in C^i} f_c(h_i; I)$, where C^i is the set of child APMs. Notice that the score $f_{i,s_i}(h_i, I)$ for an atomic APM (AAPM) is simply given by its own appearance score $f_{i,s_i}^A(h_i, I)$. These are computed first as they are the primary elements of the overall object APM structure. Using this way of aggregating the scores, the matching scores for all the parts in the APM structure can be calculated once the scores of its child APMs are computed. Notice that the time required to compute the scores is linearly related to the total number of part-types in the APM.

5.1.3 Model Properties (APM)

In the following, we discuss the important properties of our APM: **i) Sublinearity.** As illustrated in Fig. 5.1, a complex APM is constructed by reusing all APMs at finer levels. If an APM contains M parts and each part contains N types, such APM can represent N^M unique combination of part-types (poses) with the cost of storing $N \times M$ appearance and deformation parameters, respectively (i.e., in Eq. 5.4, A, d are indexed by part i and type s_i). As a result, the number of parameters in APM grows sublinearly with respect to the number of distinct poses; **ii) Efficient Exact Inference.** Despite the complex structure of APM, the "bottom-up" process is efficient, since the scores of different part-types are reused by parent APMs at higher levels. Once the matching scores are assigned, the "top-down" process is efficient as the search for the best part configuration can be done in linear time. Compared to most of the other grammar models which only find the best configuration among a smaller subset of the full configuration space, our method can efficiently explore the full configuration space (e.g., inference on a 640×480 image across ~ 30 scales and 24 orientation in about 2 minutes) making exact inference tractable.

5.2 Model Learning

The overall model parameter $w = (A, \ldots, d, \ldots, b \ldots)$ is the collection of appearance parameters As, deformation parameters ds, and biases bs. In this section, we illustrate how to learn the model parameters w. Since all the model parameters are linearly related to the matching score (Eq. 5.1, 5.2, 5.3), the score of a specific set of part hypotheses H can be computed as $w^T \Psi(H; I)$, where $\Psi(H; I)$ contains all the appearance features $\psi_a(.)$, geometric features $\psi_d(.)$. The matching score can be

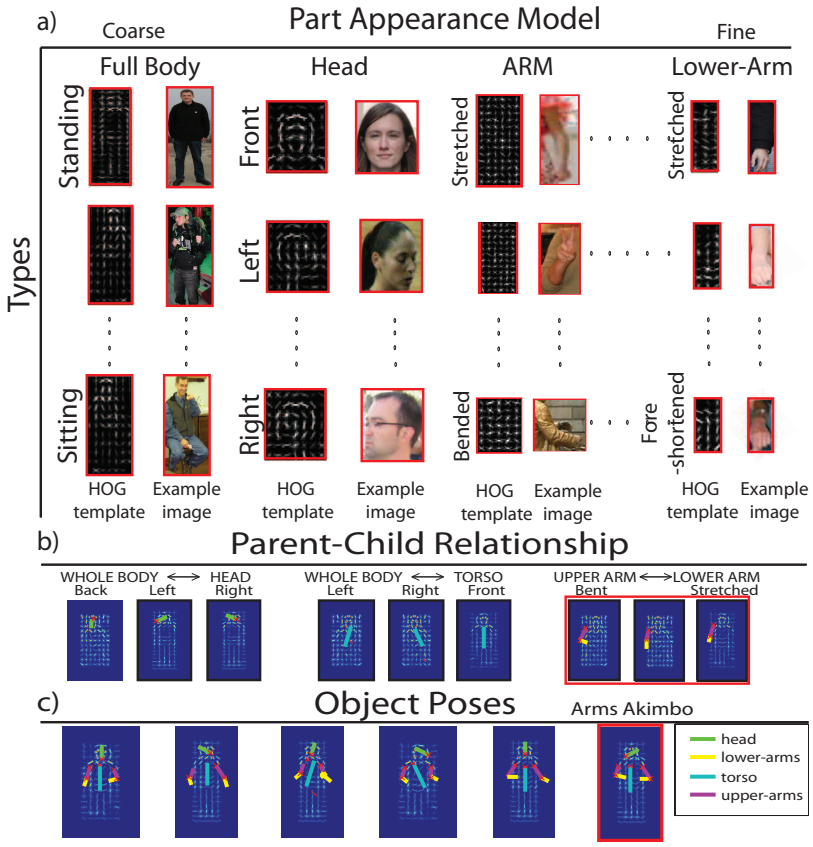


Figure 5.2: Visualization of a learned APM. Panel (a) shows the learned Histogram of Oriented-Gradient (HOG) templates with the corresponding example images for each part-type. Panel (b) shows the parent-child geometric relationships in our model, where different parts are represented as color coded sticks. Panel (c) shows samples of object poses obtained by selecting different combinations of part-types from the APM.

decomposed into

$$w^T \Psi(H; I) = \sum_{i \in \mathcal{V}} A^T_{(i,s_i)} \psi_a(h_i; I) + \sum_{(i,j) \in \varepsilon} \left(b^{(s_i,s_j)}_{i,j} - d^T_{(j,s_j)} \psi_d(h_j, T(h_i, t^{(s_i,s_j)}_{ij})) \right) \quad (5.4)$$

where $\mathcal{V}$ is the set of part indices, ε is the set of parent-child parts, $A_{(i,s_i)}$ specify the appearance parameter for type s_i of part i, $d_{(i,s_i)}$ specify the deformation parameter for type s_i of part i, and $b^{(s_i,s_j)}_{i,j}$ and $t^{(s_i,s_j)}_{ij}$ specify bias and expected displacement of selecting part j with type s_j as the child of part i with type s_i.

Consider that we are given a set of example images and part annotations $\{I^n, H^n\}_n$. We can cast the parameter learning problem into the following SSVM (*Tsochantaridis et al.*, 2004) problem,

$$\begin{aligned} \min_{w,\, \xi^n \geq 0} \quad & w^T w + C \sum_n \xi^n(H) \\ \text{s.t.} \quad & \xi^n(H) = \max_H (\triangle(H; H^n) + \\ & w^T \Psi(H; I^n) - w^T \Psi(H^n; I^n)) \\ & , \forall n \ , \forall H \in \mathcal{H} \end{aligned} \tag{5.5}$$

where $\triangle(H; H^n)$ is a loss function measuring incorrectness of the estimated part configuration H, while the true part configuration is H^n, and C controls the relative weight of the sum of the violation term with respect to the regularization term. The loss is defined to improve the pose estimation accuracy as follows,

$$\begin{aligned} \triangle(H; H^n) &= \frac{1}{M} \sum_{i=1}^{M} \triangle((h_m, s_m); (h_m^n, s_m^n)) \\ &= \frac{1}{M} \sum_{i=1}^{M} (1 - overlap((h_m, s_m); (h_m^n, s_m^n))) \end{aligned} \tag{5.6}$$

where $overlap((h_m, s_m); (h_m^n, s_m^n))$ is the intersection area divided by union area of two windows specified by the part locations and types. Here we use a stochastic subgradient descent method within the SSVM framework to solve Eq. 8.18. The subgradient of $\partial_w \xi^n(H)$ can be calculated as $\Psi(H^*; I^n) - \Psi(H^n; I^n)$, where $H^* = \arg\max_H (\triangle(H; H^n) + w^T \Psi(H; I^n))$. Since the loss function can be decomposed into a sum over local losses for each individual part i, H^* can be solved similarly to the recognition problem in Sec. 5.1.1.

Analysis of our learned model. Fig. 5.2(a) shows learned part appearance models from a person APM with 3 levels of recursion with typical part-type examples. Since all the part-type appearance models are jointly trained by minimizing the same objec-

tive function (Eq. 8.18), the appearance model captures the shapes of the part-type examples as well as the strength of the HOG weights reflecting the importance of each part-type (See Fig. 5.2 for learned HOG templates). Fig. 5.2(b) illustrates a few parent-child geometric relationships in the APM. For example, our model learns that a head appears on the upper-body of a person with different orientations (Fig. 5.2(b)-Left), and learn the stretched and bent configurations for the left-arm (Fig. 5.2(b)-Middle). Notice that these parent-child geometric relationships indeed capture common gestures that appear in daily person activities, like "arms akimbo" (Fig. 5.2(c) red box). Fig. 5.2(c) shows more object poses by selecting different combinations of part-types.

5.3 Implementation Details

Feature representation: We use the projected Histogram of Oriented-Gradient (HOG) feature implemented in *Felzenszwalb et al.* (2010) to describe part-type appearance. **Manual supervision:** In order to train an APM, a set of articulated part annotations is required. For people, we use the 19 keypoints provided in the poselet dataset (*Bourdev and Malik*, 2009) as the part supervision. **Type discovery:** We use the keypoints configuration and part length to object height ratio to initially group parts into different types. After this initial grouping, each example can be discriminatively assigned into different groups according to the appearance similarity. **Discretized part orientation:** We follow the common convention to divide the part orientation space into 24 discrete values (15° each).

5.4 Experiments

We evaluate our method on three main datasets, all of which contain objects in a variety of poses in cluttered scenes. Object detection datasets that contain objects

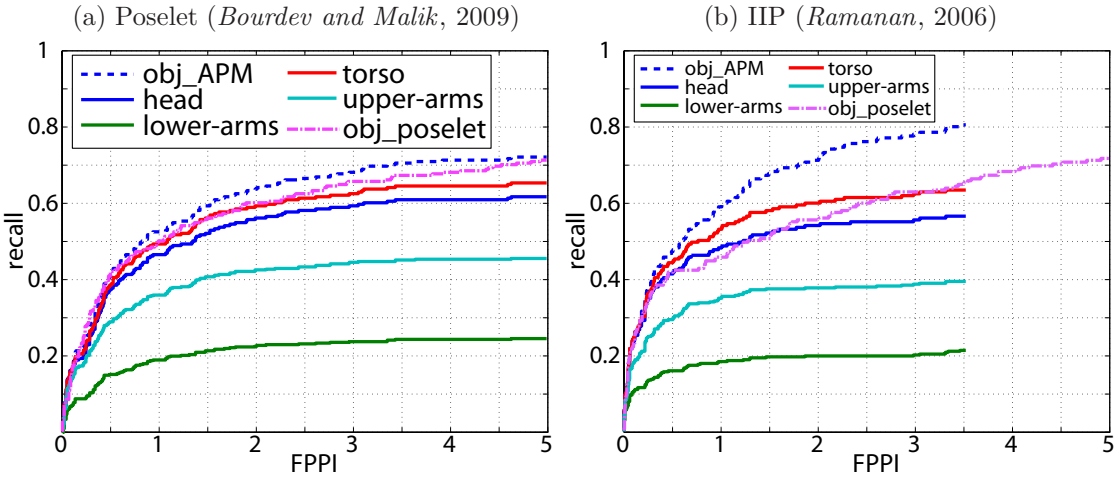


Figure 5.3: Panel (a) shows that our detector applied on Poselet dataset (*Bourdev and Malik*, 2009) slightly outperforms the state-of-the-art person detector (*Bourdev et al.*, 2010) (dashed curves). Panel (b) shows that APM significantly outperforms *Bourdev et al.* (2010) on challenging Iterative Image Parsing dataset (*Ramanan*, 2006). Recall-vs-FPPI curves are shown for each human part (with different color codes) by using our method (solid curves).

with very restricted poses (e.g., TUD-UprightPeople, TUD-Pedestrians (*Andriluka et al.*, 2009)) are not suitable for evaluation here, since we are interested in datasets that make the detection and pose estimation equally challenging. First, we compare our object detection performance on the poselet (*Bourdev and Malik*, 2009) and Iterative Image Parsing (*Ramanan*, 2006) datasets with the state-of-the-art person

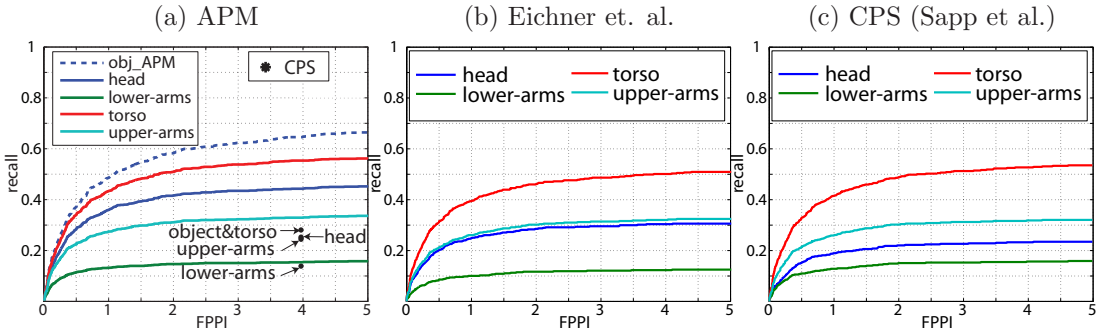


Figure 5.4: Joint object detection and pose estimation performance comparison between our method (a), *Eichner and Ferrari* (2009) (b), and CPS (*Sapp et al.*, 2010b) (c) using recall vs. FPPI for 4 upper-body parts on stickmen dataset. "obj" indicates the detection performance of our object detector.

detector (*Bourdev et al.*, 2010) and demonstrate superior performance, especially on *Ramanan* (2006) which contains challenging sport images with unknown object scale. We introduce a new evaluation metric called recall-vs-False Positive Per Image (FPPI) to show joint object detection and pose estimation performance (Sec. 5.4.1). Second, on the ETHZ stickmen dataset (*Eichner and Ferrari*, 2009), we show APM outperforms state-of-the-art pose estimators (*Sapp et al.*, 2010b; *Eichner and Ferrari*, 2009) using detection results provided by APM.

5.4.1 Evaluation Criteria

Recall v.s. False Positive Per Image (FPPI). We use recall v.s. False Positive Per Image (FPPI) curves to show the joint object detection and pose estimation (i.e.,parts localization) performance. The criteria for a correct object detection is the same as *Everingham et al.* (2009), where the intersection divided by the union between a candidate bounding box and the closest ground truth bounding box needs to be bigger than 50%. A part detection is considered to be correct if both the object corresponding to the part is detected and the Percentage of Correctly estimated body Parts (PCP) matching criteria for the part is satisfied. For person category, we used the same PCP0.5 criteria as *Ferrari et al.* (2008b). For cats and dogs categories, we use PCP0.7 criteria. When plotting the recall v.s. FPPI curves, we first collect all the candidate object instances (including object bounding boxes and part locations), then we sort the candidate instances in a descend order according to the overall matching score (Eq. 5.4). Finally, we calculate the recalls and FPPIs using different thresholds of the matching score to generate the curves.

Percentage of Correct estimated body Parts (PCP). The typical measure of performance on the buffy dataset (*Ferrari et al.*, 2008b) is a matching criteria based on both endpoints of each part (e.g., matching the elbow and the wrist correctly): the state of a body part is correct if the endpoints corresponding to the state (u, v, l, ϕ) are,

on average, within r of the corresponding ground truth segments, where r is a fraction of the ground truth part length. By varying r, a performance curve is produced where the performance is measured in the percentage of correct parts (PCP) matched with respect to r. In our experiment, we set $r = 0.5$ which is commonly used for evaluation.

5.4.2 Comparing with Poselet (Bourdev et al. (2010))

The Poselet dataset (*Bourdev and Malik*, 2009) contains people annotated with 19 types of keypoints, which include joints, eyes, nose, etc. We use the keypoints to define 6 body parts at 3 levels: at the coarsest level, the whole body has 6 types; at the middle level, head has 4 types, torso has 4 types, left&right-arms both has 7 types; at the finest level, left&right-lower-arms both has 2 types. By assuming that body parts and object bounding boxes annotations are available, we train our APM on the same positive images used in *Bourdev and Malik* (2009) and negative images from PASCAL'07 (*Everingham et al.*, 2007). Fig. 5.3(a,b) shows that our object detection performance is slightly better than *Bourdev et al.* (2010) (which achieves the best performance on PASCAL 2010 - human category) on poselet dataset (*Bourdev and Malik*, 2009) but significantly outperforms *Bourdev et al.* (2010) on *Ramanan* (2006), respectively. We observed that *Bourdev et al.* (2010) tends to fail when the aspect ratios of the object bounding boxes vary due to severe articulated parts deformations. Fig. 5.3(a,b) also show our joint object detection and pose estimation performance using part recall vs FPPI curves on these challenging datasets. Typical examples are shown in the $1 \sim 2$ rows of Fig. 5.5.

5.4.3 ETHZ Stickmen dataset

The original ETHZ stickmen dataset (*Eichner and Ferrari*, 2009) contains 549 images, and it is partially annotated with 6 upper-body parts for each person. In order to evaluate the joint object detection and pose estimation performance, we complete

PASCAL stickmen		Recall/$PCP_{0.5}$ @ 4 FPPI				Recall
Det.	Pose methods	torso	head	upper arm	lower arm	obj
APM	Eichner et al.	0.497/77.44	0.311/48.43	0.318/49.52	0.122/19.06	0.642
	CPS	0.525/81.80	0.231/35.92	0.316/49.27	0.155/24.15	
APM (ours)		0.550/85.57	0.439/68.33	0.326/50.73	0.151/23.54	

Table 5.1: Comparison with other methods ((*Eichner and Ferrari*, 2009) and CPS (*Sapp et al.*, 2010b)) for recall/$PCP_{0.5}$ @ 4 FPPI. Red figures indicate the highest recall for each part. We perform better than the state-of-the-art in term of recalls for every part except lower arms.

the annotation for all 1283 people. Previous algorithms evaluated on this dataset are just pose estimators, which rely on an upper body detector to first localize the person. Because of this, the PCP performance is only evaluated on the 360 detected people that were found by the upper body detector. In order to obtain a fair comparison of the joint object detection and pose estimation performance, we use recall/$PCP_{0.5}$ (same as *Eichner and Ferrari* (2009)) vs. FPPI curves for all parts. We believe this is a better performance measure than PCP at a specific FPPI. Indeed PCP ignores to what degree the pose estimation performance is affected by the accuracy of object detectors. Notice that PCP at different FPPI can be easily calculated from the part recall v.s. FPPI curves by dividing the recall of each part by the recall of the object. As an example, the latest PCP from *Sapp et al.* (2010b) is equivalent to the sample points (indicated by dots) at 4 FPPI shown in Fig. 5.4(a). Notice that our method significantly outperforms *Sapp et al.* (2010b) for each body part (except for lower arm where *Sapp et al.* (2010b) and ours are on par).

We apply our APM learned from the Poselet dataset (*Bourdev and Malik*, 2009) to jointly detect objects and estimate their poses on the stickmen dataset (Fig. 5.4(a)). For a more fair comparison, since APM detects 846 people which is much more than the 360 people detected by the upper body detector (*Eichner and Ferrari*, 2009), we show the performance of *Sapp et al.* (2010b); *Eichner and Ferrari* (2009) by using APM's detection results (Fig. 5.4(b)(c)) Even though *Sapp et al.* (2010b); *Eichner*

and Ferrari (2009) incorporate additional segmentation information and color cues, our method shows superior performance for almost all parts. We believe that the main reason is because that *Sapp et al.* (2010b); *Eichner and Ferrari* (2009) assume accurate person bounding boxes are given both in training and testing. Our method overcomes such limitation by performing joint object detection and pose estimation. A recall/$PCP_{0.5}$@4FPPI table comparison is also shown in Table 5.1 with the winning scores highlighted in red. We also found that our detector detects 92.5% of the 360 people detected by the upper-body detector. Among them, without knowing the object location and scale, our PCPs for torso, head, upper-arm, and lower-arm are $91.9\%, 73.0\%, 60.7\%$, and 31.1%, respectively. Typical examples are shown in the $3 \sim 5$ rows of Fig. 5.5.

5.5 Conclusion

In this chapter, we proposed the Articulated Part Model (APM) which is a recursive coarse-to-fine and multiple part-type representation for joint object detection and pose estimation of articulated objects. We demonstrated on four publicly available datasets that our method obtains superior object detection performances. Using a novel performance measure (the part recall vs. FPPI curve) we showed that our part recall at all FPPI are better than the state-of-the-art methods for almost all parts.

Figure 5.5: Typical examples of object detection and pose estimation. Sticks with different colors indicate different parts for different object categories. Blue bounding boxes are our prediction and green ones indicate missed ground truth objects. The first 2 rows show the results on Poselet dataset (*Bourdev and Malik*, 2009) and Iterative Image Parsing dataset (*Ramanan*, 2006). Rows 3 $\sim$ 5 show the comparison between our method, *Eichner and Ferrari* (2009), and CPS (*Sapp et al.*, 2010b) on the stickmen dataset (*Eichner and Ferrari*, 2009).

CHAPTER VI

Efficient Inference on Loopy Models for Articulated Objects

Estimating the pose of humans (e.g., determining body part locations) from images and videos is a core problem in computer vision and it is critical in many applications such as robotics, human computer interaction, video surveillance and gaming. Because speed is an important requirement in most of these applications, researchers have focused on approaches that put a premium on efficiency. Among them, tree-structured models (*Felzenszwalb and Huttenlocher*, 2005; *Ramanan*, 2006; *Eichner and Ferrari*, 2009; *Andriluka et al.*, 2009; *Sapp et al.*, 2010a,b; *Sun and Savarese*, 2011; *Yang and Ramanan*, 2011) (Fig. 1.7(a)) are commonly used. A tree structure typically captures only the most informative spatial relationships (i.e., kinematic constraints) between pairs of parts since the location of one part is well constrained by the location of its connected parts (e.g., the hand location is constrained by the arm location). Inference in tree models can be done efficiently using dynamic programming. As a result, such models can strike a good balance between efficiency and estimation accuracy.

Despite their success, tree models are prone to some common misclassification errors. For example, left and right limbs are often misclassified because their appearance is typically very similar and their estimated locations tend to overlap in the image

(over-counting evidence). To overcome these types of misclassification errors, more structured models such as the loopy graphical models (later referred as loopy model) have been proposed (*Jiang and Martin*, 2008; *Sigal and Black*, 2006; *Zhu et al.*, 2008; *Ren et al.*, 2005; *Tran and Forsyth*, 2010; *Wang et al.*, 2011) (Fig. 1.7(b)). By capturing interactions between a large number of pairs of parts, these methods are effective at improving pose estimation results at the expense of a significantly increased computational cost (*Sontag et al.*, 2008b; *Marinescu and Dechter*, 2007). For instance, methods based on cluster pursuit (*Sontag et al.*, 2008b) become prohibitively slow when the number of states (i.e., number of part location hypothesis) is large since its time complexity is proportional to the number of states to the power of the cluster size (typically ≥ 3). Methods based on Branch-and-Bound (BB) (*Land and Doig*, 1960) search are used in Bayesian networks with a large number of random variables (*Marinescu and Dechter*, 2007), but they become extremely inefficient when the number of states becomes larger (as in the human pose estimation problem). This is because the search proceeds by instantiating each state of every random variable sequentially so that both time and memory usages increase dramatically when the number of states increases. To improve efficiency, i) greedy methods are used to reduce the part location hypothesis (e.g., selecting sparse interest points) (*Ren et al.*, 2005; *Zhu et al.*, 2008; *Tran and Forsyth*, 2010) and/or ii) approximate inference approaches are applied (*Jiang and Martin*, 2008; *Zhu et al.*, 2008; *Ren et al.*, 2005; *Tran and Forsyth*, 2010; *Wang et al.*, 2011).

In this chapter, we propose an efficient and exact inference algorithm based on BB to solve the human pose estimation problem on loopy models, where the number of part location hypotheses is large. Our contribution is two-fold: i) similarly to linear programming relaxation, a novel bound is obtained by relaxing the loopy model into a mixture of star-models; ii) a special data structure (BMT) and an efficient search routine (OBMS) (see Sec. 6.3.2) are used to significantly reduce the time complexity

for calculating the bound in each branch of the BB search. Notice that it is possible to relax the loopy model into other forms (e.g., mixture of trees) as later mentioned in Sec.6.2.1. In this chapter we focus on "mixture of star-models" since we have developed a way to efficiently compute the bound of mixture of star-models. We empirically show that when the number of hypotheses per part is large, our new BB algorithm is an order of magnitude faster than state-of-the-art Cluster Pursuit (CP) method (*Sontag et al.*, 2008b) in solving the exact MAP inference problem. By extending a state-of-the-art tree model (*Sapp et al.*, 2010b) to a loopy model, the estimation accuracy can be significantly improved (up to 5% for lower arm) on Buffy (*Ferrari et al.*, 2008b) and Stickmen (*Eichner and Ferrari*, 2009) datasets. Moreover, our method can exactly solve the MAP inference problem on the Stretchable Models (*Sapp et al.*, 2011) (which contains a few hundreds of variables) in just a few minutes, and achieves superior performance on a number of video sequences best represented by the pre-trained model (see Sec. 6.4.5.2). Finally, since our proposed method can solve the MAP inference problem over any pair-wise MRF, we demonstrated that our method can be helpful in domains beyond computer vision such as molecular biology (Sec. 6.4.6).

The problem of finding the best human body part configuration in a human model is equivalent to finding the Maximum a Posteriori (MAP) assignment over a Markov Random Field (MRF) model, since each body part can be treated as a variable in the MRF and each part hypothesis is equivalent to an assignment of the variable (See details in Sec. 6.2). In the remainder of this chapter, we first review basic concepts on the MAP-MRF inference and its edge-consistent Linear Programming Relaxation (LPR) in Sec. 6.1 and 6.2, respectively. Our novel contribution – the efficient BB algorithm – is introduced in Sec. 6.3. The performance evaluation of our algorithm is presented in Sec. 6.4.

6.1 Introduction on MAP-MRF inference

Markov Random Fields (MRFs) (*Wainwright and Jordan*, 2008; *Koller and Friedman*, 2009) provide a principled framework for modeling problems in computer vision, computational biology, and machine learning where interactions between discrete random variables are involved. However, solving the Maximum a Posteriori (MAP) inference problem in general MRFs is known to be NP-hard (*Shimony*, 1994). Researchers have shown that MAP inference can be approximated by a Linear Programming Relaxation (LPR) problem (*Shlezinger*, 1976; *Koster et al.*, 1998; *Wainwright et al.*, 2005) (see *Werner* (2007) for a review), and such LPR can be solved more efficiently using its dual form (*Kolmogorov*, 2006; *Werner*, 2007; *Globerson and Jaakkola*, 2008; *Komodakis and Paragios*, 2008) (see *Sontag and Jaakkola* (2009) for a review).

When MRFs are characterized only by unary and pairwise potentials, the MAP inference problem can be approximated by an edge-consistent LPR (Sec. 6.2). Unfortunately, it has been shown that edge-consistent LPR cannot exactly solve the MAP inference problem in many real-world problems (*Meltzer et al.*, 2005; *Kolmogorov*, 2006; *Yanover et al.*, 2006; *Sontag et al.*, 2008b; *Komodakis and Paragios*, 2008; *Werner*, 2008). Instead, researchers have proposed the following two deterministic approaches to solve the MAP inference problem exactly:

1. **Cluster Pursuit (CP) (*Sontag et al.*, 2008b):** The upper bound of the edge-consistent LPR can be tightened by adding clusters of variables to form a cluster-based LPR. Cluster Pursuit methods aim to find a set of clusters to tighten the bound incrementally to achieve exact MAP inference.

2. **Branch-and-Bound (BB) (*Land and Doig*, 1960):** Given the upper and lower bounds from the edge-consistent LPR, BB methods systematically search for the exact solution by iteratively applying the branching and bounding strategies (Sec. 6.3.1).

Both approaches are applied to further tighten the upper bound of the edge-consistent LPR, which is obtained by solving for the optimal dual potentials (Eq. 6.4). Hence, the total time to solve the MAP inference problems for both approaches is the sum of the initial time to solve the edge-consistent LPR (T_{Init}) and the time to tighten the upper bound ($T_{Tighten}$) [1] We give more details of the computational time for each of the methods below.

Cluster Pursuit. As shown by *Sontag et al.* (2008b), the time complexity of $T_{Tighten}$ for the CP method is $O(PH^q)$, where q is the maximum cluster size and P is the number of Message Passing (MP) iterations accumulated while pursuing clusters. Theoretically, as clusters across all sizes are explored (i.e., q equals the number of variables), finding an exact solution is guaranteed. However, such exploration is intractable. In practice, CP methods explore clusters with small size (e.g., $q = 3$ or 4) only. By exploring small clusters only, researchers (*Sontag et al.*, 2008b; *Werner*, 2008; *Komodakis and Paragios*, 2008) have shown that exact solutions can be found (albeit without guarantee) in many practical cases. However, CP methods are still prohibitively slow when the number of hypotheses H is large, since the time complexity is proportional to $O(H^q)$.

Naïve Branch-and-Bound Method. A naïve BB approach (Sec. 6.3.1 for an overview of BB) can be designed by using the dual LPR (Eq. 6.4) to obtain bounds at each branch, assuming the dual potentials are *fixed* (once the initial edge-consistent LPR is solved). The time complexity for evaluating the bound is $O(H^2)$. Hence, the time complexity of $T_{Tighten}$ becomes $O(BH^2)$, where B is the number of BB branches. Notice that the naïve BB is also slow for problems with a large H due to the quadratic dependency on H.

To summarize, one can attempt to use naïve BB or CP to solve MAP inference

[1] The computation complexity of all methods reported in this chapter omits the dependency of number of variables N and edges E is the MRF since the same MRF structure is given to all methods for performing MAP inference.

exactly. However, it is challenging to find the exact solution efficiently, especially when the number of hypotheses is large. This condition is common in computer vision problems (e.g., human pose estimation) and computational biology (e.g., protein design). As a result, approximate inference algorithms or simplified representations (e.g., MRFs with only tree structure) are often used to obtain inferior but efficient solutions.In this chapter, we propose an efficient BB algorithm that can exactly solve MAP-MRF inference problems with a large number of hypotheses more efficiently than state-of-the-art methods (*Sontag et al.*, 2008b; *Marinescu and Dechter*, 2007) and other baseline BB methods as shown in Sec. 6.4.

Proposed Efficient Branch-and-Bound Method. We start by proposing in Sec. 6.3 an efficient branch-and-bound method to speed-up the naïve BB method. We utilize a data structure to calculate the bound for each branch in time linear to the number of hypotheses H (Sec. 6.3.2). Moreover, we introduce a novel opportunistic search routine to further speed up the time complexity of the bound calculation to $O(Q \log_2(H))$ for a subset of branches, where Q is a number typically much smaller than H (Sec. 6.3.2). As a result, the time complexity of $T_{Tighten}$ becomes $O(B_1 H + B_2 Q \log_2(H))$ instead of $O(BH^2)$ for the naïve BB, where $B = B_1 + B_2$. Most importantly, we empirically show that, when H is large, the proposed efficient BB is much faster than CP (*Sontag et al.*, 2008b) (Sec. 6.4).

6.2 The MAP problem and its LP Relaxation

Before discussing our proposed BB methods, we first introduce some preliminary concepts, following *Sontag and Jaakkola* (2009). For simplicity, we consider pairwise MRFs (specified by a graph $G = (\mathcal{N}, \mathcal{E})$ with a set of vertices $\mathcal{N}$ and a set of edges $\mathcal{E}$) where each edge is associated with a potential function $\theta_{ij}(h_i, h_j)$. The goal of MAP inference is to find an assignment $\mathbf{h}^{MAP} \in \mathcal{H}_{\mathcal{N}}$ ($\mathcal{H}_{\mathcal{N}}$ denotes the joint hypothesis space $\prod_{i \in \mathcal{N}} \mathcal{H}_i$, where $\mathcal{H}_i$ is the hypothesis space for the i-th variable; i.e., $h_i \in \mathcal{H}_i$)

that maximizes

$$\theta(\mathbf{h}) = \sum_{ij \in \mathcal{E}} \theta_{ij}(h_i, h_j). \tag{6.1}$$

Edge-consistent LPR. Since the problem is in general NP-hard, researchers have proposed to approximate it as a linear programming relaxation problem through pairwise relaxation. For each edge and assignment to the variables on the edge, a marginal $\mu_{ij}(h_i, h_j) \geq 0$ is introduced and $\sum_{h_i,h_j} \mu_{ij}(h_i, h_j) = 1$ is enforced. The edge consistent LPR is given by

$$\max_{\mathbf{h}} \theta(\mathbf{h}) \leq \max_{\mu \in M_L} \{\sum_{ij \in \mathcal{E}} \sum_{h_i,h_j} \theta_{ij}(h_i, h_j)\mu_{ij}(h_i, h_j)\}, \tag{6.2}$$

where M_L is the local marginal polytope enforcing that edge marginals are consistent with each other, i.e.,

$$\sum_{h_i} \mu_{ij}(h_i, h_j) = \sum_{h_k} \mu_{jk}(h_j, h_k), \ \forall h_j. \tag{6.3}$$

The inequality holds since any discrete assignment (including MAP solution) should satisfy the constraints.

6.2.1 Dual LPRs.

Many LPR problems have been proposed and demonstrated to be efficiently solvable in its dual form. *Sontag and Jaakkola* (2009) propose a common framework wherein several dual LPRs can be viewed as minimizing the following functional of dual potentials

$$J(\mathbf{f}) = \sum_{i \in \mathcal{N}} \max_{h_i} f_i(h_i) + \sum_{ij \in \mathcal{E}} \max_{h_i,h_j} f_{ij}(h_i, h_j). \tag{6.4}$$

Here, $f_i(h_i)$ are single node potentials, and $f_{ij}(h_i, h_j)$ are pairwise potentials; these dual potentials satisfy

$$F(\theta) = \left\{ \mathbf{f} : \begin{array}{l} \forall \mathbf{h}, \sum_{i \in \mathcal{N}} f_i(h_i) + \sum_{ij \in \mathcal{E}} f_{ij}(h_i, h_j) \\ \geq \sum_{ij \in \mathcal{E}} \theta_{ij}(h_i, h_j) \end{array} \right\}. \tag{6.5}$$

It has also been shown in *Sontag and Jaakkola* (2009) that without any other constraints on $F(\theta)$, the optimum of this LPR problem would give the MAP value, i.e.

$$\theta(\mathbf{h}^{MAP}) = \min_{\mathbf{f} \in F(\theta)} J(\mathbf{f}). \tag{6.6}$$

Notice that *Sontag and Jaakkola* (2009) emphasize that solving Eq. 6.6 is NP-hard, and many LPR methods can be viewed as adding additional constraints to the F space.

LPR naturally provides upper and lower bounds of the MAP-MRF inference problem that can be used by a branch-and-bound algorithm as follows:

Proposition VI.1. *$J(\mathbf{f})$ is an upper bound of $\theta(\mathbf{h}^{MAP})$ for any $\mathbf{f} \in F(\theta)$ (used as the UB in Algorithm 1).*

Proof. The proposition can be easily proved in two steps. First, $J(\mathbf{f}) \geq \min_{\mathbf{f} \in F(\theta)} J(\mathbf{f})$ is true for any $\mathbf{f} \in F(\theta)$. Then, from Eq. 6.6, we deduce that $J(\mathbf{f}) \geq \min_{\mathbf{f} \in F(\theta)} J(\mathbf{f}) = \theta(\mathbf{h}^{MAP})$ for any $\mathbf{f} \in F(\theta)$. □

Proposition VI.2. *Given any $\mathbf{h}^* \in \mathcal{H}_{\mathcal{N}}$, $\theta(\mathbf{h}^*)$ is a lower bound of $\theta(\mathbf{h}^{MAP})$ for any $\mathbf{f} \in F(\theta)$ (used as the LB($\mathbf{h}^*$) in Algorithm 1).*

Proof. It is directly evident from the definition of $\theta(\mathbf{h}^{MAP})$. □

We will introduce a method to select $\mathbf{h}^*$ in Sec. 6.2.2.

Notice that we have ignored the dependency of the hypothesis space $\mathcal{H}_{\mathcal{N}}$ on $\theta(\mathbf{h})$ and $J(\mathbf{f})$ for conciseness. In the following sections, we will highlight such dependency

of the hypothesis space only when necessary.

As we will discuss in Sec. 6.3.1, a valid upper bound, that is required to guarantee the convergence of the BB algorithm, must satisfy $J(\mathbf{f})$ =LB($\mathbf{h}^*$) when the hypothesis space $\mathcal{H}_{\mathcal{N}}$ is a singleton (i.e., a set with exactly one hypothesis). It can be shown that many dual LPRs (*Shlezinger*, 1976; *Globerson and Jaakkola*, 2008) satisfy such a requirement. In the following section, we proceed with the dual LPR proposed by *Globerson and Jaakkola* (2008), which can be solved by an efficient message passing (MP) algorithm (later referred to as MPLP).

6.2.2 MPLP

The dual LPR proposed by *Globerson and Jaakkola* (2008) can be viewed as $\min_{f \in F_{MPLP}(\theta)} J(\mathbf{f})$, where the constraint set $F_{MPLP}(\theta)$ is given by

$$\left\{ \mathbf{f}: \begin{array}{l} f_i(h_i) = \sum_{j \in \mathcal{N}(i)} \max_{\hat{h}_j} \beta_{ji}(\hat{h}_j, h_i) \\ f_{ij}(h_i, h_j) = 0 \\ \beta_{ji}(h_j, h_i) + \beta_{ij}(h_i, h_j) = \theta_{ij}(h_i, h_j) \end{array} \right\} \tag{6.7}$$

where each edge potential $\theta_{ij}(h_i, h_j)$ is divided into $\beta_{ji}(h_j, h_i)$ and $\beta_{ij}(h_i, h_j)$, and $\mathcal{N}(i)$ is the set of variables connected to variable i according to the graph G. Notice that $F_{MPLP}(\theta) \subset F(\theta)$ since

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}(i)} \max_{\hat{h}_j} \beta_{ji}(\hat{h}_j, h_i) \geq \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}(i)} \beta_{ji}(h_j, h_i) \tag{6.8}$$

$$= \sum_{ij \in \mathcal{E}} (\beta_{ji}(h_j, h_i) + \beta_{ij}(h_i, h_j)) = \sum_{ij \in \mathcal{E}} \theta_{ij}(h_i, h_j); \forall h_i \ . \tag{6.9}$$

For convenience, we follow Eq. 6.4 and 6.7 and transform the dual LPR $J(\mathbf{f})$ to

sum of node-wise potentials f_i as follows:

$$\begin{aligned} J(\beta) = J(\mathbf{f}(\beta)) &= \sum_{i \in \mathcal{N}} \max_{h_i} f_i(h_i) \\ &= \sum_{i \in \mathcal{N}} \max_{h_i} \sum_{j \in \mathcal{N}(i)} \max_{\hat{h}_j} \beta_{ji}(\hat{h}_j, h_i), \end{aligned} \tag{6.10}$$

where the dual potentials $\beta_{ji}(h_j, h_i)$ satisfy

$$\mathcal{B}(\theta) = \{\beta : \ \beta_{ji}(h_j, h_i) + \beta_{ij}(h_i, h_j) = \theta_{ij}(h_i, h_j)\} \ . \tag{6.11}$$

As shown in Proposition VI.1, $J(\beta)$ provides the upper bound. Most importantly, $J(\beta)$ is a valid upper bound as demonstrated in the following proposition.

Proposition VI.3. *$J(\beta) = LB(\mathbf{h}), \forall \beta \in \mathcal{B}(\theta)$ when the hypothesis space $\mathcal{H}_{\mathcal{N}} = \{\mathbf{h}\}$ is a singleton.*

Proof. The proposition is true since

$$\begin{aligned} J(\beta) &= \textstyle\sum_{i \in \mathcal{N}} \max_{\check{h}_i \in \{h_i\}} \sum_{j \in \mathcal{N}(i)} \max_{\hat{h}_j \in \{h_j\}} \beta_{ji}(\hat{h}_j, \check{h}_i) \\ &= \textstyle\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}(i)} \beta_{ji}(h_j, h_i) \\ &= \textstyle\sum_{ij \in \mathcal{E}} (\beta_{ji}(h_j, h_i) + \beta_{ij}(h_i, h_j)) = \theta(\mathbf{h}) = LB(\mathbf{h}). \end{aligned}$$

□

The lower bound can be obtained by defining $\mathbf{h}^* = \arg\max_{\mathbf{h}} \sum_{i \in \mathcal{N}} f_i(h_i)$. Similarly, following Eq. 6.7 and Proposition VI.2, the lower bound $\text{LB}(\mathbf{h}^*)$ can be defined

as sum of node-wise potentials $\breve{f}_i$ as follows:

$$\begin{aligned} \mathrm{LB}(\mathbf{h}^*) = \sum_{i\in\mathcal{N}} \breve{f}_i(\mathbf{h}^*) &= \sum_{i\in\mathcal{N}} \sum_{j\in\mathcal{N}(i)} \beta_{ji}(h_j^*, h_i^*) \\ &= \sum_{ij\in\mathcal{E}} (\beta_{ji}(h_j^*, h_i^*) + \beta_{ij}(h_i^*, h_j^*)) \\ &= \sum_{ij\in\mathcal{E}} \theta_{ij}(h_i^*, h_j^*) = \theta(\mathbf{h}^*) \ , \end{aligned} \tag{6.12}$$

where $\breve{f}_i(\mathbf{h}^*)$ is defined as $\sum_{j\in\mathcal{N}(i)} \beta_{ji}(h_j^*, h_i^*)$.

The upper bound can be tightened by finding the best dual potentials β using the Message Passing (MP) algorithm introduced in *Globerson and Jaakkola* (2008). The time complexity for the MP algorithm is $O(PH^2)$, where P is the number of message passing operations. The time used to obtain β is included in T_{Init}.

6.2.3 Time Complexity and Tightness of the Bound

Time Complexity. From Eq. 6.10, it is clear that it takes $O(H^2)$ operations to calculate the bound, where H is the number of hypotheses per variables. Note that the bound calculation becomes very slow when H is large. In Sec. 6.3, we describe how to utilize a data structure and a search routine to speed-up the bound calculation to $O(B_1 H + B_2 Q \log_2(H))$.

Tightness of the Bound. The tightness of the bounds can be controlled by selecting the dual potentials β (or $\mathbf{f}$ in general). As mentioned later in Sec. 6.3.1, the tightness of the bound will affect the number of branches evaluated in the BB algorithm. Hence, the tighter the bound, the smaller the number of branches evaluated. Since the MP algorithm ensures that the bound $J(\beta)$ is decreasing after each MP operation, the tightness of the bound is related to the number of MP operations (P). However, since the cost of each MP operation is quadratic to H, there is a trade-off between the tightness of the bound and the efficiency of the BB search. We explore the trade-off

Algorithm 2 Efficient Branch and Bound algorithm

Do prep(True) (**Algorithm 3**).
Set $\mathcal{H}_{\mathcal{N}}$ as initial solution space and set a priority queue Q to empty.
Do $(\mathbf{h}^*, \text{UB})$=getBound($\mathcal{H}_{\mathcal{N}}$) (**use Algorithm 6 to efficiently evaluate Eq. 6.10**).
Set GLB = LB($\mathbf{h}^*$) (**Eq. 6.12**).
Insert $(\mathcal{H}_{\mathcal{N}}, \text{UB}, \mathbf{h}^*)$ into Q.
while true **do**
  $(\hat{\mathcal{H}}_{\mathcal{N}}, \text{GUB}, \mathbf{h}^*)$ = pop(Q) (**get the branch with the global upper bound (GUB)**).
  if $\hat{\mathcal{H}}_{\mathcal{N}} \not\subset \mathcal{H}_{\mathcal{N}}$ **then**
    Do prep(False) .
  end if
  Set $\mathcal{H}_{\mathcal{N}} = \hat{\mathcal{H}}_{\mathcal{N}}$.
  if $|\text{GUB} - \text{GLB}| \leq \varepsilon$ **then**
    Return $\mathbf{h}^*$.
  else
    Do $(\mathcal{H}^1_{\mathcal{N}}, \mathcal{H}^2_{\mathcal{N}})$ = split($\mathcal{H}_{\mathcal{N}}, \mathbf{h}^*$) (**branching strategy (Algorithm 7)**).
    Do $(\mathbf{h}^*_1, \text{UB}_1)$=getBound($\mathcal{H}^1_{\mathcal{N}}$).
    Do $(\mathbf{h}^*_2, \text{UB}_2)$=getBound($\mathcal{H}^2_{\mathcal{N}}$).
    GLB = max(LB($\mathbf{h}^*_1$),LB($\mathbf{h}^*_2$),GLB) (**get global lower bound (GLB)**).
    Insert $(\mathcal{H}^1_{\mathcal{N}}, \text{UB}_1, \mathbf{h}^*_1)$ and $(\mathcal{H}^2_{\mathcal{N}}, \text{UB}_2, \mathbf{h}^*_2)$ into Q.
  end if
end while

in Sec. 6.4.

6.3 Efficient Branch-and-Bound

In this section, we first briefly introduce the basics of the BB technique. Then, we propose an efficient BB method. The efficiency is achieved by leveraging a data structure and a search routine to reduce the time complexity (Sec. 6.3.2) and exploring different branching strategies (Sec. 6.3.3).

6.3.1 Branch-and-Bound Basics

Suppose we want to maximize a function g over the hypothesis space $\mathcal{H}$, where $\mathcal{H}$ is usually discrete. A branch-and-bound algorithm has two main steps:

Branching: The space $\mathcal{H}$ is recursively split into two smaller disjoint partition $\mathcal{H}^1$ and $\mathcal{H}^2$ guided by some rules (Sec. 6.3.3) such that $\mathcal{H} = \mathcal{H}^1 \cup \mathcal{H}^2$ and $\mathcal{H}^1 \cap \mathcal{H}^2 = \emptyset$. This yields a tree structure where each node corresponds to a subspace that contains the space of all its descendant nodes.

Bounding: Consider two (disjoint) subspaces $\mathcal{H}^1$ and $\mathcal{H}^2 \subset \mathcal{H}$. Suppose that a lower bound $\text{LB}(\mathcal{H}^1)$ of $\max_{\mathbf{h}\in\mathcal{H}^1} g(\mathbf{h})$ is known, an upper bound $\text{UB}(\mathcal{H}^2)$ of $\max_{\mathbf{h}\in\mathcal{H}^2} g(\mathbf{h})$ is known, and that $\text{LB}(\mathcal{H}^1) > \text{UB}(\mathcal{H}^2)$. Then, there always exists at least one element in the subspace $\mathcal{H}^1$ that is better than all elements of $\mathcal{H}^2$. So, when searching for the global maximizer, one can safely discard such elements of $\mathcal{H}^2$ from the search, and prune the subtree corresponding to $\mathcal{H}^2$. This implies that the search will terminate when a partition $\mathcal{H}^*$ is found such that $|\text{LB}(\mathcal{H}^*) - \text{UB}(\mathcal{H}^*)| = 0$ (zero gap) and $\text{UB}(\mathcal{H}^*)$ is the global upper bound among remaining disjoint hypothesis spaces (i.e., $\text{UB}(\mathcal{H}^*) > \text{UB}(\hat{\mathcal{H}}), \forall\{\hat{\mathcal{H}} | \hat{\mathcal{H}} \cap \mathcal{H}^* = \emptyset\}$). Under this condition, every other disjoint partition $\hat{\mathcal{H}}$ will be pruned out, since $\text{LB}(\mathcal{H}^*) > \text{UB}(\hat{\mathcal{H}})$.

Valid Bound. In order to guarantee the convergence of the algorithm, $\text{UB}(\mathcal{H}^*) = \text{LB}(\mathcal{H}^*)$ must be satisfied when the hypothesis space $\mathcal{H}^*$ is a singleton.

Many BB algorithms (*de Givry et al.*, 2005; *Marinescu and Dechter*, 2007; *Hong and Lozano-Perez*, 2006) have been proposed to solve combinatorial optimization problems. Most of them explore different methods to obtain upper/lower bounds and different branching strategies to split the hypothesis space $\mathcal{H}$ to improve the empirical running time. In the next section, we first present the efficient BB algorithm to speed-up the naïve BB method (Sec. 6.3.2), and then describe our newly proposed branching strategy (Sec. 6.3.3).

6.3.2 Efficient Bound

We observed that finding the maximum value over a branch of a 1D array is the most common operation in the MAP assignment process in Eq. 6.10. In particular,

this operation appears when:

- $\max_{\hat{h}_j \in \mathcal{H}_j} \beta_{ji}(\hat{h}_j, h_i)$ needs to be calculated for all hypotheses $h_i \in \mathcal{H}_i$. Hence, the overall time complexity is $O(H^2)$, where H is the number of hypotheses per variable. Notice that the values of function β are constant.

- $h_i^* = \arg\max_{h_i \in \mathcal{H}_i} f_i(h_i; \mathcal{H}_{\mathcal{N}})$ needs to be calculated for all nodes $i \in \mathcal{N}$. Hence, the overall time complexity is $O(H)$. Notice that $f_i(h_i; \mathcal{H}_{\mathcal{N}})$ is a function of the hypothesis space ($\mathcal{H}_{\mathcal{N}}$) in the branch (i.e., not a constant value).

Since both computations will be repeatedly used for all branches, it is critical that they are implemented efficiently. In the following, we propose a data structure (Sec. 6.3.2.1) and a novel search routine (Sec. 6.3.2.2) to efficiently find the maximum over a branch of a 1D array.

6.3.2.1 Branch-Max-Tree (BMT)

The key idea of the BMT is to utilize a one-time preprocessing step to speed up the querying operation which is supposed to be repeated multiple times. Given an Array $A[1 \dots H]$, a Branch-Max-Tree (BMT) (denoted by BMT.Set(A) in Fig. 6.1) is set up in order to efficiently answer queries of the form $\max_{k \in \mathcal{H}} A[k]$ (denoted by BMT. $\max(\mathcal{H})$ in Fig. 6.1). As illustrated in Fig. 6.1, all nodes keep the pointer to the maximum value of its children nodes in the BMT. The tree is set up such that

Algorithm 3 Preprocessing: prep(InitFlag)

```
1: for i ∈ 𝒩 do
2:    Set BMT_i.set({f_i(h_i) : h_i ∈ ℋ_i}).
3:    if InitFlag then
4:       for h_i ∈ ℋ_i do
5:          for j ∈ 𝒩(i) do
6:             Set BMT_ji(h_i).set({β_ji(h_j, h_i) : h_j ∈ ℋ_j}).
7:          end for
8:       end for
9:    end if
10: end for
```

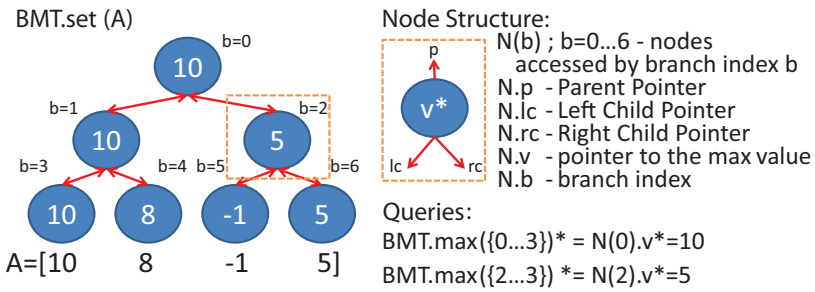


Figure 6.1: Illustration of the Branch-Max-Tree (BMT). The left panel shows an example of a BMT set up from a simple Array A with only 4 elements. Notice that each node in the tree caches a pointer to the max value of its child nodes, and the max value is shown for illustration purposes. The top-right panel shows the data structure of a node used to construct BMT. The bottom-right panel shows that once BMT is built, each branch max query can be converted to a constant lookup time from the corresponding node. Notice that the superscript $*$ denotes pointer dereferencing.

both its time and memory usage are linearly proportional to the size of the array. Once the tree is set up, the maximum value of a branch $\mathcal{H}$ can be simply looked up (in constant time) from a node in the BMT, where all its succeeding leaf-nodes fully cover $\mathcal{H}$. The requirement of using BMT is that the values of the array A must be fixed.

Now we show the computation of $\max_{\hat{h}_j \in \mathcal{H}_j} \beta_{ji}(\hat{h}_j, h_i)$ can be sped up by using the BMT. Since $\beta_{ji}(\hat{h}_j, h_i)$ for a specific h_i is a constant 1D array, $\max_{\hat{h}_j \in \mathcal{H}_j} \beta_{ji}(\hat{h}_j, h_i)$ can be obtained in $O(1)$ time, once the BMT (denoted by $\text{BMT}_{ji}(h_i)$) is set up at the beginning of the BB algorithm in $O(H)$ time. Thus, as shown in line 1 of Algorithm 1, a set of pair-wise BMTs (i.e., $\{\text{BMT}_{ji}(h_i); h_i \in \mathcal{H}_i, (j,i) \in \mathcal{E}\}$) are set up in $O(H^2)$ time to speed up the query time computation. Notice that, the data structure for Range Maximum Query (RMQ) problems (*Berkman and Vishkin*, 1993) can also be used for speed-up. In this chapter, a simpler BMT data structure is used since the ranges (branches) are predefined according to the branching strategy in Algorithm 7. Most importantly, the second computation ($\arg\max_{h_i \in \mathcal{H}_i} f_i(h_i; \mathcal{H}_{\mathcal{N}})$) cannot be sped up by RMQ, but can be handled by BMT as described below.

6.3.2.2 Opportunistic Branch Max Search (OBMS)

In this section, we present a novel search routine called Opportunistic Branch Max Search (OBMS) for speeding up the computation of $\arg\max_{h_i \in \mathcal{H}_i} f_i(h_i; \mathcal{H}_\mathcal{N})$. Similar to our previous discussion, $f_i(h_i; \mathcal{H}_\mathcal{N})$ can be treated as an 1D array. At the first glance, the computation of $\arg\max_{h_i \in \mathcal{H}_i} f_i(h_i; \mathcal{H}_\mathcal{N})$ can be sped up by constructing a BMT associated to the 1D array before the branching and bound search. However, the values of the 1D array is constantly changing since $f_i(h_i; \mathcal{H}_\mathcal{N})$ is a function of the problem space $\mathcal{H}_\mathcal{N}$, and the problem space is constantly changing during the branch and bound search. In this case, the BMT needs to be reset from scratch so that no further speed-up can be achieved for computing $\arg\max_{h_i \in \mathcal{H}_i} f_i(h_i; \mathcal{H}_\mathcal{N})$ using BMT. However, we observed that the value of $f_i(h_i; \mathcal{H}_\mathcal{N})$ for different hypotheses are distributed in a large range (Fig. 6.2(a)). Moreover, if we compare $f_i(h_i; \mathcal{H}_\mathcal{N})$ in one branch with its child branch, we find that the maximal few hypotheses do not change much (Fig. 6.2(b)). This suggests that the maximal few hypotheses of one branch are likely to be the maximal few hypotheses of its child branch as well. Intuitively, we can try to find the maximum hypothesis among these few hypotheses. Most importantly, $f_i(h_i; \mathcal{H}_\mathcal{N})$ and $f_i(h_i; \hat{\mathcal{H}}_\mathcal{N})$ are related to each other as described in the following proposition when $\hat{\mathcal{H}}_\mathcal{N}$ is a sub-space of $\mathcal{H}_\mathcal{N}$.

Proposition VI.4. *$f_i(h_i; \mathcal{H}_\mathcal{N})$ is the element-wise upper bound of $f_i(h_i; \hat{\mathcal{H}}_\mathcal{N})$ when $\hat{\mathcal{H}}_\mathcal{N}$ is a subset of $\mathcal{H}_\mathcal{N}$.*

Proof. The proposition is true since

$$\max_{\hat{h}_j \in \hat{\mathcal{H}}_j} \beta_{ji}(\hat{h}_j, h_i) \leq \max_{\hat{h}_j \in \mathcal{H}_j} \beta_{ji}(\hat{h}_j, h_i);\ \hat{\mathcal{H}}_j \subset \mathcal{H}_j;\ \ j \in \mathcal{N}\ .$$

□

Next, we propose an Opportunistic Branch Max Search (OBMS) routine to speed

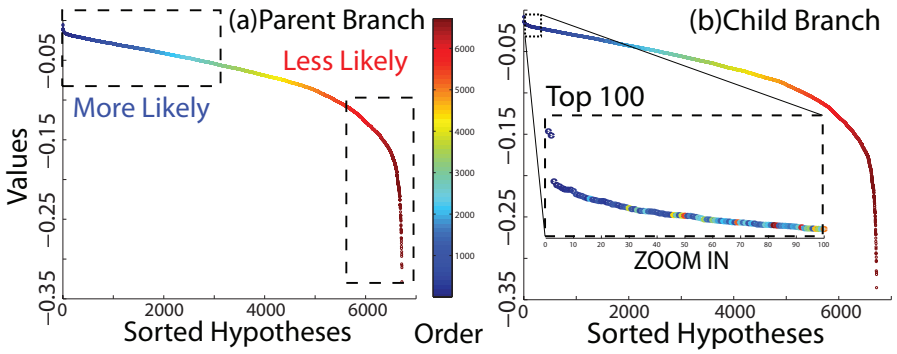


Figure 6.2: Motivation for the Opportunistic Branch Max Search (OBMS). Panel (a) shows the sorted $f_i(h_i)$ value (y axis) for each hypothesis (x-axis) of the parent branch, where colors from blue to red represent the order from large to small values. Panel (b) shows the sorted $f_i(h_i)$ value (y axis) for each hypothesis (x-axis) of the child branch, where the same color-code according to the order obtained in its parent branch is used. We clearly see that the top few hypotheses are mostly all blue. This implies that the top few hypotheses are very similar across parent and child branches.

Algorithm 4 Opportunistic Branch Max Search:(h^*, v) =OBMS($\mathcal{H}_\mathcal{N}$,i)

Input: $\mathcal{H}_\mathcal{N}$ specifies the branch, i specify the node index.
Set h^*=NULL .
while true **do**
  Set $\hat{h} = \mathrm{BMT}_i.\max(\mathcal{H}_i)$ (get maximizer of the upper bound).
  if $h^* \neq \hat{h}$ **then**
    Set $v = 0$.
    for $j \in \mathcal{N}(i)$ **do**
      Set $v = v + \mathrm{BMT}_{ji}(\hat{h}).\max(\mathcal{H}_j)$.
    end for
    Set BMT_i.update$(\mathcal{H}_i, \hat{h}, v)$ (update the upper bound).
    $h^* = \hat{h}$.
  else
    break.
  end if
end while
Return v.

up the computation without approximation by utilizing our intuition and the relation (Proposition VI.4).

Let us define $f_i(h_i; \hat{\mathcal{H}}_\mathcal{N})$ as the 1D Array $A[1 \ldots H]$ and its element-wise upper bound $f_i(h_i; \mathcal{H}_\mathcal{N})$ as $A^U[1 \ldots H]$ (i.e., $A[h] \leq A^U[h]; \ \forall h$). Both $A[1 \ldots H]$ and $A^U[1 \ldots H]$ are given. The opportunistic search strategy (Algorithm 4) tests

if the maximizer of A^U (i.e., $h^{U*} = \arg\max_{h\in\mathcal{H}} A^U[h]$ in line 4 of Algorithm 4) is also the maximizer of A. This can be done by first updating the upper bound $A^U[h^{U*}] = A[h^{U*}]$ (line 10 of Algorithm 4) and checking whether h^{U*} is still the maximizer (i.e., $h^{U*} == \arg\max_{h\in\mathcal{H}} A^U[h]$ in line 5 of Algorithm 4). If the condition $h^{U*} == \arg\max_{h\in\mathcal{H}} A^U[h]$ is satisfied, $A[h^{U*}] \geq A^U[h] \forall h \in \mathcal{H}\backslash h^{U*}$ and the maximizer of A is guaranteed to be found thanks to the following proposition.

Proposition VI.5. *$A[h^{U*}] \geq A^U[h];\ \forall h \in \mathcal{H} \setminus h^{U*}$ implies that h^{U*} is also the maximizer of A.*

Proof. Since $A^U[h] \geq A[h];\ \forall h \in \mathcal{H} \setminus h^{U*}$, $A[h^{U*}] \geq A[h];\ \forall h \in \mathcal{H} \setminus h^{U*}$ is true. □

If the condition is not satisfied, we set $h^{U*} = \arg\max_{h\in\mathcal{H}} A^U[h]$ again and redo the update and check until the condition is satisfied. As a result, the complexity of the opportunistic search routine is linearly proportional to the number of trials $Q < |\mathcal{H}|$ and we avoid evaluating all elements in $\mathcal{H}$ whose cost is $O(|\mathcal{H}|)$.

In the OBMS routine, it is critical to obtain $h^{U*} = \arg\max_{h\in\mathcal{H}} A^U[h]$ and update $A^U[h]$ very efficiently. As shown in Algorithm 4, a BMT data structure (BMT_i) is used to efficiently find the maximizer, and an efficient routine updating the value in

Algorithm 5 Efficient Update Procedure for BMT: BMT.update($\mathcal{H}$,h,v)

1: Input: $\mathcal{H}$ specifies the branch to be updated; h specifies the leaf-node where the update starts; v is the new updated value.
2: Set $b_r = b(\mathcal{H})$ to be the branch index for the whole branch; $b_l = b(\{h\})$ to be the branch index of the leaf-node; the working node $N_w = N(b_l).p$ to be the parent of the leaf-node.
3: Update $N(b_l).v^* = v$.
4: **while** $N_w.b \neq b_r$ **do**
5: **if** $N_w.lc.v^* > N_w.rc.v^*$ **then**
6: Update $N_w.v = N_w.lc.v$.
7: **else**
8: Update $N_w.v = N_w.rc.v$.
9: **end if**
10: Set $N_w = N_w.p$.
11: **end while**

Algorithm 6 Get Bounds: $(\mathbf{h}^*, UB)$=getBound$(\mathcal{H}_{\mathcal{N}})$

1: Set UB = 0.
2: Define $\mathbf{h}^* = \{h_1^*, \ldots, h_i^*, \ldots\}$.
3: **for** $i \in \mathcal{N}$ **do**
4: Get $(h_i^*, v_i) = \text{OBMS}(\mathcal{H}_{\mathcal{N}}, i)$.
5: Set UB = UB + v_i.
6: **end for**
7: Return $(\mathbf{h}^*, \text{UB})$.

the BMT is described below.

Efficient BMT Update. We assume a BMT associated to $A^U[h]$ is already built. A bottom-up procedure (Algorithm 5) efficiently updates the nodes in BMT along the path from the leaf-node corresponding to the updated element to the node corresponding to the branch (denoted by BMT.update$(\mathcal{H}, h^{U*}, A[h^{U*}])$). The time complexity is $O(\log_2 |\mathcal{H}|)$ since the update follows a single path in the BMT. The same BMT can be used for any query with branch $\hat{\mathcal{H}}$ which is the sub-space of $\mathcal{H}$. However, for other queries, the BMT needs to be reset from scratch with complexity $O(|\hat{\mathcal{H}}|)$ (line 9 in Algorithm 1).

The OBMS routine takes $O(Q \log_2 |\hat{\mathcal{H}}|) \leq O(Q \log_2 H)$ instead of $O(|\hat{\mathcal{H}}_i|) \leq O(H)$, where Q is the number of trials in the OBMS. Typically $Q \ll H$ since we observed that the order of the top few hypotheses are not changing much (Fig. 6.2(b)).

It is worthwhile to mention that, a priority queue seems to be a good data structure as well. Similar to BMT, it can be set up in $O(|\mathcal{H}|)$ time, efficiently queried in $O(1)$ time, and updated in $O(\log_2 |\mathcal{H}|)$ time. However, since we need to query for $h^{U*} = \arg\max_{h \in \mathcal{H}} A^U[h]$ multiple times in the BB search for different branches $\mathcal{H}$, a priority queue needs to be set up from scratch for each branch. Therefore, no speed-up can be achieved. On the other hand, we can simply use the sub-tree of a BMT when a sub-branch is visited in the branch and bound search.

In summary, we propose to pre-process the data structure prior to the BB search in $O(H^2)$ time. This allows to reduce the time to calculate the bound from $O(H^2)$ to

Experiments	Method	Init. MP	GVS	VHO	OBMS
Synthetic & protein design problems	Ours+GVS+NoOBMS	Y	Y	N	N
	Ours+NoGVS+NoOBMS	Y	N	N	N
	Ours+GVS+OBMS	Y	Y	N	Y
	Ours+NoGVS+OBMS	Y	N	N	Y
	Ours+NoMP	N	Y	N	N
Human pose estimation problem	NoGVS_NoVHO_NoOBMS	N	N	N	N
	NoGVS_VHO_NoOBMS	N	N	Y	N
	GVS_NoVHO_NoOBMS	N	Y	N	N
	GVS_VHO_NoOBMS	N	Y	Y	N
	NoGVS_NoVHO_OBMS	N	N	N	Y
	NoGVS_VHO_OBMS	N	N	Y	Y
	GVS_NoVHO_OBMS	N	Y	N	Y
	GVS_VHO_OBMS	N	Y	Y	Y

Table 6.1: List of different variants of our efficient Branch-and-Bound algorithm.

$O(Q\log_2(H))$ when a sub-branch is explored and $O(H)$ otherwise. Notice that the overall time complexity of the algorithm also depends on the number of BB iterations B. Hence, the overall average time complexity becomes $O(BH)$ (when only BMT is used) and $O(B_1H + B_2Q\log_2(H)) < O(BH)$ (when both BMT and OBMS are used), where B_1 is the number of times BMT needs to be re-initialized (line 9 of Algorithm 1) and $B_1 + B_2 = B$.

6.3.3 Branching Strategy

The number of branches that a BB algorithm evaluates is also closely related to the branching strategy. Here, we describe different strategies we used for variable selection and variable hypothesis ordering (Algorithm 7).

6.3.3.1 Guided Variable Selection (GVS)

Inspired by *Batra et al.* (2011), we propose a novel scoring function that we call Node-wise Local Primal Dual Gap (NLPDG) as a cue to select which variable to split. Notice that the dual objective is already the sum of the node-wise local dual

objective $f_i(h_i^*)$, where $h_i^* = \arg\max_{h_i \in \mathcal{H}_i} f_i(h_i)$ (Eq. 6.10). Similarly, the node-wise local primal objective is $\breve{f}_i(\mathbf{h}^*) = \sum_{j \in \mathcal{N}(i)} \beta_{ji}(h_j^*, h_i^*)$, where $\mathbf{h}^* = \{h_i^*\}_{i \in \mathcal{N}}$ (Eq. 6.12). The NLPDG is defined as $\delta_i(\mathbf{h}^*) = f_i(x_i^*) - \breve{f}_i(\mathbf{h}^*)$. More precisely, we show the following properties of NLPDG:

Proposition VI.6. *$\delta_i(\mathbf{h}^*)$ is always non-negative.*

Proof. Since $f_i(h_i^*) = \sum_{j \in \mathcal{N}(i)} \max_{\hat{h}_j \in \mathcal{H}_j} \beta_{ji}(\hat{h}_j, h_i^*) \geq \sum_{j \in \mathcal{N}(i)} \beta_{ji}(h_j^*, h_i^*) = \breve{f}_i(\mathbf{h}^*)$, the proposition holds. □

Proposition VI.7. *$\sum_{i \in \mathcal{N}} \delta_i(\mathbf{h}^*) = 0$ implies that the upper bound ($J(\beta)$) equals the lower bound ($\theta(\mathbf{h}^*)$), which is required to terminate the BB search.*

Proof. Since $\sum_{i \in \mathcal{N}} f_i(h_i^*) = \sum_{i \in \mathcal{N}} \max_{h_i \in \mathcal{H}_i} f_i(h_i) = J(\beta)$ and $\sum_{i \in \mathcal{N}} \breve{f}_i(\mathbf{h}^*) = \theta(\mathbf{h}^*)$, the proposition holds. □

These properties suggest that by splitting the variable with the largest NLPDG, we can reduce the primal dual gap quickly. Another heuristic for variable selection is to split the variable with the largest number of hypotheses (later referred to as NoGVS). This baseline method is an efficient way to select a variable, but it may be ineffective if one needs to reduce the number of branches that are evaluated.

Algorithm 7 Branching $(\mathcal{H}_{\mathcal{N}}^1, \mathcal{H}_{\mathcal{N}}^2)$ =split$(\mathcal{H}_{\mathcal{N}}, \mathbf{h}^*)$

1: Input: $\mathcal{H}_v = \mathcal{H}_1 \times \cdots \times \mathcal{H}_n$; $\mathbf{h}^* = (h_1^*, \ldots, h_n^*)$.
2: Select $\mathcal{H}_{s^*}$ where $s^* = \arg\max_{s \in V} \delta_s(\mathbf{h}^*)$ (**select which variable to split using NLPDG (Sec. 6.3.3.1)**).
3: Suppose $\mathcal{H}_{s^*} = [h^i \ldots h^k]$ (**hypotheses are in a fixed order**).
4: Set $\mathcal{H}_{s^*}^1 = [h^i \ldots h^{\lfloor 0.5(i+k) \rfloor}]$;
5: $\mathcal{H}_{s^*}^2 = [h^{\lfloor 0.5(i+k) \rfloor + 1} \ldots h^k]$ (**split in half**).
6: Set $\mathcal{H}_{\mathcal{N}}^1 = \mathcal{H}_1 \times \cdots \times \mathcal{H}_{s^*}^1 \cdots \times \mathcal{H}_n$;
7: $\mathcal{H}_{\mathcal{N}}^2 = \mathcal{H}_1 \times \cdots \times \mathcal{H}_{s^*}^2 \cdots \times \mathcal{H}_n$.
8: Output: $\mathcal{H}_{\mathcal{N}}^1$ and $\mathcal{H}_{\mathcal{N}}^2$.

6.3.3.2 Variable Hypothesis Ordering (VHO)

The order of the hypotheses in the space of each variable is fixed during the BB search as mentioned in line 3 of algorithm 7. On the one hand, the hypothesis space of each variable is split in a deterministic way so that an efficient bound calculation approach can be achieved (Sec. 6.3.2). On the other hand, it is important to select a good order of hypotheses so that the BB search works well. Interestingly, when some knowledge about the problem domain is available, it is possible to order the hypotheses (before BB search) so as to achieve a significant speed-up (later referred to as VHO). For instance, for the human pose estimation problem (Sec. 6.4.5), each hypothesis corresponds to a part location in the 2D image. It is possible to order the hypotheses such that hypotheses corresponding to close-by part locations are also close-by in the ordered list of hypotheses. Notice that CP methods cannot exploit domain knowledge to improve the run time performance. When no domain knowledge is available, we simply order the hypotheses using the local (unary) potentials (before BB search) so that hypotheses with high local potentials are on one side and vice versa (later referred to as NoVHO). Finally, note that all variants of our BB method split the search space in half and use a best-first (largest upper bound) search strategy on a OR search tree.

6.4 Experiments

We first compare different variants of our efficient BB method (see Table 6.1) with the improved naïve BB approach (defined in Sec. 6.4.3), Sontag et al.'s method (*Sontag et al.*, 2008b) (later referred to as MPLP-CP), and a state-of-the-art COP solver (*Marinescu and Dechter*, 2007) (later referred to as COP) on synthetic problems with different number of hypotheses H and variables N (see Sec. 6.4.4). Our analysis clearly shows that our BB methods outperform other methods by solving for

a larger number of hypotheses H for almost all values of N (number of variables) given a 20 minute time budget. Furthermore, we compare our method with MPLP-CP (the best competing method identified in Sec. 6.4.4) on two real-world problems in computer vision (human pose estimation in Sec. 6.4.5) and computational biology (protein design in Sec. 6.4.6), where the number of hypotheses is typically large.

6.4.1 General Experimental Setting

We use the MPLP implementation provided by *Sontag et al.* (2008b) to obtain the dual potentials β before further tightening the bound using CP or BB methods in almost all experiments. This way, the relative performance differences can be directly attributed to the specific method used to further tighten the upper bound (e.g., cluster pursuit or branch-and bound). In the human pose estimation experiment, since the problem can be solved most of the time by solving the edge-consistent LPR, we simply select β as $0.5 \times \theta$ and tighten the bound using our efficient BB algorithm. We evaluate this variant of our BB method (referred to as "Ours+No MP" in Fig. 6.4) on the synthetic data as well. All experiments are performed on a 64-bit 8-Core Intel Xeon 2.40GHz CPU with 48GB RAM; the codes are implemented in single thread C++, and the timing reported is CPU-time (via the C++ clock() function) including the actual run time of the algorithms ($T_{Tighten}$) and the initialization times (T_{Init}) (i.e., the time needed to update dual potentials and building a data structure for BMTs).

6.4.2 Detailed Experimental Settings

By default, the edge-consistent LPR is solved using Message Passing (MP) algorithm to initialize β until convergence[2] or for at most 1000 iterations, whichever comes first. If the gap between the upper and lower bounds is not smaller than 10^{-4}

[2]The convergence condition is when the upper bound improvement is smaller than 10^{-4}.

(stopping criteria) already, we further apply our BB method or MPLP-CP method (*Sontag et al.*, 2008b). Both methods stop when the same stopping criteria (gap $< 10^{-4}$) is reached. For MPLP-CP method (*Sontag et al.*, 2008b), by default, we alternate between adding 20 clusters at a time and running MPLP for 100 more iterations.

In the human pose estimation experiment, since the problems can be solved most of the time without cluster pursuit, we allow the MP algorithm to try harder to solve the edge-consistent LPR. We follow the suggestions from the authors of *Sontag et al.* (2008b) to allow the MP algorithm to continue running until the difference between two consecutive upper bounds is smaller than 10^{-5} (instead of 10^{-4} by default), and to add one triplet at a time (instead of 20 clusters by default). In this way, we ensure that the MPLP-CP method does not slow down by adding unnecessary clusters.

6.4.3 Improved Naive Branch-and-Bound Algorithm

Recall that the dual objective is a functional of dual potentials (in Eq. 6.10), and the dual potentials are fixed in the naive BB method. The naive BB method can be improved by further updating the dual potentials at each branch to tighten the upper bound. In this case, the time spent at each branch becomes the sum of the time to evaluate the upper bound ($O(H^2)$) and the time T_f to update the dual potentials. It is easy to show that when the dual potentials are updated using message passing (*Globerson and Jaakkola*, 2008), then $T_f \leq O(H^2)$ and that the time complexity of $T_{Tighten}$ becomes $O(\hat{B}H^2)$, where $\hat{B}$ is the number of BB branches. Notice that the number of branches B and $\hat{B}$ for the naive and the improved BB, respectively, are different quantities. The number of branches B is larger than $\hat{B}$ since the bound at each branch for the naive BB is not tightened by updating better dual potentials. Hence, the improved naive BB is always faster than the naive BB (i.e., $O(\hat{B}H^2) < O(BH^2)$). To make the improved naive BB a competitive baseline,

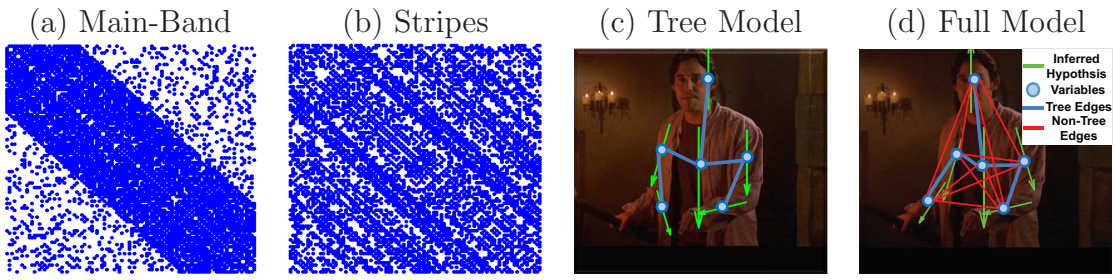


Figure 6.3: Panel (a,b) show the adjacency matrices representing the pairwise interactions in two types of MRFs: a) the main-band (K_b), and b) stripes (K_s) sparsity patterns respectively, where the blue dots denote that interactions between two variables are modeled. Panel (c,d) show the graphical representation of MRFs for six body parts. Here, circles denote variables, and blue and red edges denote the interactions between pairs of variables in the tree and full model respectively. The MAP assignments are shown in green arrows, where each arrow indicates the location and orientation of the body part. Notice the full model produces the correct pose (d), whereas the tree model produces the wrong right-lower-arm (c).

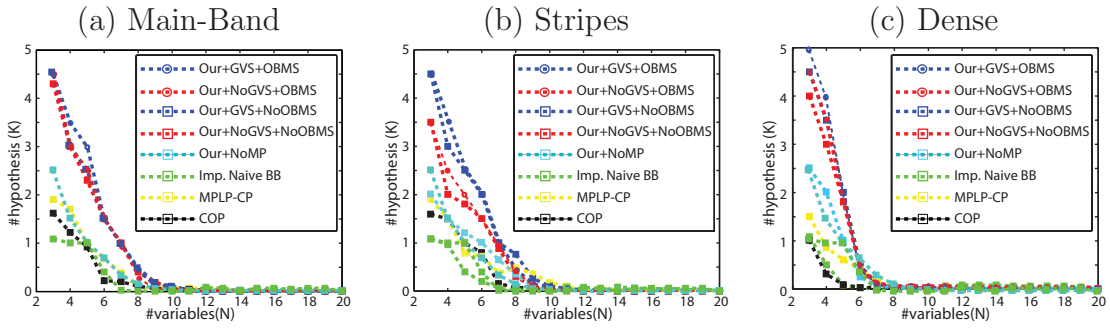


Figure 6.4: Comparison between variants of our efficient methods (blue, red, and cyan curves) and the improved naïve BB (green), MPLP-CP (*Sontag et al.*, 2008b) (yellow), and COP (*Marinescu and Dechter*, 2007) (black) on the main-band (K_b), stripes (K_s), and dense (K_d) problems. For each problem, the maximum numbers of hypotheses (y-axis) solved for problems with different values of N (numbers of variable) (x-axis) in the unit of 1K are plotted. Notice that our variants with the opportunistic branch max search (OBMS) (denoted by circle dots) always solve more hypotheses compared to the variants without OBMS (denoted by square dots).

we initialize the memory for storing the functionals once at the beginning of the algorithm and update only a subset of functionals in each branch. In this way, our implementation is not allocating memory for functionals at each branch and the functionals are not re-initialized from scratch at each branch. The same stopping criteria mentioned above is used in all the branches.

6.4.4 Experiments with Synthetic Data

We synthesize pairwise MRFs with three types of sparsity structures: (i) a sparsity structure with a single main band K_b (Fig. 6.3(a)), (ii) a sparsity structure with many stripes K_s (Fig. 6.3(b)), and (iii) a dense, fully connected model K_d. The first structure simulates problems where local interactions dominate long range interactions (e.g., chain-model, protein design, etc.). The second structure simulates the problems in which both local and long range interactions are important but the interactions are clustered together.

For all experiments, we synthesize problems with different number of variables N and number of hypotheses H. Unary and pairwise potentials are sampled from standard normal distribution (i.e., $\theta_i(x_i) \sim \mathcal{N}(0,1)$ and $\theta_{ij}(x_i, x_j) \sim \mathcal{N}(0,1)$). Given a fixed time budget T (20 min), we explore the maximum value of H (number of hypotheses per variable) that the algorithms can solve for different values of N (number of variables). The first two types of problems are 50% sparse (i.e., 50% of the entries in the adjacency matrix of the pairwise MRFs are zeros.). Fig. 6.4 shows that, for almost all values of N (x-axis), most variants of our method (except "Ours+No MP") can solve problems with more hypotheses (y-axis) than the improved naïve BB, MPLP-CP (*Sontag et al.*, 2008b), and the COP solver (*Marinescu and Dechter*, 2007) can. In some cases, our best BB method can solve problems with a few thousands more hypotheses than the best competing algorithm (*Sontag et al.*, 2008b). We found that our BB method ("Ours+GVS+OBMS") using NLPDG and OBMS achieves the best performance. Notice that, given the time budget, none of the methods can solve problems with a large number of hypotheses when the number of variables N becomes very large. This is because the problem size (hypothesis space $|\mathcal{X}_V|$) increases exponentially with the number of variables (i.e., H^N).

6.4.5 Human Pose Estimation (HPE)

The HPE problem consists of estimating the location and orientation of human body parts, such as head, torso, upper-arm, lower-arm, etc., from a single image (green arrows in Fig. 6.3(c,d)). Solving this problem is critical in many computer vision tasks such as human activity understanding (*Yang et al.*, 2010; *Yao and Fei-Fei*, 2010) and tracking (*Andriluka et al.*, 2010).

The HPE problem can be modeled as a MRF where each body part is a variable, each unique location and orientation of a body part is a unique hypothesis, and each edge between a pair of variables captures the interactions between a pair of body parts (Fig. 6.3(c,d)). Most state-of-the-art approaches (*Ramanan*, 2006; *Sapp et al.*, 2010b; *Eichner and Ferrari*, 2009; *Ferrari et al.*, 2008b; *Yang and Ramanan*, 2011; *Sun and Savarese*, 2011) follow the kinetic structure of the human body (e.g., lower-arms are connected to upper-arms, and upper-arms are connected to torso, etc.) to construct MRFs with a tree structure such that efficient and exact MAP inference can be achieved by applying dynamic programming techniques. More sophisticated approaches construct MRFs with non-tree-structures (*Lan and Huttenlocher*, 2005; *Zhu et al.*, 2008; *Wang et al.*, 2011). However, due to the large number of hypotheses per part ($\sim 1K$), these approaches rely on approximate MAP inference algorithms to obtain inferior but efficient solutions.

In the following sections, we first model the HPE problem using a fully connected pairwise MRF (later referred to as the full model) capturing the complete set of pairwise interactions between pairs of six human body parts. The performance of our model is evaluated on both Buffy (*Ferrari et al.*, 2008b) and Pascal Stickmen dataset. Then, we further evaluate the ability of our approach to handle HPE problems given a video sequence on VideoPose2.0 dataset (*Sapp et al.*, 2010c). For both set of problems, we compare the computation efficiency of our BB algorithm with MPLP-CP (the best competing method identified in Sec. 6.4.4), and the accuracy with state-of-the-art

	Buffy				Stickmen	
Parts Parts	Ours (full)	Ours (13 pairwise interactions)	Ours (7 pairwise interactions)	CPS	Ours (full)	CPS
Head	**99.15**	**99.15**	**99.15**	**99.15**	**99.44**	99.17
Torso	**99.57**	**99.57**	**99.57**	**99.57**	**99.72**	**99.72**
RUA	**95.30**	93.59	93.16	**95.30**	**82.50**	82.22
LUA	**92.31**	**92.31**	**92.31**	91.88	80.28	**81.67**
RLA	**63.25**	59.83	60.26	59.83	**56.94**	54.44
LLA	**64.53**	62.39	61.97	59.83	**53.89**	51.94

Table 6.2: Pose estimation accuracy of different variants of our models compared to CPS on Buffy and PASCAL Stickmen datasets. Ours F, Ours 13, and Ours 7 denote our fully connected model, the model with 13 pair-wise relationships (full model excluding 2 relationships of symmetric arm pairs), and the model with 7 pair-wise relationships (tree model with 2 additional symmetric arm relationships), respectively.

approaches (e.g., the Cascaded Pictorial Structure model (CPS) (*Sapp et al.*, 2010b) and Stretchable Model (SM) (*Sapp et al.*, 2011)).

6.4.5.1 HPE Problem Given a Single Image

In order to guarantee a fair comparison with CPS (*Sapp et al.*, 2010b) we extend CPS (*Sapp et al.*, 2010b) into a fully connected model by capturing pair-wise part relationships other than the kinematic constraints. Notice that the same features, types of classifiers, and learning procedures are used to build and train the fully connected model. CPS is an upper body tree model with 6 articulated parts (i.e., head, torso, left/right-upper-arms, and left/right-lower-arms) which are parametrized by the location (x, y) and orientation (μ) of the part (i.e., $h = (x, y, \mu)$). The model achieves impressive performances by capturing more sophisticated pair-wise relationships than just geometric relationships using segmentation, contour, shape, and color features. In the CPS model, 5 decision-tree-based classifiers are trained to predict the strength of pair-wise relationships given the features. On top of the existing 5 classifiers, we further train 10 additional decision-tree-based classifiers and extend the model into a fully connected pair-wise model (i.e., a loopy model). Since now all the classifiers are trained independently, we treat the responses of the classifiers as the features $\mathbf{\Psi}$

and assume all potentials are linearly related to a set of model parameters such that the overall model is linearly related to the parameters as:

$$f(\mathbf{h}; \mathbf{w}, I) = \sum_{i \in \mathcal{N}} w_i^T \psi_i(h_i, I) + \sum_{ij \in \mathcal{E}} w_{ij}^T \psi_{ij}(h_i, h_j, I) \ , \tag{6.13}$$

where $\mathbf{w} = \{w_i, \dots, w_{ij}, \dots\}$ is the set of all model parameters, $\psi_i(h_i, I)$ and $\psi_{ij}(h_i, h_j, I)$ are the unary and pair-wise features, respectively, and I is the image information. For conciseness, we define $f(\mathbf{h}; \mathbf{w}, I) = \mathbf{w}^T \mathbf{\Psi}(\mathbf{h}, I)$, where $\mathbf{w}$, $\mathbf{\Psi}(.)$, and $\mathbf{h}$ are in the concatenated vector forms. The model is learned using the max-margin formulation formulated below,

$$\begin{aligned} \min_{\mathbf{w}, \xi \geq 0} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_n \xi_n \\ s.t. \quad & \forall n, \forall \mathbf{h} \neq \mathbf{h}_n, \ \ -\mathbf{w}^T \mathbf{\Psi}(\mathbf{h}, I_n) \geq 1 - \xi_n \\ & \forall n, \ \ \mathbf{w}^T \mathbf{\Psi}(\mathbf{h}_n, I_n) \geq 1 - \xi_n \ , \end{aligned} \tag{6.14}$$

where h_n and I_n are the ground truth part configuration and the image evidence of the n_{th} image, respectively. We use a cutting plane solver (*Tsochantaridis et al.*, 2004) to solve the above quadratic programming (QP) problem with a large number of negative constraints (the constraints in the first row). We use the max-margin formulation to learn weights $\mathbf{w}$ such that the ground truth configuration (pose assignment) h_n has the highest score. This is equivalent to having the weights adjusted in such a way that the MAP estimation becomes as consistent with the ground truth as possible.

We conduct experiments on both Buffy (*Ferrari et al.*, 2008b) and PASCAL Stickmen (*Eichner and Ferrari*, 2009) dataset following the same experimental setup in *Sapp et al.* (2010b). The pose estimation performance is shown in Percentage of Correct Parts (PCP) for each part in Table 6.2. PCP is the typical measure of performance on the buffy dataset (*Ferrari et al.*, 2008b). It uses a matching criteria based

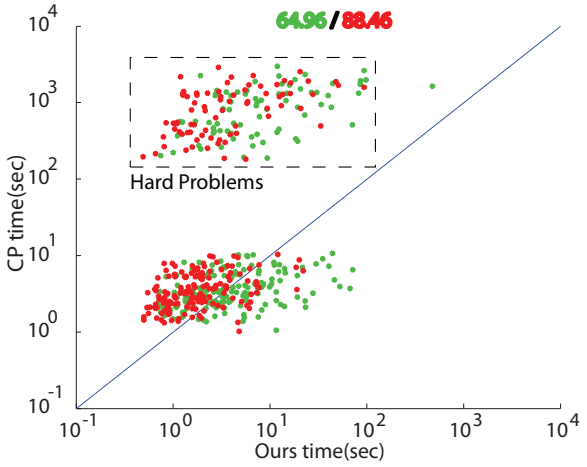


Figure 6.5: Scatter plot for the time comparison between the MPLP-CP method (y axis) and our methods (x axis) on the Buffy dataset. Green indicates results of our NoGVS_VHO_NoOBMS approach and red indicates results of our NoGVS_VHO_OBMS approach. The two percentages on top indicate how many times our two approaches are faster than the CP, respectively.

on both endpoints of each part (e.g., matching the elbow and the wrist correctly): the state of a body part is correct if the endpoints corresponding to the state (u, v, l, ϕ) are, on average, within r of the corresponding ground truth segments, where r is a fraction of the ground truth part length. By varying r, a performance curve is produced where the performance is measured in the percentage of correct parts (PCP) matched with respect to r. In our experiment, we set $r = 0.5$ which is commonly used for evaluation. We report the CPS performance reproduced by the public available code released by *Sapp et al.* (2010b). We also explore the effect of the connectivity of the model by training two sub-models with 13 and 7 pair-wise interactions on the Buffy dataset. Our fully connected model ("Ours F" in Table 6.2) outperforms the sub-models and CPS for most parts on both datasets. Moreover, our method achieves an average PCP of 85.7% which is significantly better than another fully-connected model (*Tran and Forsyth*, 2010) (67.6%) and on par with the state-of-the-art method (*Yang and Ramanan*, 2011) (89.1%) on the Buffy dataset.

Computation Efficiency Analysis. We also compare the computation efficiency

All/ Hard Problem	Avg. BB (sec)	Avg. Total (sec)	Avg. #branches
NoGVS_NoVHO_NoOBMS	37.668/ 76.823	38.275/ 77.413	626K/ 1243K
NoGVS_VHO_NoOBMS	6.637/ 12.242	7.175/ 12.762	148K /267K
GVS_NoVHO_NoOBMS	37.787/ 55.736	38.259/ 56.193	189K/ 292K
GVS_VHO_NoOBMS	12.029/ 17.990	12.483/ 18.436	**80K/ 122K**
NoGVS_NoVHO_OBMS	22.669/ 47.881	23.317/ 48.511	626K/ 1243K
NoGVS_VHO_OBMS	4.076/ 7.961	4.721/ 8.585	148K /267K
GVS_NoVHO_OBMS	8.387/ 13.548	8.890/ 14.036	189K/ 292K
GVS_VHO_OBMS	**2.920/ 4.637**	**3.408/ 5.109**	**80K/ 122K**
MPLP-CP	N.A.	344.539/ 1096.190	N.A.

Table 6.3: Time break-down and number of branches for our methods (first 8 rows) and MPLP-CP (last row). For each measurement, we show average processing times over the whole Buffy dataset (left entry) as well as over the set of hard problems (right entry). BB denotes time to run the BB search algorithm, and Total denotes the BB search time plus the time to build BMTs.

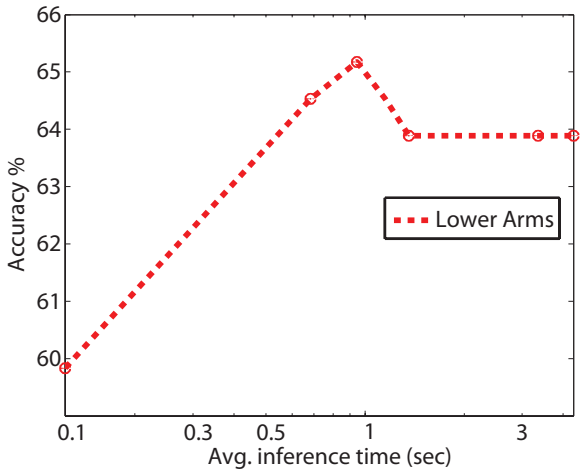


Figure 6.6: Trade-off between accuracy (y axis in PCP) and efficiency (x axis in time) in estimating "lower arms".

of different variants of our methods with MPLP-CP method. Notice that we calculate the bound in Eq. 6.10 by setting $\beta = 0.5\theta$ in this experiment, since it is more costly to do message passing to search for β at the beginning. The left entries of the first four rows in Table 6.3 show the average time break-down and number of branches for four variants of our BB methods. These correspond to : 1) using NLPDG to guide the variable selection (GVS) or not (NoGVS); 2) using domain knowledge for variable hypothesis ordering (VHO) or not (NoVHO) (see Sec. 6.3.3 for different branching

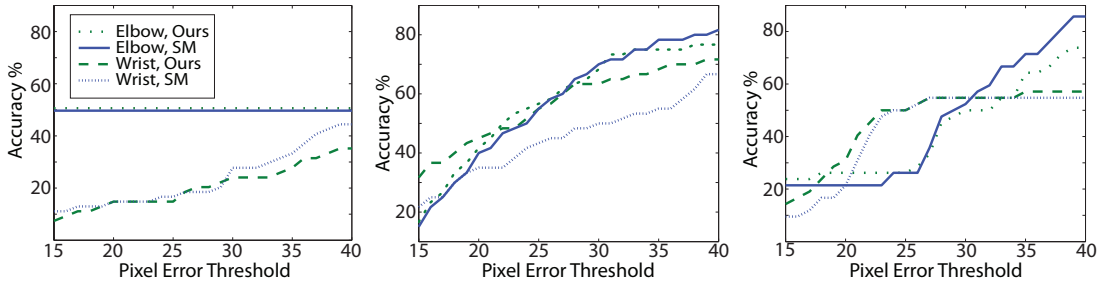


Figure 6.7: Quantitative results on three sequences in the VideoPose2.0 testset that are best represented by the pre-trained model. The predicted joint location is correct if its distance between the ground truth location is smaller than the specified pixel error threshold (x-axis). In the first column, both methods (our method and Stretchable Models (SM) (*Sapp et al.*, 2011)) only detect half of the elbows. In the second sequence (Center), our method achieves almost consistently $\sim 10\%$ better wrist accuracy than SM (*Sapp et al.*, 2011) does. In the last sequence (Right), our method obtains better accuracy when the pixel error threshold is small for both elbow and wrist.

strategies); 3) using Opportunistic Branch Max Search (OBMS) or not (NoOBMS). The smallest average total time is achieved by our method ("GVS_VHO_OBMS") using NLPDG to guide the variable selection, using the domain knowledge for variable hypothesis ordering, and using OBMS. Our best method takes 0.222 hours in total to recognize poses in the whole Buffy dataset which contains 234 testing images. This is 101 times faster than CP method (22.4 hours). A scatter plot in Fig. 6.5 shows the time comparison for each example. It shows that our method with OBMS (red dots in Fig. 6.5) is faster than our method without OBMS (green dots in Fig. 6.5) (on average about 1.5 times faster). Moreover, we identify two groups of examples. The group on the top is a set of hard examples since the CP method needs to search for more complex constraints in order to solve these problems. We observe that our method is faster than CP in 88% of the images in the dataset. Moreover, our BB algorithm requires less memory usage (on average 640MB) than the CP method (on average 7GB).

We also explore the trade-off between the pose estimation accuracy and inference time by allowing our method to stop early (increasing ϵ in line 12 of Algorithm 1). As shown in Fig. 6.6, when we allow approximate inference to run on average for 1.5

seconds, the algorithm already reaches the same performance as the exact inference algorithm which takes more than 3 seconds on average. Typical results of both the loopy model and the original tree model (*Sapp et al.*, 2010b) on Buffy as well as Stickmen datasets are shown in Fig. 6.8.

6.4.5.2 HPE Problem Given a Video Sequence

Sapp et al. (2011) propose the Stretchable Model (SM) which models 6 body joint locations in each frame and captures interactions within frames as well as across consecutive frames. Since no existing methods can solve exact inference efficiently on such a large loopy model ($\sim$200 variables and a few hundred hypotheses per variable), authors in *Sapp et al.* (2011) propose multiple inference techniques to solve the joint estimation problem. These includes: A) exact inference on relaxed models, B) approximate inference on a full model (dual decomposition). Their experimental results on VideoPose2.0 dataset (*Sapp et al.*, 2010c) show that (A) is both more efficient and accurate than (B). In this experiment, we first use message passing to select the best β in order to avoid having a looser upper bound. We show that our BB algorithm can be directly applied to exactly infer the MAP solution over their pre-trained model. 13 out of 18 test sequences are solved within 20 minutes (on average 5.546 minutes). Whereas, a dual decomposition approximate inference algorithm (*Globerson and Jaakkola*, 2008) can only solve 4 out of 18 problems. Moreover, the CP method can only solve the same 4 problems within one hour. Interestingly, although our method solves the MAP estimation exactly, our method achieves similar prediction accuracy in estimating the position of the elbow but about 5% lower accuracy in predicting the position of the wrist. This observation suggests that the pre-trained model does not appropriately represent the video sequences. Indeed, we have noticed that the learned model typically assigns much lower values to the ground truth assignments than it does with the values of the MAP assignments (on average

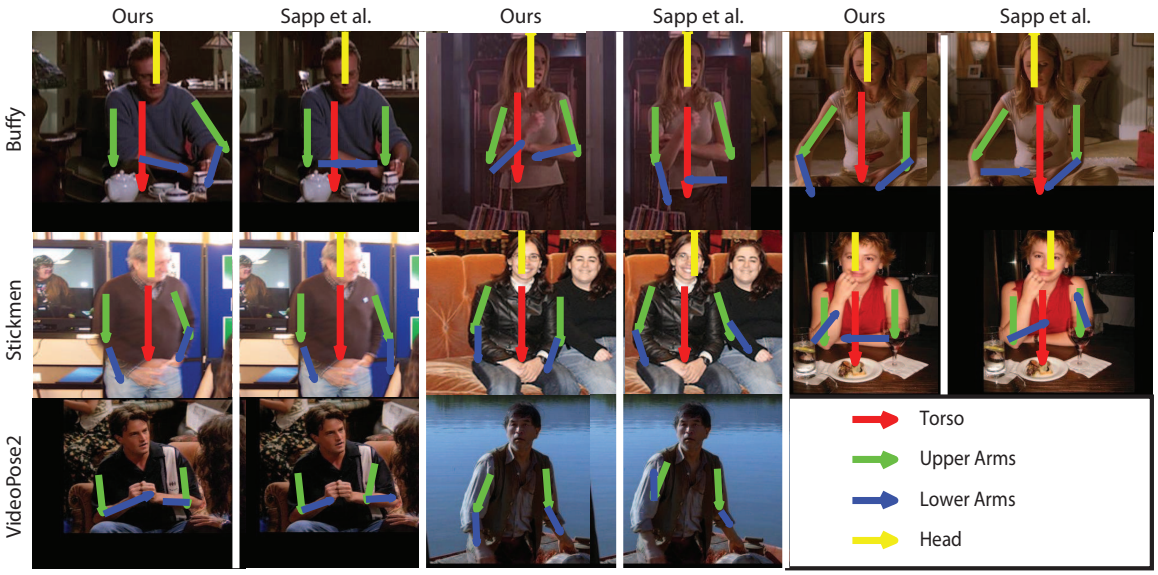


Figure 6.8: Typical results from Buffy, Pascal Stickmen, and VideoPose2 datasets shown using a Stickmen representation from top to bottom, respectively. In each set of results, we show our results on the left and the Sapp et al.'s results (*Sapp et al.*, 2010b, 2011) on the right.

60% smaller). This means that the model often does not agree with the ground truth assignments. For example, in the first test sequence, the values of the MAP, *Sapp et al.* (2011)'s approximate inference, and the ground truth assignments are $20755, 17901, 9257$, respectively. The value of the ground truth assignment is closer to the value of the approximate inference assignment by *Sapp et al.* (2011) than to the value of the MAP assignment. We follow this observation and select the top 3 sequences where the value of the ground truth assignment is closer to value of the MAP assignment with respect to the absolute difference between the values of the ground truth and the approximate inference assignments by *Sapp et al.* (2011). In these cases, exact inference obtained by our method achieves comparable or superior accuracy (Fig. 6.7). Typical estimated body joint locations are shown in Fig. 6.8. This suggests that a better set of model parameters must be learned to fully demonstrate the power of the loopy model.

6.4.6 Other Application: Protein Design

Since our proposed method can solve the MAP inference problem over any pairwise MRF, we demonstrated that our method can be helpful in domains beyond computer vision such as protein design problems in molecular biology. The protein design problem consists of finding a sequence of amino-acids that are as stable as possible for a given 3D shape of the protein. This is done by finding a set of amino-acids and rotamer configurations that minimizes an approximate energy. *Yanover et al.* (2006) introduce a dataset and model this problem as a MAP-MRF inference problem. Due to the large combinations of rotamers and amino-acids at each location, the hypothesis space is large (up to 180 hypotheses per variable for most cases). Thus, these problems require a significant amount of time to solve (e.g., ranging from several minutes to several days) (*Sontag et al.*, 2008b). Since computation efficiency is an important factor in many applications, we show that our method can be used to solve a number of problems faster than CP method within a limited amount of time. We show in Fig. 6.9 that given a 20 minute maximum time budget T (the same budget as used in the synthetic data experiment), our best BB method ("Ours+GVS") in synthetic data experiment i) consistently solves more problems than MPLP-CP does for all T, and ii) is consistently faster than MPLP-CP is when solving the same number of problems (5.8 times faster in average).

6.5 Conclusion

In this chapter, we have proposed an efficient BB method to solve the MAP-MRF inference problem by leveraging a data structure and a novel search routine to reduce the time complexity. Moreover, we have proposed a novel branching strategy that reduces the number of branches evaluated. Our method is faster than the proposed improved naïve BB algorithm and state-of-the-art methods (*Sontag et al.*, 2008b;

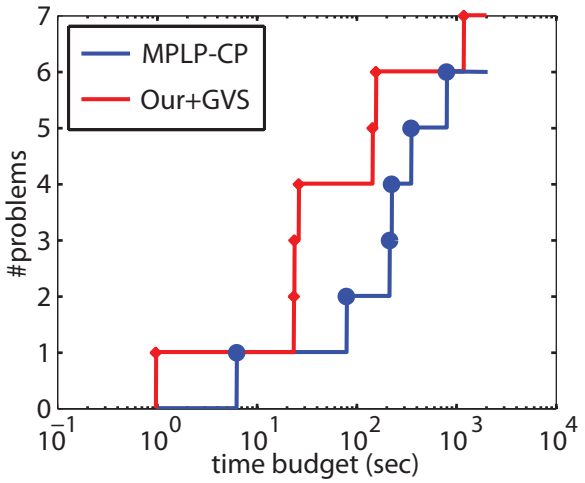


Figure 6.9: Comparison between our method ("Ours+GVS") (red) and MPLP-CP (blue) on protein design problems. Number of problems solved (y-axis) given different time budgets (x-axis) up to 20 minutes are plotted.

Marinescu and Dechter, 2007) on synthesized data and two problems in computer vision and computational biology where the number of hypotheses is large. Finally, we have shown that when domain knowledge is available, a significant speed-up can be achieved.

CHAPTER VII

Models for Capturing Interplay between Objects and Scene Layout

As more and more reliable and accurate object recognition methodologies become available, increasing attention has been devoted to the design of algorithms that go beyond the individual object detection problem and seek to coherently interpret complex scenes such as the one in the center of Fig. 1.8. Coherent scene interpretation requires the joint identification of object semantic labels (object classification), the estimation of object 2D/3D location in the physical scene space (2D object localization, depth inference) as well as the estimation of the geometrical structure of the physical space in relationship with the observer. The latter includes the 3D geometry of the supporting surfaces (i.e., orientation and location of the surfaces that are supporting objects in the scene) as well as their 2D extent in the image (supporting surface segmentation).

Researchers have recognized the value of contextual reasoning as an important tool for achieving coherent scene understanding. Two main types of contextual information have been explored: Semantic context and geometrical context. Semantic context captures the typical semantic relationship among object classes co-occurring in the same scene category (*Torralba et al.*, 2003; *Li and Fei-Fei*, 2007; *Li et al.*, 2009a; *Ladicky et al.*, 2010a; *Gonfaus et al.*, 2010; *Rabinovich et al.*, 2007) (e.g. cars

and roads are likely to co-occur within an urban scene). Geometrical context captures typical spatial and geometrical relationships between object classes and the scene geometric structure (*Gupta and Davis*, 2008b; *Sudderth et al.*, 2008; *Hoiem et al.*, 2006, 2008; *Gould et al.*, 2009a; *Hedau et al.*, 2009; *Heitz et al.*, 2008; *Li et al.*, 2010; *Saxena et al.*, 2009; *Bao et al.*, 2010b) (e.g., a car is likely to be located on top of the road and unlikely to float in the air).

In this chapter, we present a new way to establish the contextual relationship between objects and the scene geometric structure. Specifically, we are interested in modeling the relationship between:

- **objects and their supporting surface geometry.** Geometrical configuration of objects in space is tightly connected with the geometry (orientation) of the surfaces holding these objects (Fig. 7.1 - Intuition 1);

- **objects and observer's geometry.** Object appearance properties such as the scale and pose are directly related to the observer's intrinsic (focal length) and extrinsic properties (camera pose and location) (Fig. 7.1 - Intuition 2);

- **objects and supporting regions.** The statistics describing the 2D appearance (features, texture, etc.) of foreground objects are different from those describing the 2D appearance of the supporting surfaces (Fig. 7.1 - Intuition 3).

Following these intuitions, our main contributions are:

1. A new coherent framework to model contextual reasoning for object detection, 3D layout estimation, and object supporting region segmentation, which is based on the mutual interactions among three modules: i) object detector; ii) scene 3D layout estimator; iii) object supporting region segmenter (Fig. 1.8). The interactions between such modules capture the contextual relationships discussed above.

2. Our approach leverages the estimations returned by the detector (i.e, class label, object location, scale, and pose) in order to establish such contextual relationship. Thus, it does not rely on using external holistic or local surface detectors (*Hoiem et al.*, 2005a; *Hedau et al.*, 2009) or explicit 3D data (*Cornelis et al.*, 2006; *Brostow et al.*, 2008).

3. Unlike other methods such as *Li et al.* (2009a); *Ladicky et al.* (2010a); *Gonfaus et al.* (2010); *Rabinovich et al.* (2007); *Li and Fei-Fei* (2007); *Torralba et al.* (2003) where the typical co-occurrence between objects and background (e.g., a car on road) is learnt during a training stage and used to provide semantic context, our method exploits the local appearance coherency of objects and supporting surfaces (within a specific image) as well as the typical joint spatial arrangement of objects and supporting surfaces in order to reinforce (or weaken) the presence of objects and to segment the object from its supporting surface.

4. The estimation of the scene 3D layout (orientation and location of the supporting planes, location of objects in 3D and camera parameters (focal length)) is carried out from just one un-calibrated single image. Unlike other methods such as *Hoiem et al.* (2005a); *Hedau et al.* (2009) wherein assumptions about the relationship between the geometry of the ground plane and the camera parameters are made (e.g, the camera is located at given height from the ground plane and only one ground plane is allowed), our approach can handle multiple supporting planes and arbitrary observer viewing directions.

5. Most importantly, we introduce a new paradigm where the object detector module is capable of adaptively changing the confidence in establishing whether a certain region of interest contains an object (or not) as new evidence is gathered from the plane 3D layout estimator and supporting region segmenter. Our method is conceptually different from other methods such as *Hoiem et al.* (2008);

Cornelis et al. (2006) where geometric context only modifies the confidence of the object detector *a posteriori* (i.e., the detector always produces the same confidence output which is subsequently modified by a geometric context module). This enables an iterative estimation procedure where the detector *itself* becomes more and more accurate as additional evidence about a specific scene becomes available.

6. We validated our method against an augmented table-top dataset (introduced in Chapter IV) (so as to test the system level properties of our framework) as well as on existing databases (viz. labelme (*Russell et al.*, 2008) and Office (*Sudderth et al.*, 2008) datasets). The experiments demonstrate that our method: i) is scalable to generic scenes (indoors, outdoors) and generic object categories; ii) achieves state-of-the-art detection results; iii) can successfully infer scene 3D layout information and reason about supporting regions from a single image in challenging and cluttered scenes.

The rest of this chapter is organized as follows. In section 7.1, we first describe in detail the model representation and learning procedure of our object detector, 3D layout estimator, and object supporting region segmenter modules; we then summarize the types of interactions we used during inference. In section 7.2, we show quantitative and qualitative experimental results on three different datasets. Finally, we draw conclusions in section 7.3.

7.1 Geometrical Context Feedback Loop

In this section we first give an overview of our model which fuses the information from the object detector (OD), layout estimator (LE), object supporting region segmenter (RS) modules in a coherent fashion (Fig. 1.8).

(a) Intuition 1 (b) Intuition 2 (c) Intuition 3

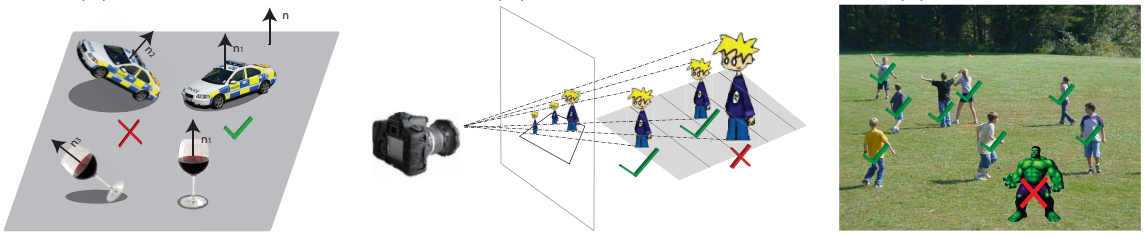


Figure 7.1: List of intuitions in this chapter and comparison with related works. (a) Intuition 1: Rigid objects typically lie up-right on the supporting plane. The coherence between object pose and plane normal is used by our algorithm as well as our preliminary work (*Bao et al.* (2010b), Chapter IV), but not in *Hoiem et al.* (2006); *Gould et al.* (2009a); *Hoiem et al.* (2008). (b) Intuition 2: Under the perspective camera model, the size of an object in the 2D image is an inversely proportional function of its distance to the camera when the object pose is fixed. *Hoiem et al.* (2006, 2008) use this relationship too. (c) Intuition 3: The statistics describing the 2D appearance (features, texture, etc.) of foreground objects are likely to be different enough from those describing the 2D appearance of the supporting surfaces (e.g., we rarely see green hulk playing on grass.). Unlike *Rabinovich et al.* (2007); *Gupta and Davis* (2008b); *Li et al.* (2009a); *Sudderth et al.* (2008) where the typical co-occurrence between objects and background is used to provide semantic context, we exploit the local appearance coherency of objects and supporting surfaces (within a specific image) as well as the typical joint spatial arrangement of objects and supporting surfaces.

7.1.0.1 Model Overview

The critical building block of our system is the object detector as it generates cues (e.g., object scale, location, and pose) that can be fed to the layout estimator and the region segmenter modules. We use a novel detector called Depth-Encoded-Hough-Voting (DEHV) which is based on our own work (Chapter IV). DEHV has the crucial capability to produce an object detection confidence score which is not just a function of the image local appearance but also a function of the geometric structure of the scene (i.e., the 3D layout information **L** and supporting region information **S**). This information restricts the object's likely scale, pose, and background/foreground configurations. At the beginning of the inference process (iteration 1 of the loop), no information about 3D layout information **L** and supporting region information **S** is available so the detector returns a number of detection hypotheses by explor-

ing all object categories, the complete scale space, all possible object poses, and all background/foreground configurations in the image. Each detection hypothesis is associated to the object class O, location x, scale (1-to-1 mapped to depth d^o (Eq. 4.4)), and pose ϕ^o (zenith and azimuth angles). This information is fed to both the layout estimator and region segmenter modules. In turn, the layout estimator module produces an estimate of the 3D layout of the scene. The layout information $\mathbf{L}$ includes the camera focal length f and a set of supporting planes L_i, where L_i is parameterized by camera-to-plane height η and 3D orientation n in the camera reference system. By following intuitions 1 and 2 (Fig. 7.1), this can be done if at least three objects are detected in the image (proved by *Bao et al.* (2010b), see Proposition VII.1). Moreover, as we shall see in the region segmenter module, using the object's location and scale provided by the detector, the region segmenter module returns probability of each pixel belonging to a supporting region $\mathbf{S}(l)$, where l specifies the 2D location of the pixel. This information allows us to identify the extent of the supporting region. Following intuition 3 (Fig. 7.1), this can be done by using a superpixel representation to capture local appearance coherency of objects and supporting surfaces, and by exploiting the typical joint spatial arrangement of objects (whose location and scale are given by the detector) and supporting regions in the image. In turn, the outputs for the layout estimator and region segmenter modules are fed back to the object detector module and are used to help reduce the detector's search space (i.e., object scale, location, and pose). Specifically, location and orientation of the supporting planes in the camera reference system, and camera focal length (returned by the layout estimator) simplify the complexity of the scale and pose search space. Moreover, the estimation of the object supporting surface (returned by the region segmenter) helps remove spurious patches (features) that are used to build the Hough voting score in the DEHV. Overall, the detector leverages these additional pieces of evidence to increase the confidence of true positives and decrease that of false alarms following

the iterative inference procedure described in Sec. 7.1.3. An overview of the inference procedure is shown in Algorithm 8.

7.1.1 Model Representation

We introduce in detail our three modules (object detector, layout estimator, and supporting region segmenter) in this section.

7.1.1.1 Object Detector Module

We employ a modified version of the Depth-Encoded-Hough-Voting (DEHV) object categorical detector (Chapter IV) to obtain an estimate of the object location, scale, pose, and depth. Similar to *Leibe et al.* (2004), the DEHV detector constructs a voting space $V(O, x|D)$ (Eq. 7.1), where O is object class (i.e. an object category with a unique pose), x is the object's 2D image location and scale (i.e. a 2D bounding boxes enclosing the object), D is the depth information (i.e, the distance from the camera to the object), and different poses are encoded as different object classes. The voting space V is constructed by collecting probabilistic votes cast by the set of patches describing object class O. Notice that the voting space $V(O, x|D)$ depends on the geometric structure of the scene since the object hypothesis (O, x) is related to D. This novel property gives DEHV the ability to detect objects whose locations and poses are compatible with the underlying layout of the scene.

The DEHV detector. Let $\{(C_j, d_j^p, l_j)\}$ be a set of patch attributes, where C_j denotes the appearance of image patch j centered at image location l_j, and d_j^p denotes the distance from the camera center to the corresponding 3D location of a patch. Appearance C_j is a discrete codeword label (*Csurka et al.*, 2004). Notice that each patch is associated with a physical 3D distance to the camera which affects the size of the patch in 2D. We define $V(O, x|D)$ as the sum of individual probabilities over

all observed images patches at location l_j and for all possible depth $d_j^p \in D$, i.e,

$$V(O,x|D) = \sum_j \sum_{d_j^p \in D} p(O,x,C_j,d_j^p,l_j)$$
$$= \sum_j \sum_{d_j^p \in D} p(x|O,C_j,d_j^p,l_j)p(O|C_j,d_j^p,l_j)p(C_j|d_j^p,l_j)p(d_j^p|l_j)$$

where the summation over j aggregates the evidence from individual patch location, and the summation over depth d_j^p marginalizes out the uncertainty of depth corresponding to each image patch location. Since C_j is calculated deterministically from l_j and d_j^p, and assuming O only depending on C_j, we obtain:

$$V(O,x|D) \quad \propto \quad \sum_j \sum_{d_j^p \in D} p(x|O,C_j,d_j^p,l_j)p(O|C_j)p(d_j^p|l_j)$$

We further assign image patches with different depths to different index j. As a result, we can take only the summation over patch index j and obtain the equation below.

$$V(O,x|D) \propto \sum_j p(x|O,C_j,d_j^p,l_j)p(O|C_j)p(d_j^p|l_j) \tag{7.1}$$

The first term $p(x|O,C_j,d_j^p,l_j)$ characterizes the distribution of object location x given the predicted object class O and patch attributes $\{(C_j,d_j^p,l_j)\}$. The second term, $p(O|C_j)$ captures the probability that each codeword belongs to an object class O. Finally, $p(d_j^p|l_j)$ models the uncertainty of the depth information of patch j.

Similar to Chapter IV, our detector enforces a 1-to-1 mapping m between scale s and depth d for each patch. This way, given the 3D information, our method deterministically selects the scale of the patch at each location l, and given the selected patches, our method can infer the underlying 3D information (Fig.4.4). For details, please see Eq. 4.4 in Chapter 4.2.1.

Generating object hypotheses. After accumulating votes into the Hough voting

space $V(O, x|D)$, a set of detection hypotheses $\{(O_i, x_i)\}$ corresponding to peaks in the voting space can be obtained. Given the object class O and the 2D location x, the image patches that cast votes for the hypothesis can be retrieved (later referred as supporting image patches). Hence, the depth to image patch information described in Eq. 4.4 can be used to calculate the depths of all image patches $\{d_j^p\}$. The depth of the object d^o is defined as the median depth of the depths of image patches d_j^p. Similarly, we can also retrieve the corresponding zenith angles $\{\phi_j^p\}$ corresponding to all supporting image patches, and we use a verification support-vector-machine (SVM) classifier to find the most likely zenith angle of the object ϕ^o among the candidate zenith angles. Hence, the final output of the detector is a set of hypotheses $\{(O_i, x_i, \phi_i^o, d_i^o)\}$.

One of the main contributions of this chapter is that the detector can modify its behavior as knowledge about the scene layout (denoted by $\mathbf{L}$) and the object supporting regions (denoted by $\mathbf{S}$) are available.

Knowledge of Supporting Region S. The region segmenter module provides knowledge about the supporting region $\mathbf{S}$ and affects $p(O|C_j)$. We replace $p(O|C_j)$ with $p(O|C_j, l_j, \mathbf{S})$, and we show that it can be decomposed as follows,

$$\begin{aligned} p(O|C_j, l_j, \mathbf{S}) &= p(O, O \notin bg|C_j, l_j, \mathbf{S}) && (7.2) \\ &= p(O|O \notin bg, C_j)p(O \notin bg|C_j, l_j, \mathbf{S}) && (7.3) \end{aligned}$$

where $O \notin bg$ means the object class does not belong to the background class. The first equality is true since we only need to evaluate object classes that belong to the foreground object classes during Hough voting. The second equality follows the chain rule in probability theory and conditional independent assumption between $(l_j, \mathbf{S})$ and O given $(O \notin bg, C_j)$. As a result, only the second term $p(O \notin bg|C_j, l_j, \mathbf{S})$ is

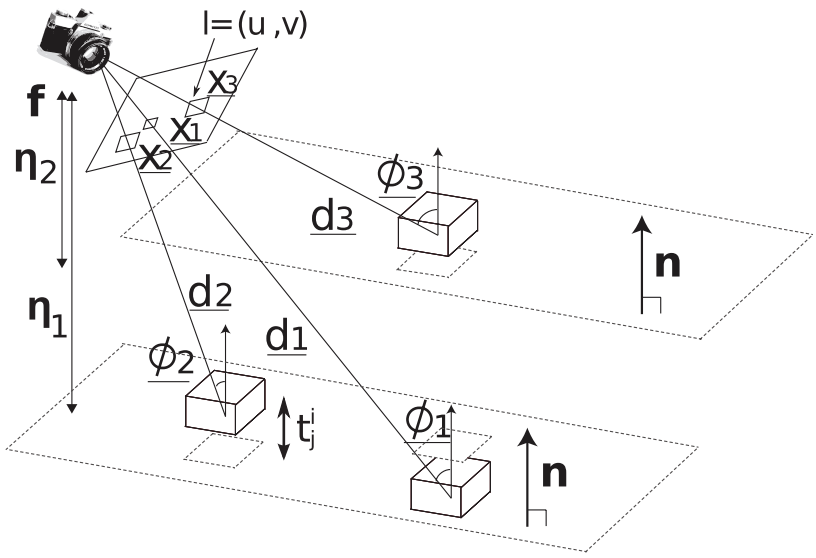


Figure 7.2: The notations used in the layout estimator module. The bold fonts indicate parameters that are estimated by the layout estimator module. The underline fonts indicate parameters that are estimated by the object detector module. In this example, two planes are visualized. The measurements are: x: object's 2D image location and scale; d: object-to-camera distance; ϕ: observed object pose; t: the 3D object center to supporting place distance; l: observed patch image location. The unknowns are: f: camera focal length; n: the plane normal; η: the camera-to-plane distance.

related to the supporting region $\mathbf{S}$. We define $p(O \notin bg|C_j, l_j, \mathbf{S})$ as follows,

$$p(O \notin bg|C_j, l_j, \mathbf{S}) := p(O \notin bg|C_j)(1 - \mathbf{S}(l_j)) \tag{7.4}$$

where $p(O \notin bg|C_j)$ is the probability that the codeword C_j does not belong to the background class, and it is reweighed by $1 - \mathbf{S}(l_j)$. Here, $\mathbf{S}(l_j)$ is the probability that a pixel at location l_j belongs to a supporting region which is equivalent to saying that such a pixel belongs to a background region (see the region segmenter module for details). This probability is estimated by the region segmenter and allows the algorithm to reduce the importance of patches that are likely to belong to the supporting region.

Knowledge of Scene Layout L. The layout estimator module provides knowledge about the scene layout $\mathbf{L}$ and affects $p(d_j^p|l_j)$. In order to explicitly incorporate

knowledge about the scene layout, the term $p(d_j^p|l_j)$ is calculated as follows:

$$p(d_j^p|l_j, \mathbf{L}) \propto \sum_{i \in |\mathbf{L}|} \delta(t_j^i) \tag{7.5}$$

where t_j^i is the distance from the 3D location of the image patch j to the ith plane parameterized by its normal direction n and camera height η (See Eq. 7.6 in section of the layout estimator module for details); and $|\mathbf{L}|$ denotes the number of plane hypotheses. Notice that knowledge of the scene layout $\mathbf{L}$ allows the algorithm to estimate the probability that an image patch j is located at depth d_j^p from the camera. Hence, effectively, the search space of object scale in 2D is reduced.

To summarize, modified DEHV takes into account the knowledge of supporting region S by deemphasizing votes from supporting regions to the space $V(O, x|D)$. Furthermore, the uncertainty of the corresponding depth d_j^p of each image patch j is reduced by the knowledge of surface layout $\mathbf{L}$. Hence, the noise in the voting space $V(O, x|D)$ is reduced and the number of false detections decreases. Notice that the detection hypotheses $\{(O, x)\}$ may also be further pruned by checking if the object bounding box x is consistent with the underlying layout information $\mathbf{L}$ (similarly to *Hoiem et al.* (2006)).

7.1.1.2 3D Layout Estimator Module

The goal of the 3D layout estimator is to estimate the 3D layout $\mathbf{L}$ associated with a single image from candidate object detections. Our layout estimator module is built upon *Bao et al.* (2010b). However, instead of using the probability inference in *Bao et al.* (2010b), we employ Hough voting to efficiently estimate the 3D layout. As shown in Fig. 7.2, $\mathbf{L}$ contains the camera focal length f and a set of supporting planes $\{L_i\}$ each parameterized by camera-to-plane height η and 3D orientation n. Notice that the orientation n is a normalized vector such that $\|n\|_2 = \sqrt{n_1^2 + n_2^2 + n_3^2} = 1$,

and (n, η) specifies a unique plane in 3D such that any 3D point $q \in R^3$ lying on the plane satisfies $q^T n = \eta$. Moreover, the closest distance t from a 3D point q to a plane parameterized by (n, η) can be calculated as follows,

$$t = |q^T n - \eta| \tag{7.6}$$

The 3D point q corresponding to the 2D point $l = (u, v)$ with depth d is equivalent to the normalized ray from camera center to the 2D point on the image plane times the depth:

$$q = \frac{[u \ v \ f]^T}{\sqrt{u^2 + v^2 + f^2}} \times d \tag{7.7}$$

Following intuitions 1 and 2, we formulate the plane estimation problem as a Hough-voting problem. A Hough voting space Q is constructed with axes associated with the plane's orientation n, the camera height η, and the focal length f. Each candidate object detection $(O_i, x_i, \phi_i^o, d_i^o)$ casts votes in Q for a set of camera focal length $\{f\}$ and supporting plane $\{n, \eta\}$ following the distribution $p(n, \eta, f|O_i, x_i, \phi_i^o, d_i^o)$. We use geometrical constraints to help compute $p(n, \eta, f|O_i, x_i, \phi_i^o, d_i^o)$. The geometrical relationship relating object detections and the supporting planes is derived in the same manner as *Bao et al.* (2010b). As illustrated in Fig 7.2, let (u_i, v_i) be the center location of object detection location x_i. The zenith angle ϕ_i[1] is the angle between the light ray (u_i, v_i, f) from the camera to the object and the plane normal n_1, n_2, n_3.

[1]Here we omit the superscript o to have a concise notation.

The layout $\{f, \eta, n\}$ and its supporting object satisfies the following equations,

$$\begin{cases} u_i n_1 + v_i n_2 + f n_3 & = -\cos(\phi_i)\|u_i \; v_i \; f\|_2 \\ \sqrt{(n_1)^2 + (n_2)^2 + (n_3)^3} & = 1 \\ \eta & = d_i^o * \cos(\phi_i) \end{cases} \tag{7.8}$$

Given a candidate object detection $(O_i, x_i, \phi_i^o, d_i^o)$, we compute $p(n, \eta, f|O_i, x_i, \phi_i^o, d_i^o)$ as the following:

$$p(n, \eta, f|O_i, x_i, \phi_i^o, d_i^o) \propto \begin{cases} 1 & \text{if } (n, \eta, f) \text{ satisfies Eq. 7.8} \\ 0 & \text{otherwise} \end{cases} \tag{7.9}$$

The final voting space $Q(n, \eta, f)$ is defined as the weighted sum over distribution of each candidate detection as follow,

$$Q(n, \eta, f) \;\; = \;\; \sum_i p(n, \eta, f|O_i, x_i, \phi_i^o, d_i^o) V(O_i, x_i) \tag{7.10}$$

such that the contribution of each candidate detection is weighed by the detection score $V(O_i, x_i)$.

As a result, high values in the layout voting space Q is accumulated by geometrically consistent detection candidates. This model can easily incorporate scene layout with multiple supporting planes by associating each plane to a peak in the Hough voting space Q. However, in order to regularize the co-occurrence of multiple supporting planes, we assume all the supporting planes are parallel to each other similarly to the assumption in *Bao et al.* (2010b). This allow us to compress the Hough voting space to a lower dimension space $\hat{Q}(n, f)$ by summing over the axis of η in $Q(n, \eta, f)$. We first find the peak (n^*, f^*) in $\hat{Q}(n, f)$. Then, we select multiple peaks of $\{\eta\}$

in $Q(n^*, \eta, f^*)$. As shown in *Bao et al.* (2010b), it is necessary to have at least 3 non-collinear object supported by parallel planes to have a unique peak in $\hat{Q}(n, f)$ (see Proposition VII.1. It is important to point out that the 3 non-collinear objects do not have to be located on the *same* supporting plane. More specifically, since we assume that multiple planes are parallel to each other, we only need at least 3 objects to estimate the plane orientation, and each plane height can be estimated from one single object. Finally, the estimated layout $\mathbf{L} = (\{n, \eta\}, f)$ is fed to the detector to further reduce the uncertainty of the patches' depth distribution $p(d_j^p | l_j, \mathbf{L})$, as already described in Eq. 7.5.

In the following proposition, we proof the three objects requirement.

Proposition VII.1. *Equation (7.11) admits one or at most two non-trivial solutions of* $\{f, n_1, n_2, n_3\}$ *if at least three non-aligned observations* (u_i, v_i) *(i.e. non-collinear in the image) are available. If the observations are collinear, then Eq.(7.11) has infinite number of solutions.*

$$\begin{bmatrix} u_1 & v_1 & f \\ u_2 & v_2 & f \\ u_3 & v_3 & f \\ & \vdots & \\ u_N & v_N & f \end{bmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \begin{pmatrix} -\cos\phi_1 \sqrt{u_1^2 + v_1^2 + f^2} \\ -\cos\phi_2 \sqrt{u_2^2 + v_2^2 + f^2} \\ -\cos\phi_3 \sqrt{u_3^2 + v_3^2 + f^2} \\ \vdots \\ -\cos\phi_N \sqrt{u_N^2 + v_N^2 + f^2} \end{pmatrix} \tag{7.11}$$

Proof. Suppose at least three objects are not collinear in a image, then the rank of the left matrix in the left-hand side of Eq. (7.11) is 3. Therefore Eq. (7.11) provides 3 independent constraints. Recall the unknowns in Eq.(7.11) are n_1, n_2, n_3, f. With these constraints, each of n_1, n_2, n_3 can be expressed as a function of f, i.e. $n_i = n_i(f)$. Because $\|n\| = 1$, we obtain an equation about f:

$$\sum_{i=1...3} n_i^2(f) = 1$$

In the above equation, f appears in the order of f^2 and f^4. Therefore, there are at most two real positive solutions of f. Given f, $\{n_1, n_2, n_3\}$ can be computed as $n_i = n_i(f)$.

On the other hand, if all objects are collinear in the image, then infinite number of solutions of Eq.(7.11) exist. If all objects are collinear, the rank of the left matrix in the left-hand side of Eq.(7.11) is 2. Without loss of generality, assume $(u_1, v_1) \neq 0$. In such a case, after using Gaussian elimination, Eq.(7.11) will be in the following form:

$$\begin{bmatrix} \alpha & \beta & f \\ \gamma & \epsilon & 0 \\ 0 & 0 & 0 \\ & \vdots & \end{bmatrix} \begin{pmatrix} n_1 \\ n_2 \\ n_3 \end{pmatrix} = \begin{matrix} \zeta \\ \eta \\ 0 \\ \vdots \end{matrix} \tag{7.12}$$

If $\widehat{f}, \widehat{n}_1, \widehat{n}_2, \widehat{n}_3$ is solution, then $\widehat{f}, \widehat{n}_1 + km_1, \widehat{n}_2 + km_2, \widehat{n}_3 + km_3$ is also a solution of Eq. 7.12, where (m_1, m_2, m_3) is the non-trivial solution the following equation:

$$\begin{bmatrix} \alpha & \beta & f \\ \gamma & \epsilon & 0 \end{bmatrix} \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix} = 0$$

Hence, Eq. (7.11) admits infinite solutions. □

7.1.1.3 Supporting Region Segmenter Module

Following the observation that the supporting region is likely to have consistent appearance in the surrounding of the object and following intuition 3, our region segmenter module is capable of segmenting out the object from its supporting surface. We use a superpixel decomposition method (*Felzenszwalb and Huttenlocher*, 2004a) to identify regions with consistent appearance. Similarly to *Hoiem et al.* (2005a), multiple segmentation hypotheses $H = \{h_j\}$ are used to mitigate the problem of

segmentation errors. A segmentation hypothesis h_j is an ensemble of disjoint set of superpixels which fully cover the image. Each set of superpixels is a unique region r (Fig. 7.3).

Given a region $r \in h_j$ from the jth segmentation hypothesis, we train a logistic regression classifier to predict the probability $P(y|r, \{x, O\})$ which captures how likely the region r belongs to a supporting region (i.e. $y = 1$) or not. By averaging out the contribution of each segmentation hypothesis, we obtain the probability of a superpixel i belonging to a supporting region as follows,

$$P(y_i|\{x, O\}, I) = \sum_j P(y_i h_j(i)|\{x, O\}, I) \tag{7.13}$$

$$= \sum_j P(y_i|h_j(i), \{x, O\}) P(h_j(i)|I) \tag{7.14}$$

where I is the image and $h_j(i)$ is the image region including the ith superpixel in the jth segmentation hypothesis h_j. Notice that the output of the logistic regression $P(y_i|h_j(i), \{x, O\})$ is weighed by $P(h_j(i)|I)$ which indicates the probability that $h_j(i)$ is a region containing superpixel i given the image evidence. Given the probability that each superpixel belongs to a supporting region $P(y_i|\{x, O\}, I)$, and the mapping between pixel index to superpixel index, we obtain the probability (confidence) s that each pixel belongs to a supporting region. Finally, we denote by $\mathbf{S}$ the collection of probabilities $\{s_1, s_2, \dots\}$ for all pixels in the image. This allows the algorithm to calculate the probability $p(O \notin bg|C_j, l_j, \mathbf{S})$ in Eq. 7.3 that an image patch does not belong to the background.

7.1.2 Model Learning

In this section, we describe how the model parameters are learned in the object detector and supporting region segmenter modules in detail.

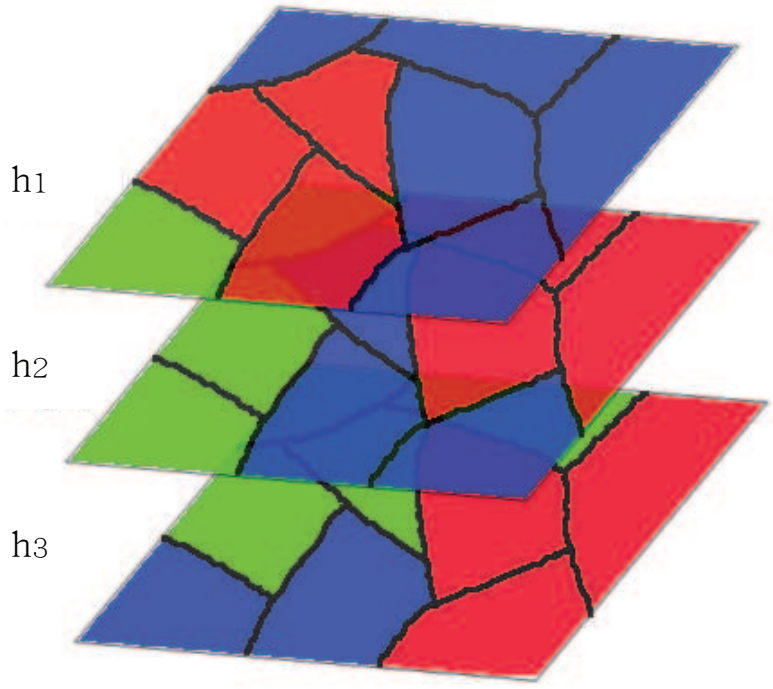


Figure 7.3: Illustration of the concept of multiple segmentation hypotheses, where different hypotheses are shown at different layers. Here we show three segmentation hypotheses, where each color indicates a region corresponding to a set of superpixels, and the image is partitioned into 9 superpixels separated by the dark boundaries.

7.1.2.1 Object Detector Module

Recall that the Hough voting space $V(O, x|D)$ in Eq. 7.1 aggregates votes from each unique combination of image patch location l_j, codeword label C_j, and depth d_j^p. Our goal is to learn the codebook mapping for a codeword C, the distributions of object class $p(O|.)$ (voting weight) and location $p(x|.)$ (voting direction). Notice that computation of $p(d|.)$ is already described in Eq. 7.5.

Similar to Chapter IV, we assume that for a number of training object instances, the 3D reconstruction D of the object is available. This corresponds to having available the distance (depth) of each object patch from its physical location in 3D.

Here we define location x of an object as a bounding box with center position q, height h, and aspect ratio a. We sample each image patch centered at location l and select the scale $s = m(l, d)$ using the 1-to-1 mapping described in Eq. 4.4. Then the

appearance $I(l,s)$ is extracted from the patch (l,s). When the image patch comes from a foreground object, we cache: 1) the information of the relative voting direction b as $\frac{q-l}{s}$; 2) the relative object-height/patch-scale ratio w as $\frac{h}{s}$; 3) the object aspect ratio a.

Random Forest Codebook. We use both the foreground patches (positive examples) and background patches (negative examples) to train a random forest discriminative codebook. Hence, the mapping $C(I(l,s))$ is a unique index of the leaf node in the random forest.

Voting Weight $p(O|.)$. For each codeword entry C_j, we use the training data to estimate $p(O|O \notin bg, C_j)$ and $p(O \notin bg|C_j)$ by counting the frequency that patches of O falls in the codebook entry C. Then, we can calculate $p(O|C_j, l_j, \mathbf{S})$ using Eq. 7.3.

Voting Direction. $p(x|O,C,s,l)$ can be evaluated given the cached information $\{(b_k, w_k, a_k)\}$ as follows:

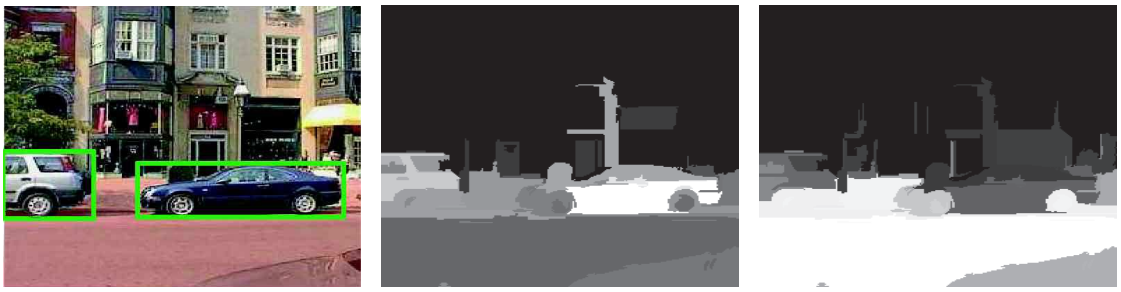
$$\begin{aligned} p(x|O,C,s,l) &= p((q,h,a)|O,C,s,l) \\ &\propto \sum_{k \in g(O,C)} \delta(q - b_k \cdot s + l, h - w_k \cdot s, a - a_k) \end{aligned}$$

where $g(O,C)$ is a set of patches from O mapped to codebook entry C. Notice that $p(x|O,C,s,l)$ is equivalent to $p(x|O,C,d^p,l)$ in Eq. 7.1, since $s = m(d^p, l)$.

7.1.2.2 Supporting Region Segmenter Module

Here, we describe how to learn the probability $P(h_j(i)|I)$ and $P(y_i|h_j(i), \{x, O\})$ in Eq. 7.14.

We model the intuition 3 by introducing the region-based statistics described below. Such statistics capture the joint typical spatial arrangement of objects (whose location and bounding box are given by the detector) and the object supporting regions in the image. Using these statistics, each region can be eventually labeled



(a) Ori. Img. (b) Avg. S1 (c) Avg. S2

Figure 7.4: Illustration of the segmentation statistics. Panel (a) shows the original image overlaid with ground truth supporting region (red) and ground truth object bounding boxes (green). Panel (b) and (c) show the average statistics over multiple segmentation hypotheses for S1, S2, respectively. Notice that white indicates higher value.

as supporting regions or not. Based on the candidate object detections $\{x, O\}$, our statistics are:

- The median detection confidence of those candidate object detections that sufficiently overlap with a candidate supporting region[2]. Intuitively speaking, the higher the statistic, the more likely the region belongs to the foreground region and the less likely belongs to a supporting region (Fig. 7.4 (b)).

- The 95th percentile of the detection confidence of the candidate object detections supported by the image region. Intuitively, the higher the statistic, the likelier the region belongs to a supporting region (Fig. 7.4 (c)).

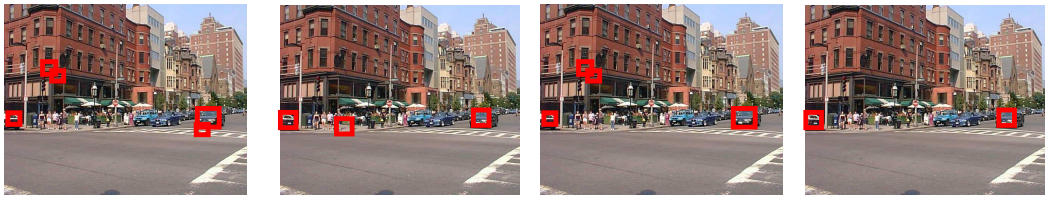
Using the designed statistics for each region r, we train a logistic regression classifier to estimate the probability $P(y|r, \{x, O\})$. Finally, $P(h_j(i)|I)$ is trained similarly to the segment homogeneity classifier described in *Hoiem et al.* (2007).

7.1.3 Model Inference using Context Feedback Loop

[2]When the area of the intersection between the foreground region (fg) and the object bounding box over the area of the object bounding box is bigger than 0.5, the object is considered as sufficient overlap with the foreground region.

Algorithm 8 Context Feedback Loop

$\mathbf{S}$:= empty
$\mathbf{L}$:= empty
for iter $\leq$ MaxIter **do**
  $\{(O, x, d^o, \phi^o)\} = \text{OD}(\mathbf{S}, \mathbf{L})$
  $\mathbf{L} = \text{LE}(\{(O, x, d^o, \phi^o)\})$
  $\mathbf{S} = \text{RS}(\{(O, x, d^o, \phi^o)\})$
  iter = iter + 1
end for



(a) OD (b) OD+LE (c) OD+RS (d) Full sys.

Figure 7.5: Interactions between different modules contribute to improve the detection performance. Panels show the results of the baseline detection (a), joint detection and 3D layout estimation (b), joint detection and supporting region segmentation (c), and our full system (d).

Our inference algorithm (see algorithm 8) starts by applying the object detector module assuming no prior knowledge about the scene layout $\mathbf{L}$ and supporting regions $\mathbf{S}$ is available. Hence, $p(d^p|.)$ in Eq. 7.1 is an uniform distribution which implies image patches can appear at any depth, and $p(O|C)$ in Eq. 7.1 is equal to $p(O|O \notin bg, C_j)p(O \notin bg|C_j)$.

The object detector returns the first set of candidate results $\{(O, x, d, \phi)\}$ (Fig. 7.5 (a)). Given the initial, possibly noisy, detections and pose estimations, the layout estimator generates an estimation of the possible layout parameters $\mathbf{L}$ which can be further used to improve detection (Fig. 7.5 (b)). Similarly, the region segmenter takes the noisy detection results to estimate the likely location of the supporting region which can be further used to improve detection (Fig. 7.5 (c)). In practice, the layout estimator and region segmenter act simultaneously and contribute to obtain more accurate detections which in turn yields more accurate layout and supporting

region estimates (Fig. 7.5 (d)). The system gradually converges into a steady state where the final object detection, pose estimation, layout estimation, and supporting region segmentation results are consistent with each other. Although we do not have a theoretical proof of convergence, experimental results suggest that such a point of convergence exists in most cases.

7.1.4 Implementation details

For the object detector, we use the following binning in the Hough voting space: i) 60 scales (from the 0.05 of the original scale multiplied by 1.05 to the original scale); ii) 10 object aspect ratio (from 0.6113 to 2.1611); iii) Each object class is discretized into 8 object poses corresponding to different azimuth angles; iv) For each scale, the object hypothesis is shifted in both horizontal and vertical directions by 2 pixels.

For layout estimator, the binning in the Hough voting process is: i) plane normal has 20 bins for tilt direction from 15° to 75° and 5 bins for camera-rotation from $-10°$ to 10°, ii) plane height has 20 bins from $30cm$ to $80cm$ for office dataset and from $1.5m$ to $2m$ for street dataset. iii) camera focal length has 20 bins from 0.8 to 1.25 fraction of an initial camera focal length guess.

7.2 Experiment

We evaluate quantitatively and qualitatively our system on three datasets. The first dataset is an augmented table-top object dataset (introduced in IV) with ground truth depth and foreground/background segmentation. We conduct experiments on object detection, plane layout estimation, and supporting region segmentation. We also evaluate our system on two publicly available datasets: a subset of label-me dataset (*Russell et al.*, 2008) (so as to compare our performance with the state-of-the-art method (*Hoiem et al.*, 2006)) and the office dataset (*Sudderth et al.*, 2008). Typical results on these 3 datasets are shown in Fig. 7.9,7.10,7.11.

	Loop Iterations	e_n (radius)	e_η (%)	e_f (%)	e^{FA}_{seg} (%)	e^{MS}_{seg} (%)
1 Plane	First Loop	0.125	25.9	13.2	2.07	51.2
	Final Loop	0.118	21.2	11.7	1.79	56.9
2 Plane	First Loop	0.133	24.9	12.1	4.00	49.0
	Final Loop	0.121	29.9	11.1	3.50	51.0

Table 7.1: Estimation errors (refer to Sec. 7.2.1 for definition of errors) of surface layout parameters (n, η, f), and supporting regions. Column 3∼5 show the errors of the estimated surface normal e_n, camera height e_η, and focal length e_f. The least two columns show the two types of errors of the supporting region segmentation. All five types of errors are further reduced as the number of iterations increases (the table reports results for the 1st and 7th iteration).

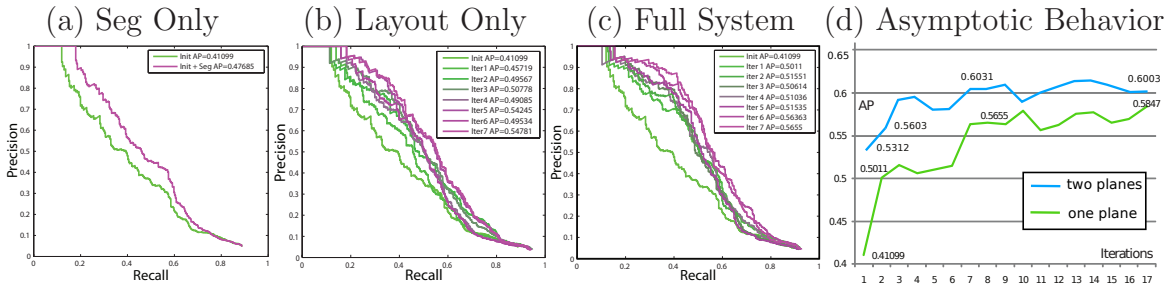


Figure 7.6: Detection performance using precision-recall measurement. Panel (a) compares baseline detection results (Chapter IV) with our system using only supporting region segmentation. Notice that joint object detection and supporting region segmentation lead to an one-time improvement only. Panel (b) shows results combining detector and layout estimator for 7 iterations. Panel (c) shows the results when all modules (the object detector, layout estimator, region segmenter) are used in the loop for 7 iterations. Notice the results in panel (a,b,c) are evaluated in the testset containing one single plane. Panel (d) shows that the performances of our full system for the single plane and two planes cases. Notice that the system appear to asymptotically converge to a steady state on both scenarios (with a single plane and with 2 planes.)

7.2.1 Table-top Object Dataset

We test our system on an augmented table-top object dataset introduced in Chapter IV which contains three common table-top object categories: computer mice, mugs, and staplers, where each image is associated to depth range data collected using a structure-light stereo camera. This allows us to easily estimate the ground truth 3D layout and supporting plane segmentation. The images are captured in

daily office place under generic lighting conditions. Please see the last three rows in Fig. 7.9 for examples. We follow the training procedure described in Chapter IV to train the DEHV detector using 200 images with their corresponding 3D information. The remaining 100 images (some with a single plane (80 images) and some with 2 planes (20 images)) are used for testing. Notice that the original dataset is introduced in Chapter IV contains 80 images. Each image from either training or testing sets contains $3 \sim 8$ object instances in random poses and locations[3]. During the testing stage, we only use 2D images and all the 3D information is inferred by our algorithm. Fig. 7.6 shows the overall Precision Recall curve (i.e, combining three classes (computer mice, mugs, and staplers)). We use the same definition of precision-recall as in *Everingham et al.* (2007). That is the precision is defined as $\frac{NumOfTruePositive}{NumOfDetection}$, the recall is defined as $\frac{NumOfTruePositive}{NumOfTrueObject}$, where a detection is considered to be true if

$$a_o = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \geq 0.5$$

where B_p is the predicted bounding box and B_{gt} is the ground truth bounding box.

Moreover, multiple detections of the same object in an image are considered false detections. The precision recall curve is calculated by varying the threshold to select the detections.

To obtain the initial detection result (first iteration of the loop), we apply the baseline detector (proposed in Chapter IV) with no information provided by the region segmenter and the layout estimator. Table 7.1 further shows the accuracy in estimating the layout parameters (n, η, f) and segmenting the supporting regions. Each of the errors are defined as follows: $e_n = \arccos(n_{est} n_{gt})$, $e_\eta = \frac{|\eta_{est} - \eta_{gt}|}{\eta_{est}}$, and $e_f = \frac{|f_{est} - f_{gt}|}{f_{est}}$, where subscript labels est and gt indicate estimated and ground truth values respectively. The last two columns report two types of segmentation errors: e_{seg}^{FA} and e_{seg}^{MS} are the amount to which the segmenter mistakenly predicts a foreground

[3]The training instances and testing instances are separated.

region as supporting region and the segmenter misses the truth supporting region, respectively. In detail, let I_P denotes the supporting region predicted by our model, I_{SR} denotes the ground-truth supporting regions, and I_F denotes the ground truth foreground objects. We define $e_{seg}^{FA} = \frac{|I_P \bigcap I_F|}{|I_F|}$ and $e_{seg}^{MS} = \frac{|I_P \bigcap I_{SR}|}{|I_{SR}|}$, where $|\bullet|$ counts the pixel number. The smaller e_{seg}^{FA}, the lower the false alarm rate is for confusing foreground pixels as background. The higher e_{seg}^{MS}, the larger the area our algorithm can classify as supporting region.

Both Table 7.1 and Fig. 7.6 (d) validate that the feedback loop is effective in improving i) object detection, ii) scene layout orientation estimation, iii) focal length estimation. The improvement of surface height η estimation is less consistent since the algorithm uses fewer objects to estimate the height of the surface (compared to the orientation of the surface). The improvement in segmenting the supporting region is also less significant. We believe that if more object categories are used, larger evidence about the appearance properties of the supporting regions can be produced, which in turn should produce more accurate segmentation results.

7.2.2 Label-Me Outdoor Dataset

We compare our system with another state-of-the-art method (*Hoiem et al.*, 2006) that uses geometrical contextual reasoning for improving object detection rates and estimating scene geometrical properties such as the horizon line. The experiment is conducted on $\sim$ 100 images that include at least 3 cars in any single image from Label-Me dataset provided by *Hoiem et al.* (2006) [4]. The training images for our detector are extracted from Pascal 2007 cars training set (*Everingham et al.*, 2007). Fig. 7.7 (a) compares the detection performance of our full model at different iterations with *Hoiem et al.* (2006). Although both methods rely on different baseline detectors, similar to *Hoiem et al.* (2008), our method shows that geometric context

[4] As explained in *Bao et al.* (2010b) and in Sec. 7.1.1.2, at least 3 objects are necessary for estimating the layout.

provides high-level cues to iteratively improve detection performance. Notice that our algorithm: i) does not require the estimation of horizontal or vertical surfaces as it extracts spatial contextual information from the object itself (enabling our algorithm to work even if the ground plane is not visible at all); ii) it works even if objects are supported by multiple planes located at different heights with respect to the camera.

We further evaluate the object detection performance of our model with supporting region information provided by different methods (Fig. 7.7 (b)). The detection performance of our model with supporting region information provided by our segmenter (AP=27.6%) is comparable to the performance (AP=28.8%) of our model with the ideal supporting region information. We generate the ideal supporting regions by using the ground truth bounding boxes to remove mistaken supporting regions predicted by *Hoiem et al.* (2006). Our proposed segmenter is also flexible in that it can easily incorporate ground plane segmentation results provided by *Hoiem et al.* (2006) as an additional cue. This leads to the best detection performance AP= 28.4%. We further evaluate the performance of our 3D layout estimation algorithm by comparing the estimated vanishing lines (i.e, corresponding to the most confident plane estimated by our full algorithm) with the ground truth vanishing lines. At the first iteration, the relative L_1 error [5] is 6.6%. And at the final (5th) iteration, the relative L_1 error is 4.2% which is comparable to the 3.8% error of *Hoiem et al.* (2006). Typical examples are shown in the first three rows of Fig. 7.9. All results validate that the feedback loop is effective in improving i) object detection, ii) scene layout orientation estimation, iii) focal length estimation, in an outdoor environment. Moreover, our method is flexible enough to incorporate different cues such as the ground plane segmentation results provided by *Hoiem et al.* (2006).

[5] $e_H = \frac{1}{N}\sum_i |\frac{\widehat{H_i} - H_i}{H_i}|$, where $\widehat{H_i}$ and H_i are the best estimated and ground truth vanishing line.

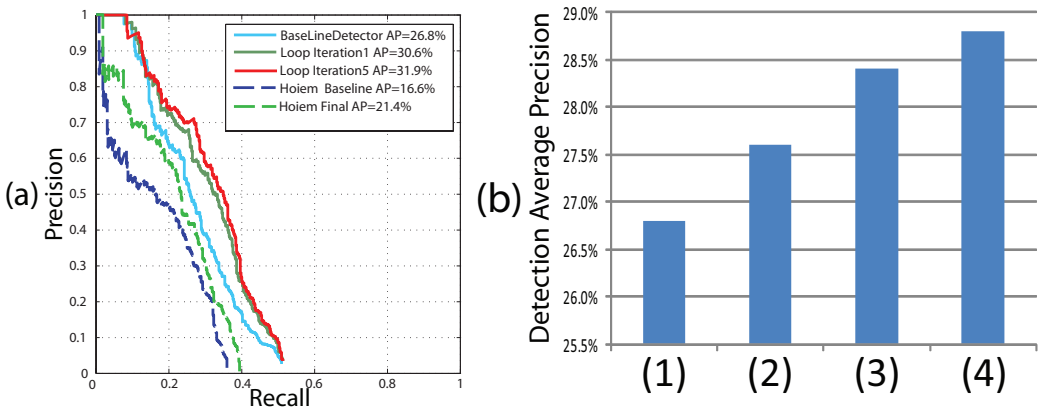


Figure 7.7: Detection performance on labelme (*Russell et al.*, 2008) dataset. Panel (a) shows the results after applying the full system from iteration 1 to 5. This figure also shows the results of *Hoiem et al.* (2006) and its baseline method (*Dalal and Triggs*, 2005). Panel (b) shows average detection precision (at the final iteration) using (1) the baseline detector (proposed in Chapter IV), (2) our supporting region segmenter module, (3) supporting regions provided by *Hoiem et al.* (2005a) as an additional cue to our region segmenter module, (4) ideal supporting regions provided by *Hoiem et al.* (2005a) where mistaken supporting regions are removed by using ground truth object bounding boxes.

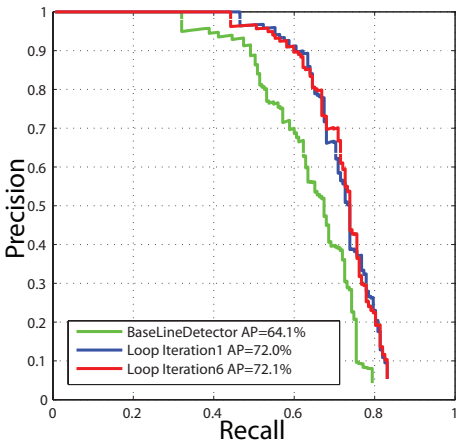


Figure 7.8: Detection performance using full system form iteration 1 to 6 on the office dataset (*Sudderth et al.*, 2008). The baseline detector proposed in Chapter IV.

7.2.3 Office Dataset

We use the office dataset (*Sudderth et al.*, 2008) for additional evaluation. 150 images are randomly selected for training and the remaining 54 images (which contain at least 3 objects of interest) are used for testing. Average overall detection performances for computer mice, monitors, and keyboards are shown in Fig. 7.8. Notice the improvement of almost 8%. We validate that our method generalizes well to another

indoor environment dataset. Typical examples are shown in Fig. 7.11.

7.3 Conclusion and Future Work

In this chapter, we have presented a framework for jointly detecting objects, estimating the scene layout and segmenting the supporting surfaces holding these objects. Our approach is built upon an iterative estimation procedure where the object detector becomes more and more accurate as evidence about the scene 3D layout and the object supporting regions becomes available and vice versa. Quantitative and qualitative experimental results on both indoor (dataset introduced in Chapter IV, *Sudderth et al.* (2008)) and outdoor (*Hoiem et al.*, 2006) datasets support our claims empirically.

As future work, we would like to develop an estimation procedure which guarantees convergence and global optimality. Moreover, since, in the current implementation, the model parameters are all learned separately, we plan to develop a learning algorithm which learns all the model parameters in a joint fashion. The limitation of at least three objects per image can be overcome if a prior is placed on the focal length and the support plane parameters. We plane to explore how such prior knowledge can help improving the overall object detection, layout estimation, and supporting region segmentation accuracy.

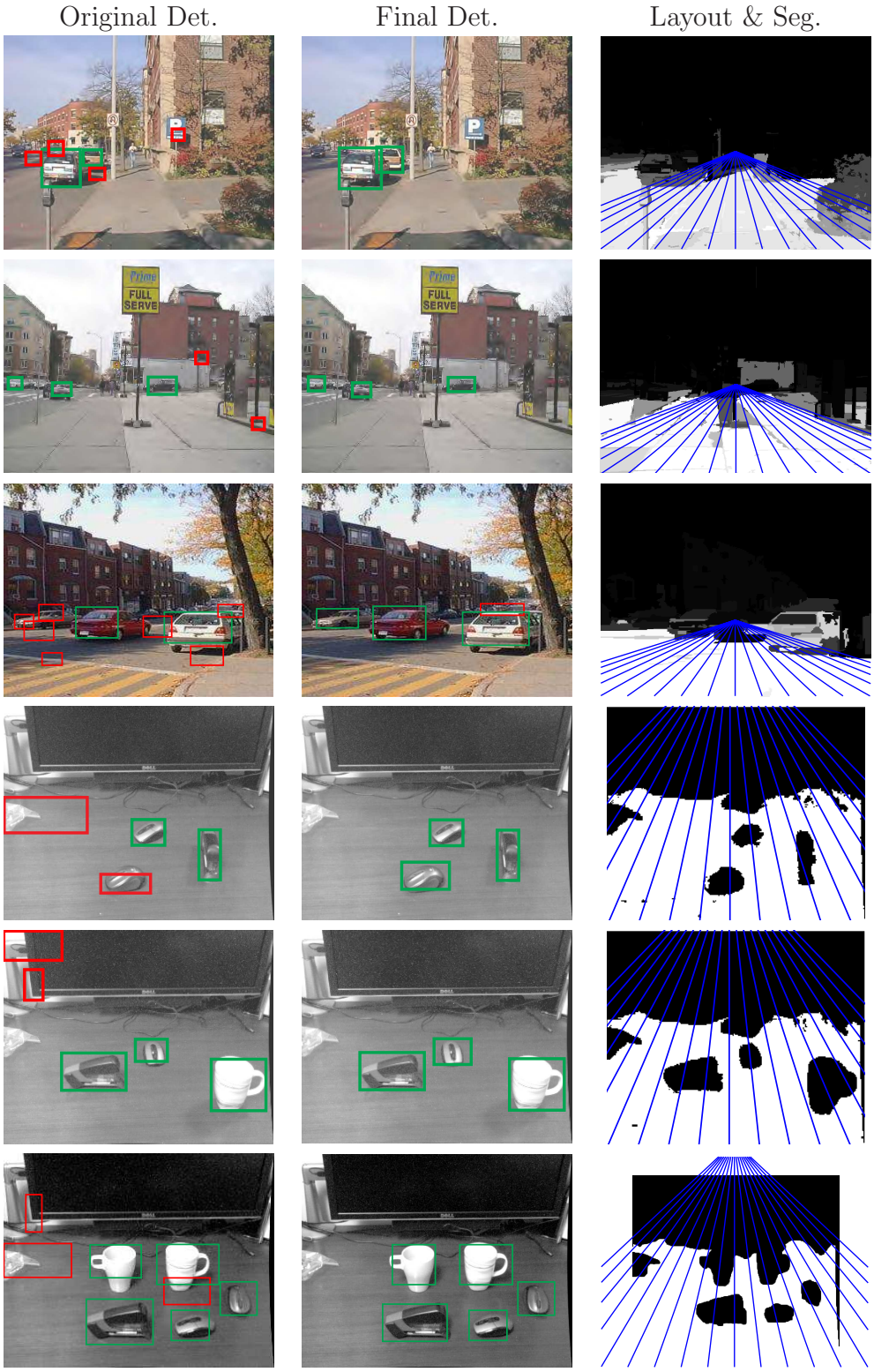


Figure 7.9: Typical results of joint object detection (green), layout estimation (blue), and original false detections (red). The supporting region is visualized by showing the confidence that a pixel belongs to a supporting region (white indicate high confidence). Results on labelme (*Hoiem et al.*, 2006), table-top (introduced in Chapter IV) datasets are shown in row $1 \sim 3$ and $4 \sim 6$, respectively. Notice that the modules jointly improve the original detection and enable convincing layout estimation and supporting region segmentation.

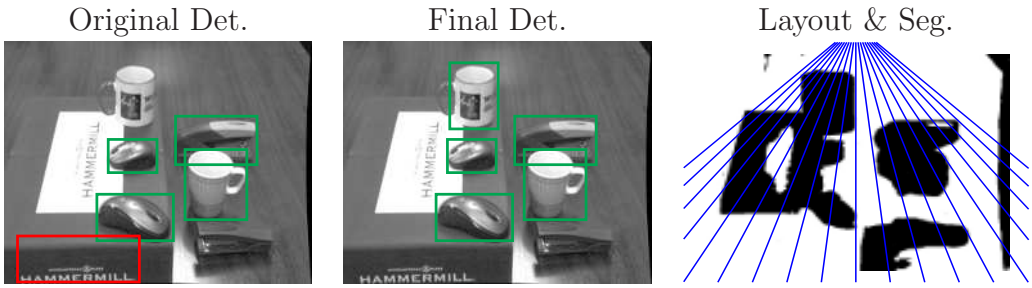


Figure 7.10: Typical results of joint object detection (green), layout estimation (blue), and original false detections (red) on images with 2 planes in table-top dataset (introduced in Chapter IV). The supporting region is visualized by showing the confidence that a pixel belongs to a supporting region (white indicate high confidence). Notice that we assume multiple planes must be parallel to each other. Therefore, multiple planes will share the same vanishing line.

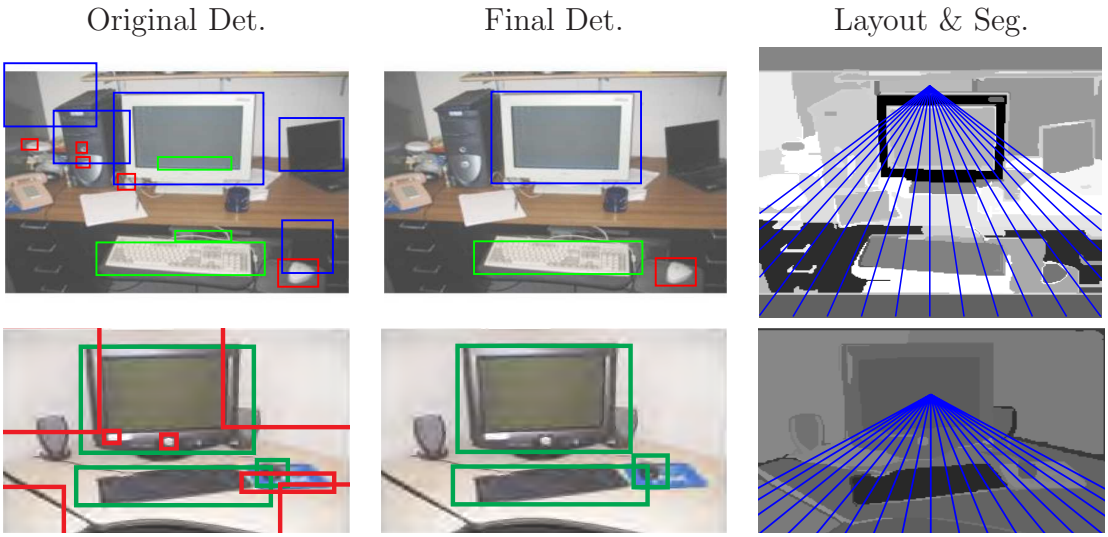


Figure 7.11: Typical results of joint object detection (green), layout estimation (blue), and original false detections (red) on office dataset (*Sudderth et al.*, 2008). The supporting region is visualized by showing the confidence that a pixel belongs to a supporting region (white indicate high confidence).

CHAPTER VIII

Models for Coherent Scene Understanding

The last decade has seen the development of a number of methods for object detection, segmentation and scene understanding. These methods can be divided into two broad categories: methods that attempt to model and detect objects that have distinct shape properties such as cars or humans, and methods that seek to model and identify scene elements whose internal structure and spatial support are more heterogeneous such as grass or sky. In the former case, we find that methods based on pictorial structures (i.e., *Felzenszwalb et al.* (2008)), pyramid structures (i.e., *Grauman and Darrell* (2005)), generalized Hough transform (*Leibe et al.*, 2004; *Gall and Lempitsky*, 2009; *Maji and Malik*, 2009; *Barinova et al.*, 2012; *Sun et al.*, 2010b), or multi-view model (*Sun et al.*, 2009; *Xiang and Savarese*, 2012) work best. These representations are appropriate for capturing shape or structural properties of objects, and typically identify the object by a bounding box. For the latter case, methods aiming at segmenting the image into semantically consistent regions (*He et al.*, 2004; *Kohli et al.*, 2008; *Shotton et al.*, 2008) work well for scene elements, like sky or road.

In order to coherently interpret the depicted scene, various types of contextual relationships among objects and scene elements have been explored. For example, co-occurrence relationships (e.g., cow and grass typically occur in the same image)

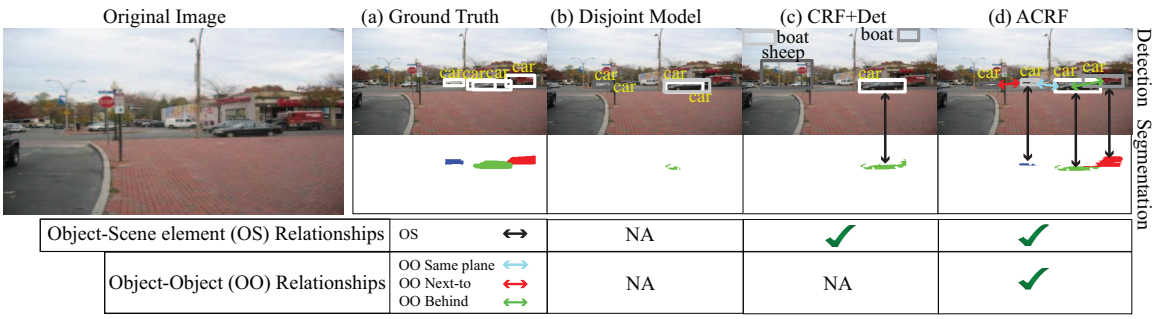


Figure 8.1: Our goal is to segment the image into objects (e.g., cars, humans, etc) and scene elements (e.g., road, sky, etc) by combining segmentation (bottom) with object detection (top). Results from different variants of our method (capturing a subset of critical contextual relationships) are shown from left to right columns. At the top of each column, we show the top 4 probable bounding boxes, where light and dark boxes denote the confidence ranking from high to low. Instance-based segmentation are shown in each bottom column, where different colors represent different object instances. Notice that our final ACRF captures the key relationships and recovers many missing detections and segmentation labels. Object-scene element and object-object relationships are indicated by color-coded arrows connecting pairs of objects/scene elements. Different color codes indicate different types of relationships.

have been captured in *Ladicky et al.* (2010b); *Rabinovich et al.* (2007), 2D geometric relationships (e.g., below, next-to, etc) have been utilized in *Gupta and Davis* (2008a); *Desai et al.* (2009); *Heitz and Koller* (2008), 2.5D geometry relationships (e.g., horizon line) have been incorporated by *Hoiem et al.* (2006) and *Bao et al.* (2010a). The use of such contextual relationships have inspired the development of robust algorithms for various recognition and scene understanding tasks. For instance, many segmentation methods (*Ladicky et al.*, 2010b; *Winn and Shotton*, 2006; *Gould et al.*, 2009a) have been proposed to capture scene element-scene element relationships in a random field formulation. Similarly, object-object relationships have been incorporated into a random field for jointly detecting multiple objects (*Desai et al.*, 2009).

Recently, researchers have proposed methods to jointly detect objects and segment scene elements. *Gould et al.* (2009b) propose a random field model incorporating both scene element-scene element, object-scene element, and object-horizon relationships. One limitation of their approach is that it cannot capture 2D geometric and co-

occurrence relationships between objects. Moreover, inference is computationally very demanding and typically takes around five minutes per image. To overcome this limitation, some authors have proposed inference procedures which leverage existing approaches for detection and segmentation and use the output of such approaches as input features in an iterative fashion (*Heitz et al.*, 2008; *Sun et al.*, 2010a; *Hoiem et al.*, 2008). Unfortunately, optimality is not guaranteed for most of these approaches.

We propose a novel framework for jointly detecting objects and segmenting scene elements that, for the first time, can coherently capture many known types of contextual relations between object-object, scene element-scene element, and object-scene element. Our contributions are three-fold. First, the model infers both geometric and semantic relationships describing the objects (i.e., object x is behind object y) via property interactions. Second, the model enables instance base segmentation (see color coded segments in Fig. 8.1(d)) by associating segments to object instance-specific labels (e.g., first car, second car, etc.). Finally, the special design of model potentials allows efficient inference which takes a few seconds per image in average and is performed using a combination of state-of-the-art discrete optimization techniques.

Hypothesis and property lists. Our framework extends the basic conditional random field (CRF) formulations for segmentation (i.e., recognizing scene elements) (*Shotton et al.*, 2006; *Kohli et al.*, 2008) by introducing the concept of hypotheses for objects and scene elements. Every hypothesis is described by a *property list* (Fig. 8.2-Top). This list includes semantic properties, such as the category label l. Further, if the category label belongs to objects such as car, human, etc., the hypothesis can also be characterized by some geometric properties, such as the 2D location (u, v), and the distance from the camera (depth) d.

We augment the above-mentioned CRF formulation with indicator variables which capture the presence or absence of hypotheses (see Fig. 8.2(a)-Top). We refer to our model as the augmented CRF, or ACRF, to highlight the newly added indicator vari-

ables. The indicator variables can take only two states: 0 or 1 which represents the absence or presence of an hypothesis, respectively. The key benefit of the indicator variables is that they allow us to easily encode sophisticated semantic and geometric relationships between pairs of hypotheses. For instance, simple pairwise potentials defined over indicator variables can allow to incorporate i) 2D geometric relationships such as "above" which model the property that one hypothesis lies above the other (e.g., a person sitting on a bike), ii) depth and occlusion relationships such as "in-front" which model the property that one hypothesis lies in front of the other (e.g., a person standing in front of a car), and iii) support relationships which model the property that one hypothesis is supported by another hypothesis (e.g., pedestrians walking on a road). More sophisticated relationships such as a composition of these basic 2D or 2.5D relationships can also be supported. Critically, the ACRF model generalizes Ladicky et al.'s model (*Ladicky et al.*, 2010b) which captures scene element-scene element co-occurrence contextual relationships only. In contrast, our model cannot only encode relationships between scene elements that depend on their geometrical properties such as (orientation, depth) but can also encode geometrical and semantic properties between scene elements and objects as well as objects and objects. Our model can also handle multiple instances of objects and, thus, also generalizes the work of *Barinova et al.* (2012). We illustrate the efficacy of our approach in Fig. 8.1. As seen in the figure, detections typically do not agree with the segmentation results (Fig. 8.1(b)) if the detection and segmentation are applied separately. A model capturing object-scene element relationships ensures consistency between detection and segmentation results (Fig. 8.1(c)). Finally, when object-object relationships (e.g., next-to, behind, same plane, etc) are included, even small objects, that are hard to detect and segment, can be discovered (Fig. 8.1(d)).

Learning. Relationships encoded by our model are regulated by a number of model parameters that we learn from training data. The relationships can be both attractive

(e.g., a person is likely to sit on a motorbike) and repulsive (e.g., car and airplane are unlikely to co-occur), and are enforced by adding positive or negative costs to the energy of the model. We formulate the problem of learning these costs as a Structured SVM (SSVM) (*Tsochantaridis et al.*, 2004) learning problem with two types of loss functions related to the segmentation loss and detection loss, respectively (see Sec. 8.3 for details).

MAP Inference. Jointly estimating the segmentation variables X and indicator variables Y (Fig. 8.2(c)) is challenging due to the intrinsic difference of the variable space and the complex types of pair-wise relationships between object-object, and object-scene element. We design an efficient graph-cut-based move making algorithm by combining state-of-the-art discrete optimization techniques. Our method is based on the α-expansion move making approach (*Boykov et al.*, 2001), which works by projecting the energy minimization problem of segmentation variables X into a binary energy minimization problem over a domain space of indicator variables Y. We use the "probing" approach similar to the one described by *Rother et al.* (2007) to handle the non-submodular function related to pair-wise object relationships (i.e., object-object). Our MAP inference algorithm takes only a few seconds per image in average as opposed to five minutes by *Gould et al.* (2009b).

Outline of the Chapter. The rest of the chapter is organized as follows. We first describe model representation, inference, learning, and implementation details in Sec. 8.1, 8.2, and 8.3, respectively. Experimental results are given in Sec. 8.4.

8.1 Augmented CRF

We now explain our Augmented Conditional Random Field (ACRF) model. ACRF jointly models object detection and segmentation (Fig. 8.2 (a)), and can incorporate contextual relationships between objects and scene element, and between multiple objects (Fig. 8.2 (b)).

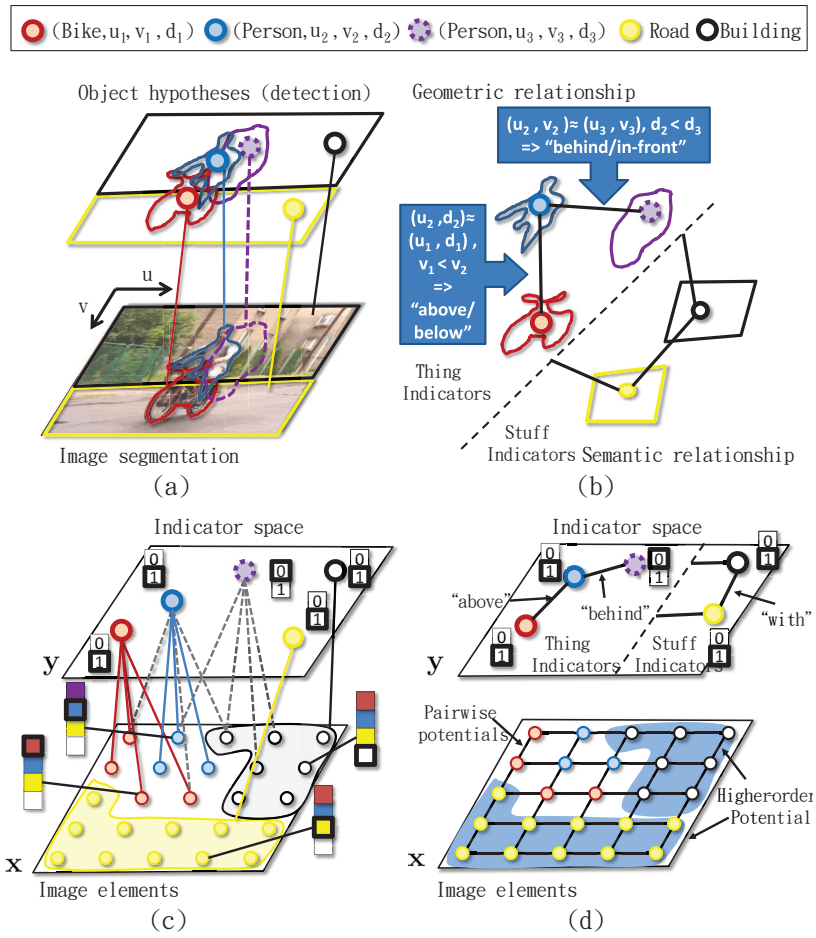


Figure 8.2: Our Augmented CRF model (ACRF). In panel (a), we show an image and the indicator variables corresponding to the different object hypotheses present in it. The instance hypotheses for object categories such as person and bike have geometric properties (e.g., spatial and depth). These properties are absent for scene element categories such as building and road. In panel (b), we demonstrate how relationships (e.g., behind, above, etc) are selected given a pairs of property lists. In panel (c), the figure shows the label space of the segmentation variables $\mathbf{X}$ and the indicator variables $\mathbf{Y}$ and the interaction between them. In panel (d)-Bottom, the figure shows the pairwise and higher order interactions among segmentation variables $\mathbf{X}$ which are present in standard CRF formulations. In panel (d)-Top, the figure shows the pairwise interactions among indicator variables $\mathbf{Y}$ which can encode different geometric and semantic relationships. The two edges connected to the scene element indicators which end in the dashed separator line indicate that scene element indicators are interacting with indicators with only category property.

Segmentation, like other image labeling problems, is commonly formulated using Conditional Random Fields (CRF). The conventional CRF model is defined over a set

of random variables $X = \{x_i\},\ i \in \mathcal{V}$ where $\mathcal{V}$ represents the set of image elements, which could be pixels, patches, super-pixels, etc (Fig. 8.2 (c)-Bottom). Each random variable x_i is assigned to a label from a discrete label space $\mathbf{L}$, which for the task of category segmentation, is considered the set $\mathcal{L}$ of categories including objects and scene elements such as cars and grass, respectively.

The energy (or cost) function $E(X)$ of the CRF is the negative logarithm of the joint posterior distribution of the model and has the following form: $E(X) = -\log P(X|\mathcal{E}) = -\log \phi_{eRF}(X|\mathcal{E}) + \mathrm{K} = \sum_{c \in \mathcal{C}^X} \psi_c(X_c) + \mathrm{K}$, where $\mathcal{E}$ is the given evidence from the image and any additional information (e.g., property lists), $\phi_{eRF}(X|\mathcal{E})$ takes the form of a higher order CRF model defined over image elements (Fig. 8.2 (d)-Bottom). $\phi_{eRF}(X|\mathcal{E})$ can be decomposed into potential ψ_c which is a cost function defined over a set of element variables X_c (called a clique) indexed by $c \in \mathcal{C}^X$; $\mathcal{C}^X$ is the set of cliques for image elements, and K is a constant related to the partition function. The problem of finding the most probable or maximum a posteriori (MAP) assignment of the CRF model is equivalent to solving the following discrete optimization problem: $X^* = \arg\min_{X \in \mathcal{L}^{|\mathcal{V}|}} E(X)$.

The standard CRF model mostly relies on bottom-up information. It is constructed using unary potentials based on local classifiers and smoothness potentials defined over pairs of neighboring pixels. Higher-order potentials such as the ones used in *Kohli et al.* (2008) encourage labels of groups of image elements to be the same. This classic representation for segmentation has led to excellent results for the scene elements, but has failed to replicate the same level of performance for objects. The reason for this dichotomy lies in the model's inability to explicitly encode the relationship between the shape and relative positions of different parts of structured objects such as the head and the torso of a person.

Part-based models such as Pictorial Structures (*Felzenszwalb and Huttenlocher*, 2005), Latent SVM (LSVM) (*Felzenszwalb et al.*, 2008), and Hough transform based

models (*Leibe et al.*, 2004; *Barinova et al.*, 2012) have shown to be much more effective at detecting objects. One of our contributions is proposing a unified framework to incorporate all instance hypotheses from these methods as additional object-instance evidences. In our model, every piece of object-instance evidence is characterized by the property lists. These properties include the category l_j, the spatial location in the image (u_j, v_j) at which the object is seen, and the depth or distance d_j of the object instance from the camera.

In addition to the variables representing image elements, our model contains a set of indicator variables (later referred as indicators) $Y = \{y_j \in \{0, 1\}\}$ for every possible configuration $j \in \hat{\mathcal{Q}}$ of a hypothesis (Fig. 8.2 (c)-Top). The configuration set $\hat{\mathcal{Q}}$ is a Cartesian product of the space of all possible category labels $\mathcal{L}$, all possible spatial locations $U \times V$ in the image, and all depth or distance values within a range $[0, D]$. For example, a configuration $j \in \hat{\mathcal{Q}}$ specifies that an instance of the category $l_j \in \mathcal{L}$ exists at location (u_j, v_j) in the image at a distance d_j away from the observer. We also associate with each instance a segmentation mask $\mathcal{V}_j$ which is the set of image elements associated with the hypothesis. In order to handle uncertainty in location and distance from the camera for scene elements (e.g., a grass region may have a large spatial extent and it may be at a range of distances from the camera) and objects which are not detected, we allow a configuration j including general hypothesis specified by l_j only (i.e., without specifying the location of the hypothesis).

As mentioned earlier, variables X representing the image elements in the classical CRF formulation for segmentation take values from the set of categories $\mathcal{L}$ only. In contrast, in our framework, these variables take values from a set of all possible configuration $x_i \in \mathbf{L} = \hat{\mathcal{Q}}$ (refer as *augmented labeling space*). On the one hand, this allows us to obtain segmentations of individual instances of particular categories which the classical CRF formulations are unable to handle. On the other hand, the space $\hat{\mathcal{Q}}$ of all possible detections is clearly huge, which makes learning and inference

much more challenging. We will come back to this issue later.

The joint posterior distribution of the segmentation X and indicator variables Y can be written as:$P(X,Y|\mathcal{E}) \propto \phi_{eRF}(X|\mathcal{E})\ \phi_{oRF}(Y|\mathcal{E})\ \phi_{con}(X,Y|\mathcal{E})$. The functions ϕ_{oRF} take the form of a CRF model defined over indicator variables as follows: $\phi_{oRF}(Y|\mathcal{E}) = \prod_{c\in\mathcal{C}^Y} e^{\varphi_c(Y_c)}$, where the potential $\varphi_c(Y_c)$ is a cost function defined over a set of indicator variables Y_c indexed by $c \in \mathcal{C}^Y$, and $\mathcal{C}^Y$ is the set of cliques of indicators. The potential function ϕ_{con} enforces that the segmentation and indicator variables take values which are consistent with each other (Fig. 8.2 (c)). The term is formally defined as: $\phi_{con}(X,Y|\mathcal{E}) = \prod_{j\in\hat{\mathcal{Q}}} e^{\Phi(y_j,X)}$, where $\Phi(y_j,X)$ is the potential relating each indicator y_j with a specific set of elements $\mathcal{V}_j$ in X. Hence, the model energy can be written as:

$$E(X,Y) = \sum_{c\in\mathcal{C}^X} \psi_c(X_c) + \sum_{j\in\hat{\mathcal{Q}}} \Phi(y_j,X) \tag{8.1}$$

$$+ \sum_{c\in\mathcal{C}^Y} \varphi_c(Y_c)\ . \tag{8.2}$$

The first term of the energy function is defined in a manner similar to *Kohli et al.* (2008). We now describe other terms of the energy function in detail in the following subsections.

Implicit representation of inactive configurations. It is easy to see that the space $\hat{\mathcal{Q}}$ of all possible configurations is huge, which would make learning and performing inference in the above model completely infeasible. However, in real world images, only a few possible configurations are actually present. Thus, most indicator variables $y_j,\ j \in \hat{\mathcal{Q}}$ are inactive (take value 0), and similarly the label set for the segmentation variables is typically quite small. We use an object detector that has been trained on achieving high recall rate to generate the set of *plausible* object configurations $\mathcal{Q}$ instances that are likely to be present in any given image. In this way, we reduce the problem into a manageable size for the inference algorithm.

8.1.1 Relating Y and X

The function $\Phi(y_j, X)$ (Fig. 8.2(c)) is a likelihood term that enforces consistency in the assignments of the jth indicator variable y_j and a set of segmentation variables X. It is formally defined as:

$$\Phi(y_j, X) = \begin{cases} \inf & \text{if } y_j \neq \delta_j(X) \\ \gamma_{l_j} \cdot |\mathcal{V}_j| & \text{if } y_j = \delta_j(X) = 1 \\ 0 & \text{if } y_j = \delta_j(X) = 0 \end{cases}, \tag{8.3}$$

where j is any possible configuration in $\mathcal{Q}$, the function $\delta_j(X)$ indicates whether the indicator j shares a consistent category label with image elements in $\mathcal{V}_j$, and is defined as:

$$\delta_j(X) = \begin{cases} 1 & \text{if } R_j(X) = \frac{|\mathcal{V}_j(X)|}{|\mathcal{V}_j|} \geq R(l_j) \\ 0 & \text{otherwise} \end{cases}, \tag{8.4}$$

where $|\mathcal{V}_j(X)| = |\{i; x_i = l_j \texttt{ for } i \in \mathcal{V}_j\}|$ is the number of elements in $\mathcal{V}_j$ assigned with label l_j, $|\mathcal{V}_j|$ is the total number of elements in $\mathcal{V}_j$, $R_j(X)$ is the consistency percentage, and $R(l_j) \in [0\ 1]$ is a category-specific consistency threshold. Hence, the first condition in the above function ensures that $y_j = 1$ if and only if the detection j shares a label with at least $R(l_j)$ percent of the pixels (or image element) in $\mathcal{V}_j$ (i.e. $R_j(X) \geq R(l_j)$). The remaining conditions in Eq. 8.3 shows that this potential is an Occam razor or MDL prior, similar to *Ladicky et al.* (2010b); *Barinova et al.* (2012) so that the model is penalized by $\gamma_{l_j} \cdot |\mathcal{V}_j|$ when $y_j = 1$.

8.1.2 Indicator CRF

The indicator CRF potential $\varphi_c(Y_c)$ in Eq. 8.2 can be decomposed into two terms as follows, $\sum_{c \in \mathcal{C}^Y} \varphi_c(Y_c) = \sum_{j \in \mathcal{Q}_1} \varphi_u(y_j) + \sum_{(j,k) \in \mathcal{U}} \varphi_p(y_j, y_k)$, where $\mathcal{Q}_1 \subset \mathcal{Q}$ is the set

of object indicators with geometric properties and $\mathcal{U}$ is the set of pairs of indicators, which interact with each other.

The term $\varphi_u(y_j)$ is the unary potential for the object indicator, defined as:

$$\varphi_u(y_j) = \begin{cases} \beta_j \cdot |\mathcal{V}_j|, & \texttt{if } y_j = 0 \\ 0, & \texttt{if } y_j = 1 \end{cases} , \tag{8.5}$$

such that the cost of suppressing hypothesis j (i.e., label y_j as 0) is $\beta_j \cdot |\mathcal{V}_j|$ (proportional to the detection confidence).

The term $\varphi_p(y_j, y_k)$ (black edges in Fig. 8.2 (d)-Top) represents the interactions between any pair of indicator variables. Depending on the types of properties associated with the pair of indicator variables, this term can represent a number of relationships. It can not only model spatial relationship in 2D such as the ones learned and employed in the approach proposed by *Desai et al.* (2009), but also model behind and in-front relationships given the depth property. The term can also encode co-occurrence relationships (*Ladicky et al.*, 2010b) for pairs of indicators with only category properties.

For any pair of indicators $j, k \in \mathcal{Q}$, the term is formally defined as:

$$\varphi_p(y_j, y_k) = w^{r_{jk}}_{l_j,l_k}(y_j, y_k) \cdot \max(|\mathcal{V}_j|, |\mathcal{V}_k|) , \tag{8.6}$$

where r_{jk} is the type of relationship that we want to enforce between the pair of object instances j and k, and is a subset of the overall relationship set $\mathcal{R}$, which is defined as: $\mathcal{R} = \{$*co-occur, above, below, next-to, in-front, behind, or the composition of them*$\}$.

The pseudo-boolean function $w^{r_{jk}}_{l_j,l_k}(y_j, y_k) : \{0, 1\}^2 \rightarrow \mathbb{R}$ specifies the cost of all 4 possible joint assignments of y_j and y_k under the relationship r_{jk} for a pair of object categories l_j, l_k. As a result, the potential can capture both attractive (i.e.,

$w(0,0)+w(1,1) \leq w(0,1)+w(1,0)$) and repulsive interactions (i.e., $w(0,0)+w(1,1) \geq w(0,1)+w(1,0)$). For example, a person usually is sitting on a motorbike (attractive), and cars do not overlap with each other in 3D (repulsive).

The relationship r_{jk} is specified by the properties associated with the detection instances j and k. For instance, if the indicators i and j have only category properties, the relationship r_{jk} models the co-occurrence cost of the categories. In this case, we assume

$$w^{co}_{j,k}(y_j, y_k) = \begin{cases} \gamma_{l_j,l_k} & \text{if } y_j = y_k = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{8.7}$$

where γ_{l_j,l_k} is the co-occurrence cost for categories l_j and l_k. From the above definition, it is easy to see that our model generalizes both CRF models proposed in *Ladicky et al.* (2010b); *Desai et al.* (2009).

8.2 Inference

We now show that the MAP inference problem in our ACRF model can be solved by minimizing the energy function using an efficient graph cut based expansion move making algorithm (*Boykov et al.*, 2001).

Standard move making algorithms repeatedly project the energy minimization problem into a smaller subspace in which a sub-problem is efficiently solvable. Solving this sub-problem produces a change to the solution (referred to as a move) which results in a solution having lower or equal energy. The *optimal* move leads to the largest possible decrease in the energy.

The *expansion* move algorithm projects the problem into a Boolean label sub-problem. In an α-expansion move, every segmentation variable X can either retain its current label or transit to the label α. One iteration of the algorithm involves making

moves for all α in $\mathcal{L}$ successively. Under the assumption that the projection of the energy is pairwise and submodular, it can be exactly solved using graph cuts (*Boros and Hammer*, 2002; *Kolmogorov and Zabih*, 2004). We derive graph construction only for energy terms related to indicator variables Y, for all other terms, the constructions are introduced in *Kohli et al.* (2008); *Boykov et al.* (2001).

8.2.1 Functions of indicator variables Y with only category property.

The energy terms related to the indicator variables, whose only property is a category label, are $\Phi(y_j, X)$ in Eq. 8.3 and $\varphi_p(y_j, y_k)$ in Eq. 8.6 and 8.7. Observe that we can represent the combination of these terms as a function, $\mathcal{F} : L \subset \mathcal{L} \rightarrow \mathbb{R}$ as

$$\mathcal{F}(L(Y)) = \min_{X} \sum_{j \in \mathcal{Q}_2} \Phi(y_j, X) + \sum_{(j,k) \in \mathcal{U}_2} \varphi_p(y_j, y_k) \ , \tag{8.8}$$

where $L(Y) = \{l_j; k \in \mathcal{Q}_2, y_j = 1\}$ is a set of existing categories (i.e., $y_j = 1$), $\mathcal{Q}_2$ is any subset of the indicator variables, whose only property is a category label, and $\mathcal{U}_2$ is a subset of $\mathcal{U}$ such that $j, k \in \mathcal{Q}_2$. From the definition of the term in section 8.1.1 and 8.1.2, we can see that $\mathcal{F}(\{l_j\}) = \gamma_{l_j}|\mathcal{V}_j|$. Furthermore, $\mathcal{F}(\{l_j, l_k\}) = \mathcal{F}(\{l_j\}) + \gamma_{l_k}|\mathcal{V}_k| + \gamma_{l_j,l_k}$; $\mathcal{F}(\{l_j, l_k, l_q\}) = \mathcal{F}(\{l_j, l_k\}) + \gamma_{l_p}|\mathcal{V}_p| + \gamma_{l_j,l_p} + \gamma_{l_k,l_q}$. It can be easily seen that the above function satisfies the properties of the co-occurrence potential:

$$L_1 \subset L_2 \Longrightarrow \mathcal{F}(L_1) \leq \mathcal{F}(L_2) \ , \tag{8.9}$$

proposed by *Ladicky et al.* (2010b) allowing us to use their graph construction for minimizing this energy function.

8.2.2 Functions of indicator variables Y with instance properties.

The energy terms related to the instance indicator variables are $\Phi(y_j, X)$ in Eq. 8.3 and $\varphi_p(y_j, y_k)$ in Eq. 8.6. Since $\varphi_p(y_j, y_k)$ in Eq. 8.6 captures both repulsive and at-

tractive pair-wise relationships, it can not be combined with $\Phi(y_j, X)$ in Eq. 8.3 to form a co-occurrence potential satisfying Eq. 8.8. However, $\Phi(y_j, X)$ can be approximated as:

$$\Phi(y_j, X) = \gamma_j(y_j \frac{1 - R_j(X)}{1 - R(l_j)} + (1 - y_j)\frac{R_j(X)}{R(l_j)}) \ . \tag{8.10}$$

The detailed derivation and the corresponding α-expansion move energy for this term is described in the Sec. 8.2.3.

The graph construction of the pair-wise instance indicators in Eq. 8.6 is equivalent to the construction of binary variables which is clearly described in *Boykov et al.* (2001). However, since we would like to capture both attractive (i.e., both indicators having the same labels) and repulsive (i.e., both indicators having different labels) interactions, some functions could be submodular and some could be non-submodular in (y_j, y_k), respectively. Since each indicator only interacts with other nearby indicators, a simple "probing" approach similar to the one described in *Rother et al.* (2007) can effectively handle the non-submodular function related to pair-wise object-instance interaction. As a result, our inference algorithm does not rely on sophisticated techniques such as QPBO (*Rother et al.*, 2007) which requires more memory and computation time.

8.2.3 Functions of Indicators Y

We observe in Eq. 8.3, when $y_j = 1$

$$\Phi(y_j, X) = \begin{cases} \inf & \text{if } \delta_j(X) = 0 \\ \gamma_j & \text{if } \delta_j(X) = 1 \end{cases} \approx \gamma_j \frac{1 - R_j(X)}{1 - R(l_j)} \ . \tag{8.11}$$

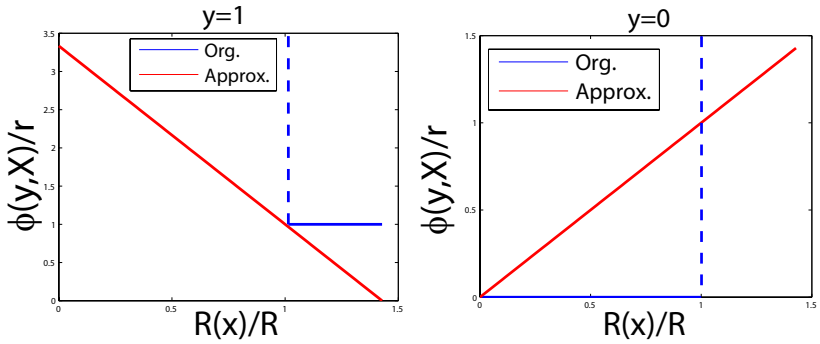


Figure 8.3: Comparison between the original function $\Phi(y, X)$ (blue line) and the approximated function (red lines) in Eq. 8.12 and 8.11. The left panel shows the case when $y = 1$. The right panel shows the case when $y = 0$. Notice the dash blue lines indicate the sharp transition from finite values to infinite values.

When $y_j = 0$

$$\Phi(y_j, X) = \begin{cases} \inf & \text{if } \delta_j(X) = 1 \\ 0 & \text{if } \delta_j(X) = 0 \end{cases} \approx \gamma_j \frac{R_j(X)}{R(l_j)} \ . \tag{8.12}$$

Hence, $\Phi(y_j, X)$ becomes,

$$\Phi(y_j, X) = \gamma_j (y_j \frac{1 - R_j(X)}{1 - R(l_j)} + (1 - y_j) \frac{R_j(X)}{R(l_j)}) \ . \tag{8.13}$$

The effect of the approximation in Eq. 8.12 and 8.11 are shown in Fig. 8.3. Instead of imposing an infinite cost when $\delta(X) \neq y$, our approximation imposes an cost which is linearly proportional to the consistency percentage $R(X)$. When $y = 1$, the ratio between the consistency percentage and the consistency threshold $R(X)/R(l)$ are encouraged to be large, which means the more elements in X are labeled as l, the better (Fig. 8.3-Left). On the contrary, when $y = 0$, the ratio between the consistency percentage and the consistency threshold $R(X)/R(l)$ is encouraged to be small, which means the less elements in X are labeled as l, the better (Fig. 8.3-Right).

8.2.3.1 α-expansion move energy

We first define the transformation function $T_\alpha(x_i; t_i)$ for the α-expansion move which transforms the label of a random variable x_i as:

$$T_\alpha(x_i, t_i) = \begin{cases} \alpha, & \text{if } t_i = 0 \\ x_i, & \text{if } t_i = 1 \end{cases} \tag{8.14}$$

The corresponding α-expansion move energy for the term in Eq. 8.13 can be written as: $\Phi(y_j, T_\alpha(X, T)) =$

$$\begin{cases} \gamma_j(\frac{y_j}{1-R(l_j)}(1 - R_j(X) + \sum_{i \in \mathcal{V}_j(X)} \frac{(1-t_i)}{|\mathcal{V}_j|}) \\ +\frac{1-y_j}{R(l_j)}(\sum_{i \in \mathcal{V}_j(X)} \frac{(t_i)}{|\mathcal{V}_j|})), \text{if } \alpha \neq l_j \\ \gamma_j(\frac{1-y_j}{R(l_j)}(R_j(X) + \sum_{i \in \mathcal{V}_j \setminus \mathcal{V}_j(X)} \frac{(1-t_i)}{|\mathcal{V}_j|}) \\ +\frac{y_j}{1-R(l_j)}(\sum_{i \in \mathcal{V}_j \setminus \mathcal{V}_j(X)} \frac{(t_i)}{|\mathcal{V}_j|})), \text{if } \alpha = l_j \end{cases} \tag{8.15}$$

where $\mathcal{V}_j \setminus \mathcal{V}_j(X)$ is the remaining set of elements in $\mathcal{V}_j$ with labels (i.e.,$\{x_i \neq l_j; i \in \mathcal{V}_j\}$). Notice that when $\alpha \neq l_j$ the function is submodular in (y_j, t_i), but when $\alpha = l_j$ it is submodular in $(\overline{y}_j, t_i)$, where $\overline{y}_j = 1 - y_j$ is the negation of y_j.

After the transformation, the first two terms of the original model energy (Eq. 8.2) becomes a pairwise and submodular function of T, Y, and $\overline{Y}$ as follows,

$$E(T, Y, \overline{Y}) = \sum_{c \in \mathcal{C}^X} \psi_c(T_c) + \sum_{j \in \hat{\mathcal{Q}}_1} \Phi(y_j, T) + \sum_{j \in \hat{\mathcal{Q}}_2} \Phi(\overline{y}_j, T) \ .$$

where $\hat{\mathcal{Q}}_1 = \{y_j; l_j \neq \alpha\}$ and $\hat{\mathcal{Q}}_1 = \{y_j; l_j = \alpha\}$. Therefore, we will construct the graph using T, partially using indicator y_j, and partially using the negation of indicator $\overline{y}_j$ depending on whether $l_j = \alpha$.

8.3 Learning

The full CRF model in Eq. 8.2 contains several terms. In order to balance the importance of different terms, we introduce a set of linear weights for each term as follows,

$$
\begin{aligned}
W^T\Psi(X.Y) &= \sum_{c\in\mathcal{C}} w_c\psi_c(X_c) + \sum_{(j,k)\in\mathcal{U}_1} w_{l_j,l_k}^{r_{jk}}(y_j,y_k) && (8.16)\\
&+ \sum_{j\in\mathcal{Q}_1} w^u(l_j)(\Phi(y_j,X)+\varphi_u(y_j)) \\
&+ w^{co}(\sum_{(j,k)\in\mathcal{U}_2}\varphi_p(y_j,y_k) + \sum_{j\in\mathcal{Q}_2}\Phi(y_j,X)) \ , && (8.17)
\end{aligned}
$$

where w_c models weights for unary, pair-wise, and higher-order terms in X. $w^u(l)$ is the category specific weight for unary term in y, w^{co} is the weight for image element (X)- indicators (Y), and $w_{l_j,l_k}^{r_{jk}}$ is the pair-wise weights for a specific co-occurrence type r_{jk} related to the pair of categories l_j, l_k in Eq. 8.6. Recall from Sec. 8.1.2 and 8.2 that $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are the set of indicator variables for objects and scene elements respectively. Similarly, $\mathcal{U}_1$ and $\mathcal{U}_2$ are the subset of $\mathcal{U}$ such that $j,k\in\mathcal{Q}_1$ or $j,k\in\mathcal{Q}_2$ respectively. Since all these weights are linearly related to the energy function, we formulate the problem of jointly training these weights as a Structured SVM (SSVM) learning problem (*Tsochantaridis et al.*, 2004) similar to *Desai et al.* (2009).

Assume that a set of example images, ground truth segment category labels, and ground truth object bounding boxes $\{I^n, X^n, Y^n\}_{n=1,\ldots,N}$ are given. The SSVM problem is as follows,

$$
\begin{aligned}
\min_{W,\xi\geq 0} \quad & W^TW + C\sum_n \xi^n(X,Y) && (8.18)\\
\text{s.t.} \quad & && (8.19)\\
\xi^n(X,Y) \quad & = \max_{X,Y}(\triangle(X,Y;X^n,Y^n) && (8.20)\\
& \quad + W^T\Psi(X^n,Y^n) - W^T\Psi(X,Y)), \forall n \ ,
\end{aligned}
$$

where W concatenates all the model parameters which are linearly related to the potentials $\Psi(X,Y)$; C controls the relative weight of the sum of the violated terms $\{\xi^n(X,Y)\}$ with respect to the regularization term; $\triangle(X,Y;X^n,Y^n)$ is the loss function that generates large loss when the X or Y is very different from X^n or Y^n. Depending on the performance evaluation metric, we design different loss functions as described in the Sec. 8.3.1

Following the SSVM formulation, we propose to use a stochastic subgradient descent method to solve Eq. 8.18. The subgradient of $\partial_W \xi^n(X,Y)$ can be calculated as $\Psi(X^n,Y^n) - \Psi(X^*,Y^*)$, where $(X^*,Y^*) = \arg\min_{X,Y}(W^T\Psi(X,Y) - \triangle(X,Y;X^n,Y^n))$. When the loss function can be decomposed into a sum of local losses on individual segments and individual detections, (X^*,Y^*) can be solved using graph-cut similar to the inference problem (Sec. 8.2). For other complicated loss functions, we found that it is effective to set (X^*,Y^*) approximately as $\arg\min_{X,Y} W^T\Psi(X,Y)$, when the loss is bigger than a threshold.

The remaining model parameters are set as follows. The category-specific $R(l)$ in Eq. 8.3 are estimated using the median values observed in training data. The γ involved in Eq. 8.8 are estimated from the MSE as described in *Ladicky et al.* (2010b). The β in Eq. 8.5 are set to be the detection confidence.

8.3.1 Loss Function

For the experiment on Stanford dataset, since the performance is measured by the average classification accuracy across different categories, we define the following loss function. The overall loss function $\triangle(X,Y;X^n,Y^n)$ is decomposed into sum of the segmentation loss $\triangle(X;X^n)$ and the detection loss $\triangle(Y;Y^n)$.

The segmentation loss $\triangle(X;X^n)$ is defined as

$$\triangle(X;X^n) = \frac{1}{Q}\sum_{i\in\mathcal{V}} \mathbf{1}\{x_i \neq x_i^n\} c_x(l_i) \ , \tag{8.21}$$

where $\mathcal{V}$ captures the indices for the set of segments, $\mathbf{1}\{STATEMENT\}$ is 1 if the $STATEMENT$ is true, $c_x(l_i)$ is the category l_i dependent cost (used to re-weight the loss contributed from different categories), and $Q = \sum_{i \in \mathcal{V}} c_x(l_i)$. Therefore, the overall segmentation loss can be decomposed into a sum over local loss for each segment $\frac{1}{Q}\mathbf{1}\{x_i \neq x_i^n\}c_x(l_i)$.

The detection loss $\triangle(Y; Y^n)$ is defined as

$$\triangle(Y; Y^n) = \frac{1}{M} \sum_{i \in \mathcal{B}} \mathbf{1}\{y_i \neq y_i^n\} c_y(l_i) \ , \tag{8.22}$$

where $\mathcal{B}$ captures the indices for the set of detections, $M = \sum_{i \in \mathcal{B}} c_y(l_i)$. Similarly, the overall detection loss can be decomposed into a sum over local loss for each detection $\frac{1}{M}\mathbf{1}\{y_i \neq y_i^n\}c_y(l_i)$.

For the experiment on PASCAL dataset, since the segmentation performance is measured by $\frac{\text{true positive}}{\text{true positive+ false positive + false negative}}$, the overall loss function $\triangle(X, Y; X^n, Y^n)$ is decomposed into sum of the segmentation loss $\triangle(X; X^n)$ and the detection loss $\triangle(Y; Y^n)$. The segmentation loss is 1-segmentation performance and the detection loss is the same as before. Since the segmentation loss cannot be decomposed into a per segment loss, we obtain (X^*, Y^*) approximately as $\arg\min_{X,Y} W^T \Psi(X, Y)$, when $\triangle(X^*, Y^*; X^n, Y^n)$ is bigger than a threshold.

8.4 Experiments

We compare our full ACRF model with *Gould et al.* (2009a); *Tighe and Lazebnik* (2010); *Munoz et al.* (2010); *Ladicky et al.* (2010a,b) on Stanford Background (referred as to Stanford) dataset (*Gould et al.*, 2009a) as well as with several state-of-the-art techniques on PASCAL VOC 2009 (referred as to PASCAL) dataset (*Everingham et al.*, 2009). As opposed to other datasets, such as MSRC (*Shotton et al.*, 2006), the Stanford dataset contains more cluttered scenes and more object instances per image.

(a) Global Accuracy					
Tighe and Lazebnik	Gould et al.	Munoz et al.	*Ladicky et al.* (2010a)	*Ladicky et al.* (2010b)	ACRF
77.5	76.4	76.9	80.2	80.0	**82.4**

(b)	Back-ground	Car	Person	Motor-bike	Bus	Boat	Cow	Sheep	Bi-cycle	Global	Avg
CRF	77.4	49.1	39.9	15.3	76.3	18.9	65.0	**70.4**	17.3	79.9	47.7
C+D	77.1	56.7	**61.7**	9.3	69.7	**36.9**	88.1	62.8	64.2	82.0	58.5
ACRF	77.2	**74.9**	60.1	**17.2**	**79.4**	**36.9**	**88.6**	58.2	**64.7**	**82.4**	**61.9**

Table 8.1: Segmentation performance comparison on the Stanford dataset. (a) Global segmentation accuracy of our ACRF model compared with state-of-the-art methods, where "Global" is the overall percentage of pixels correctly classified. (b) System analysis of our model. The CRF row shows the results by using only the scene element-scene element relationship component (first term in Eq. 8.2) of our ACRF model. The C+D row shows results by adding independent detections indicators to the CRF model (first two terms in Eq. 8.2). The last row shows results of the full ACRF model. Notice "Avg." is the average of the percentage over eight foreground classes and one background class.

Hence, segmenting and detecting "objects" is particularly challenging. Conversely, the PASCAL dataset contains larger number of "objects" labels with a single "scene element" label, with limited number of object instances in each image.

For all the experiments below, we use the same pre-trained LSVM detectors (*Felzenszwalb et al.*, 2008) to obtain a set of object-instance hypotheses for categories such as car, person, and bike. The object depths are inferred by combining both cues from the size and the bottom positions of the object bounding boxes similar to *Hoiem et al.* (2006); *Bao et al.* (2010a). The responses from off-the-shelf scene element classifiers are used as the unary scene element potentials in our model. We model different types of pair-wise scene elements relationships using a codebook representation similar to *Bosch et al.* (2010).

The following geometric pair-wise object relationships are used for the experiments to incorporates geometric spatial relationship between two bounding boxes: *next-to*, *above*, *below*, *in-front*, *behind*. On top of that, we have one additional geometric relationship based on horizon lines agreement between two detections.

Geometric spatial relationships are determined by following steps. To establish the geometric relationship given a pair of detection bounding boxes, we firstly set

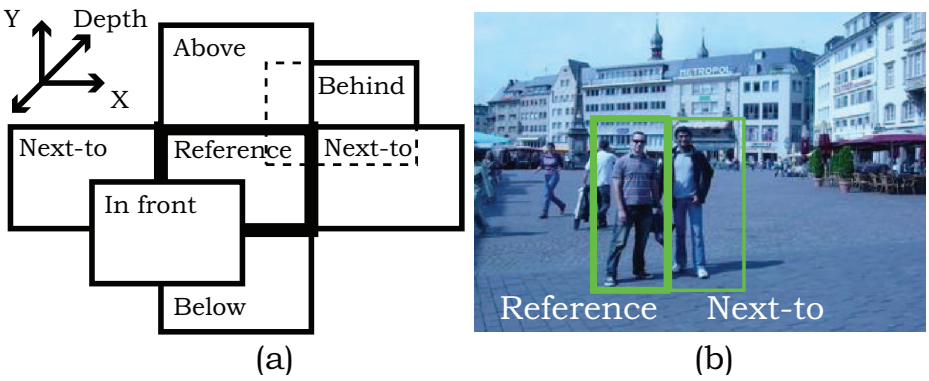


Figure 8.4: (a) Pairwise objects relationships can be determined by drawing an additional box with respect to a reference box. (b) In this example, a person (right) is 'next-to' a person on the left side.

	Background	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining table	Dog	Horse	Motor bike	Person	Potted plant	Sheep	Sofa	Train	TV/monitor	Average
CRF	69.0	19.7	2.4	8.8	6.8	8.8	21.6	17.5	13.1	0.5	8.6	7.1	6.5	6.5	13.8	19.2	5.8	13.6	3.3	21.3	9.5	13.5
CRF+Det	71.0	21.8	**14.8**	21.1	**18.7**	**34.8**	**48.3**	33.3	16.7	**12.3**	**27.5**	10.5	14.1	27.5	35.4	31.2	29.8	**28.7**	17.5	31.8	27.7	27.3
ACRF	**75.5**	**29.1**	14.7	**23.0**	18.2	34.0	47.8	**40.4**	**17.2**	11.4	27.0	**12.6**	**17.5**	**30.1**	**40.1**	**34.9**	**30.6**	28.2	**20.7**	**31.0**	**30.1**	**29.4**

Table 8.2: The segmentation accuracy of different variants of our model (i.e., CRF, CRF+Det (denotes CRF+Detection), and full ACRF models) on PASCAL dataset.

one of them as a reference box. Then, we draw an additional box with respect to a reference bounding box for a certain spatial relationship (i.e. *above*: draw on top of a box; *next-to*: draw on left or right side of a box, etc. See Fig. 8.4 for details). If a drawn box overlaps more than 50% with the other detection bounding box which is not selected as a reference box, we can specify a relationship to the given pair of boxes. This procedure is repeated for all geometric relationships.

The additional geometric spatial relationship is based on whether two horizon lines from two bounding boxes are in agreement. Specifically, horizon lines for two boxes are estimated assuming objects' average heights, similar to *Hoiem et al.* (2006). If two lines are close to each other within a certain range, which is a function of the specific class (i.e. person or car have smaller variance, boat have a larger variance), they have the same horizon line.

Stanford dataset. The Stanford dataset (*Gould et al.*, 2009a) contains 715 images

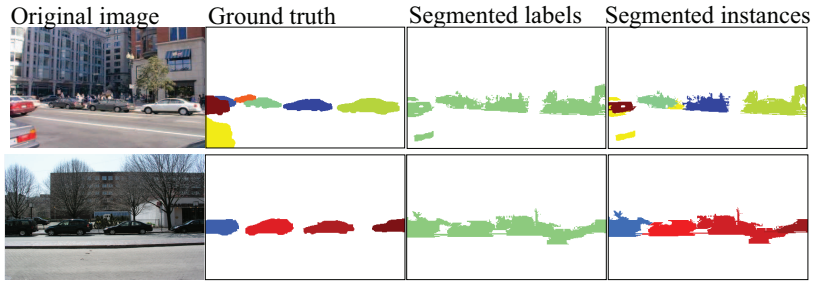


Figure 8.5: Typical segmentation results on the Stanford dataset. Notice that our model can obtain instance-based segmentations (last column) due to the ability to reason in the augmented labeling space $\hat{\mathcal{Q}}$.

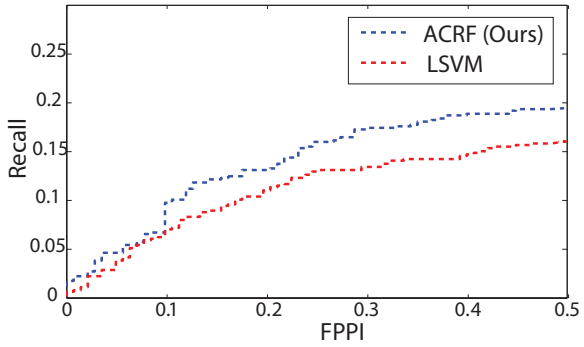
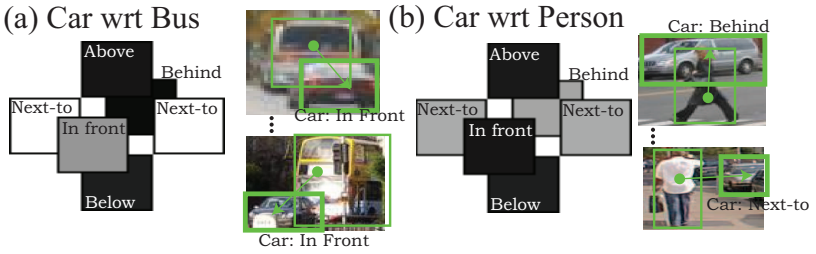


Figure 8.6: Recall v.s. FPPI curves of our ACRF and LSVM on Stanford dataset. Our ACRF achieves better recall at different FPPI values.

	Background	Aeroplane	Bicycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow	Dining table	Dog	Horse	Motor bike	Person	Potted plant	Sheep	Sofa	Train	TV/monitor	Average
BONN	83.9	64.3	21.8	21.7	32.0	40.2	57.3	49.4	38.8	5.2	28.5	22.0	19.6	33.6	45.5	33.6	27.3	40.4	18.1	33.6	46.1	36.3
CVC	80.2	67.1	26.6	30.3	31.6	30.0	44.5	41.6	25.2	5.9	27.8	11.0	23.1	40.5	53.2	32.0	22.2	37.4	23.6	40.3	30.2	34.5
NECUIUC	81.8	41.9	23.1	22.4	22.0	27.8	43.2	51.8	25.9	4.5	18.5	18.0	23.5	26.9	36.6	34.8	8.8	28.3	14.0	35.5	34.7	29.7
ACRF	75.5	29.1	14.7	23.0	18.2	34.0	47.8	40.4	17.2	11.4	27.0	12.6	17.5	30.1	40.1	34.9	30.6	28.2	20.7	31.0	30.1	29.4
UoCTTI	78.9	35.3	22.5	19.1	23.5	36.2	41.2	50.1	11.7	8.9	28.5	1.4	5.9	24.0	35.3	33.4	35.1	27.7	14.2	34.1	41.8	29.0
NECUIUC	81.5	39.3	20.9	22.6	21.7	26.1	37.1	51.5	25.2	5.7	17.5	15.7	24.2	27.4	35.3	33.0	7.9	23.4	12.5	32.1	33.3	28.3
LEAR	79.1	44.6	15.5	20.5	13.3	28.8	29.3	35.8	25.4	4.4	20.3	1.3	16.4	28.2	30.0	24.5	12.2	31.5	18.3	28.8	31.9	25.7
BROOKES	79.6	48.3	6.7	19.1	10.0	16.6	32.7	38.1	25.3	5.5	9.4	25.1	13.3	12.3	35.5	20.7	13.4	17.1	18.4	37.5	36.4	24.8
UCI	80.7	38.3	30.9	3.4	4.4	31.7	45.5	47.3	10.4	4.8	14.3	8.8	6.1	21.5	25.0	38.9	14.8	14.4	3.0	29.1	45.5	24.7
MPI	70.9	16.4	8.7	8.6	8.3	20.8	21.6	14.4	10.5	0.0	14.2	17.2	7.3	9.3	20.3	18.2	6.9	14.1	0.0	13.2	13.2	15.0

Table 8.3: Segmentation accuracy of our ACRF model compared with other state-of-the-art methods on PASCAL dataset.

from challenging urban and rural scenes. On top of 8 background ("scene element") categories, we annotate 9 foreground ("objects") categories - car, person, motorbike, bus, boat, cow, sheep, bicycle, others. We follow the 5-fold cross-validation scheme which splits the data into different 572 training and 143 test images. We use the



(c)	Before	BL1	BL2	ACRF
next-to	5.4%	25.9%	32.2%	**92.5%**
above/below	6.1%	51.72%	61.2%	**76.2%**
in-front/behind	5.7%	50.0%	60.4%	**72.3%**
different horizon	0.1%	38.5%	15.0%	**60.0%**

Figure 8.7: Examples of the learned pair-wise objects relationships are visualized in panel (a,b). The grayscale color code indicates to what degree the relationship is encouraged (white means it is encouraged, black means it is not encouraged and suppressed). Our model learned that (a) a car is likely to be in front of a bus, and a car is unlikely to be below a bus, (b) a car is likely to be behind a person. (c) Prediction accuracy of the objects co-occurrence for each type of relationship averaged over 5-fold validations. The first and last columns show the accuracy before and after applying inference on our full ACRF model, respectively. Notice that there is a consistent improvement across all types. The performance of two baseline methods are reported in the middle two columns which are all inferior then our results.

same STAIR Vision Library (*Gould et al.*, 2009c) used in *Gould et al.* (2009a) to obtain the scene element unary potentials. Pixel-wise segmentation performance is shown in Table 8.1. Our ACRF model outperforms all state-of-the-art methods (*Tighe and Lazebnik*, 2010; *Gould et al.*, 2009a; *Munoz et al.*, 2010; *Ladicky et al.*, 2010a,b) [1] (Table 8.1(a)). A system analysis of our model (Table 8.1(b)) shows that the performances of most foreground classes (seven out of eight) are significantly improved when additional components are added on top of the baseline CRF model, while the performance of the background classes remain almost unchanged. As a result, the full ACRF model obtains the best performance for six out of eight foreground classes and a 14.2% average improvement over the baseline model. Typical results are shown in Fig. 8.8-Top. We highlight that our model can generate instance-based segmentations due to the ability to reason in the *augmented labeling space* $\hat{\mathcal{Q}}$ (Fig. 8.5). Our method

[1] We implement *Ladicky et al.* (2010a,b) by ourselves and evaluate the performance.

can predict the numbers of instances per image accuractely with an average errors of 0.27.

Another advantage of using our model is to improve detection accuracy. We measured detection performance in terms of Recall v.s. False Positive Per Image (FPPI) in Fig. 8.6, where detection results from 5-fold validations are accumulated and shown in one curve. The performance of the proposed model is compared with the pre-trained LSVM (*Felzenszwalb et al.*, 2008). Our model achieves consistent higher recall than the LSVM baseline as shown in Fig. 8.6.

PASCAL dataset. This dataset contains 14,743 images with 21 categories including 20 object categories and 1 scene element category. Only a subset of images have segmentation labels, and we used the standard split for training (749 images), validation (750 images), and testing (750 images). A system analysis of our model (Table 8.2) shows that the performances of most classes were improved when additional components are added on top of the baseline CRF model. Notice that the baseline CRF has a fairly low performance, since we use only the pixel-wise unary responses from the first layer of the hierarchical CRF (*Ladicky et al.*, 2010b). However, our ACRF model is able to significantly boost up the performance and achieves competitive accuracy compared other teams in the challenge (ranked in 4_{th} in Table 8.3) Moreover, the average in predicting the error of numbers of instances per image is only 0.06. Typical results are shown in Fig. 8.8-Bottom.

8.4.1 Relationship Analysis

The learned model parameters for a few typical pair-wise objects relationships are visualized in Fig. 8.7(a,b). In Fig. 8.7(c), we compare the accuracy of predicting the correct relationship of objects before (i.e., raw detections from LSVM (*Felzenszwalb et al.*, 2008)) and after applying inference on our ACRF. A relationship of objects is considered correct if both their object bounding boxes overlap more than 50%

with the ground truth bounding boxes. The accuracy reported in Fig. 8.7(c) is the percentage of correct pairs of objects for each type of relationship. Notice that a significant improvement is achieved by our ACRF model over two baseline methods.

Our baseline methods BL1 and BL2 are defined as follows:

BL1 uses only the detection confidence to prune out detections. In specific, for each pair of bounding boxes with a certain relationship, we assign a score as a sum of scores for both bounding boxes from LSVM. Then, we sample $p\%$ of pairs with highest scores, where p is the percentage of correct ratios for a certain relationship from the training set.

BL2 incorporates pairwise object relationships and prune out detections. Again, for each pair of bounding boxes with a certain relationship, we assign a score as a sum of detection scores for both bounding boxes. Then, we sample pairs within top $p(c_1, c_2)\%$, where $p(c_1, c_2)$ is a class-pair specific percentage of correct ratios from the training set, and c_1 and c_2 is classes corresponding to two bounding boxes.

Using the inferred relationships we can provide high level geometrical description of the scene and determine properties such as: object x is behind object y. Finally, we can obtain 3D pop-up models of the scene from a single image as in Fig. 8.9

8.5 Conclusion

In this chapter, we have presented a unified CRF-based framework for jointly detecting and segmenting "object" and "scene element" categories in natural images. We have shown that our framework incorporates in a coherent fashion various types of (geometrical and semantic) contextual relationships by using property list. Our new formulation generalizes previous results based on CRF where the focus was to capture the co-occurrence between scene element categories only. We have quantitatively and qualitatively demonstrated that our method: i) produces better segmentation results than state-of-the art on the Stanford dataset and competitive results on PASCAL'09 dataset; ii) improves the recall of object instances on Stanford dataset; iii) enables the

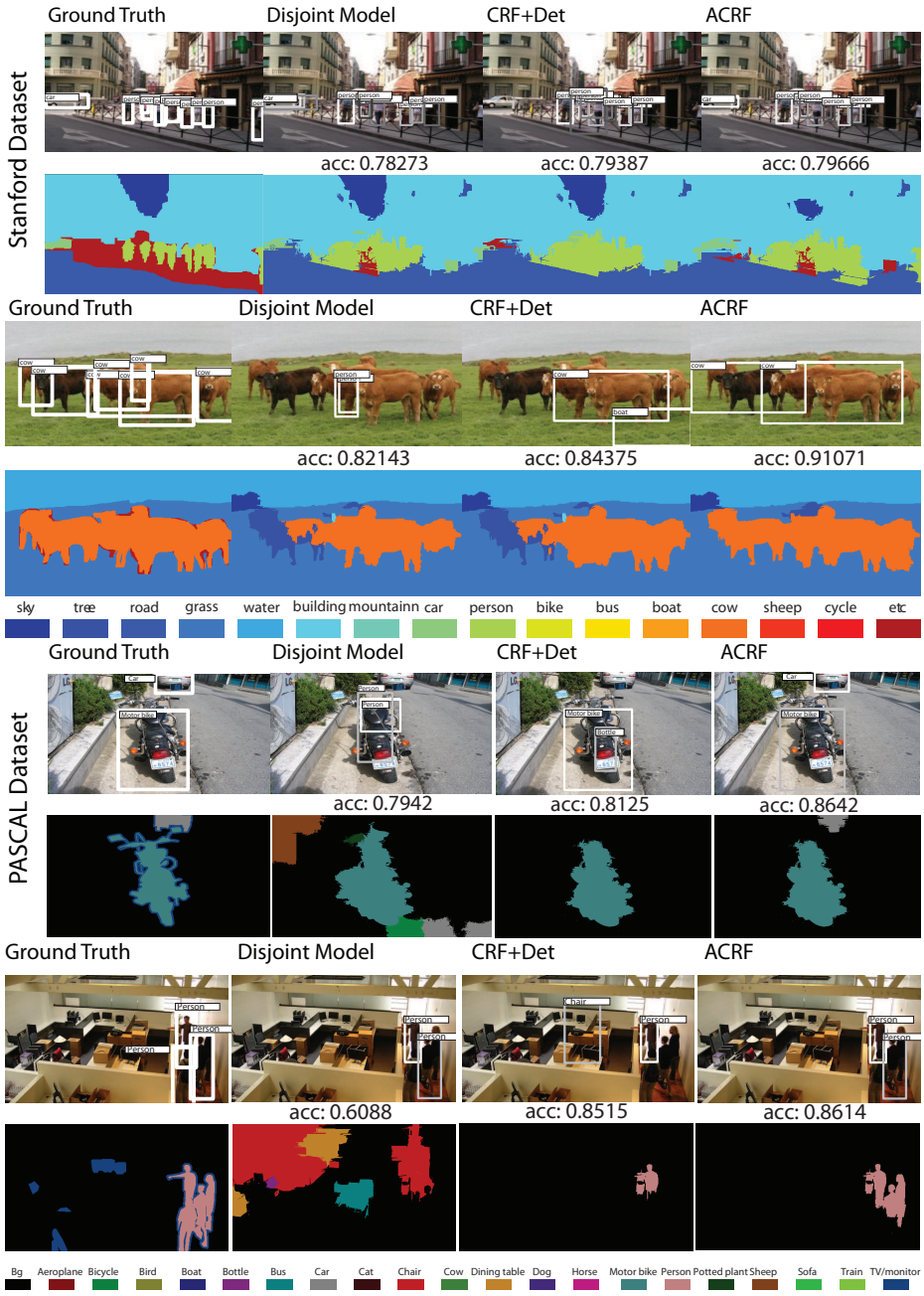


Figure 8.8: Typical results on Stanford (top 4 rows) and PASCAL datasets (bottom 4 rows). Every set of results compare ground truth annotation, disjointed model (disjointedly applied object detection and segmentation), CRF+Det, ACRF, from left to right, respectively. The odd rows show the top K object hypotheses (color-coded bounding boxes representing the confidence ranking from light to dark), where K is the number of recalled objects in the ACRF result. The even rows show the segmentation results (color-code is shown at the bottom).

estimation of contextual relationship among objects and scene elements. Extensions for future work include incorporating more sophisticated types of properties such as

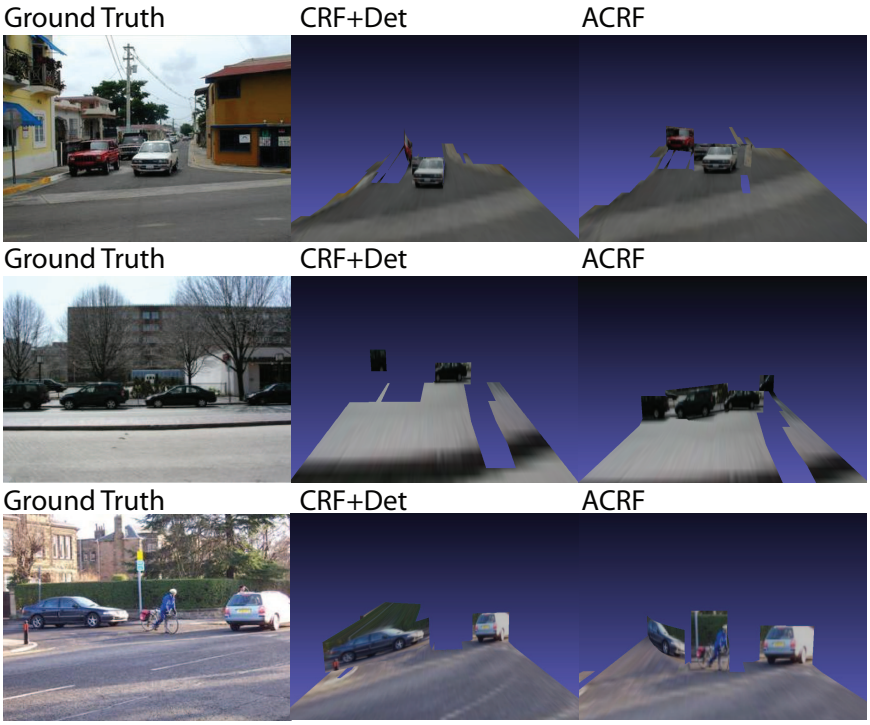


Figure 8.9: 3D pop-up models from Stanford dataset. Videos related to above 3D pop-up models can be found in the project page: http://www.eecs.umich.edu/vision/ACRFproj.html

relative appearance, density, etc.

CHAPTER IX

Conclusion

In this thesis, we have proposed a number of 2D and 3D models for object and scene understanding progressively working towards having the ability to interpret the 3D scene as human can easily do. Here we summarize our contributions and the future work for object recognition, articulated object recognition, and scene understanding, respectively.

9.1 Object Recognition

In the thesis, we demonstrate that object properties such as location, viewpoint, 3D shape, etc. can be extracted by our proposed generative model (Chapter III) and depth-encoded hough voting method (Chapter IV). The proposed methods have also shown the ability to synthesize new view (Sec. 3.3.3) and generate the 3D reconstruction (Sec. 4.2.2) of an object instance during recognition just from a single image.

Despite these success, state-of-the-art methods are still far from the level of accuracy, efficiency and robustness that the human visual system achieves in recognizing, detecting and categorizing objects from images. Recently, several new paradigms have been explored to address the above limitations. One major effort involves large scale object recognition. With the introduction of ultra-large scale datasets such as

the ImageNet (*Deng et al.*, 2009) - a collection of millions of images organized into a hierarchical ontology of thousands of categories - it is now possible to evaluate methods for object categorization that seek to: i) efficiently process these many images and categories; ii) understand objects at different level of specificity; this is also referred as to the fine-grain categorization problem (*Yao et al.*, 2012a; *Duan et al.*, 2012; *Branson et al.*, 2010; *Perona*, 2010). Another major effort is related to the introduction of a recent paradigm whereby objects are modeled and recognized by means of their attributes. As pioneered by *Farhadi et al.* (2009a); *Ferrari and Zisserman* (2007); *Lampert et al.* (2009), visual attributes such as "it is metallic"; "it has wheels" can be used to obtain more effective and descriptive characterizations of object categories (i.e., a car or a truck). This has the benefit of: i) making the "boundaries" between different categories more fluid than in traditional parameterizations; ii) enabling more powerful methods for fine-grained categorization (*Duan et al.*, 2012); iii) provides critical building blocks for transferring visual properties across categories (transfer learning; one short learning) (*Farhadi et al.*, 2009a; *Lampert et al.*, 2009).

9.2 Articulated Object Recognition

We also demonstrate the advantage of incorporating rich relationships among human body parts for human pose estimation. In Chapter V, we have proposed a compositional model for joint human detection and pose estimation. The method achieves a good trade-off between efficiency and accuracy by restricting the model to have a tree structure. Further in Chapter VI, we have proposed a novel efficient branch-and-bound algorithm for finding the best human body configuration described by a complex loopy model which incorporates most of the relationships among human body parts. The proposed branch-and-bound algorithm is very general and we have applied it to solve computational biology problems such as protein design.

All the methods that we proposed focus on recognizing a single articulated ob-

ject. However, people often appear in groups and their body parts interact to each other which causes occlusion and confusion of the membership of the parts (i.e., this arm belongs to the first person, that arm belongs to the second person, and so on). Researchers have identified this problem and proposed methods for jointly recognizing multiple people and resolving the ambiguity of parts membership and occlusion. *Eichner and Ferrari* (2010) present a novel multi-person pose estimation framework, which extends pictorial structures to explicitly model interactions between people and to estimate their poses jointly. *Yang et al.* (2012) present a computational formulation of visual proxemics by attempting to label each pair of people using a set of physically-based "touch codes" (e.g., Hand-hand, Elbow-shoulder, etc.). Comparing to *Eichner and Ferrari* (2010), *Yang et al.* (2012) focus on handling more complex interactions for only a pair of people. In the future, we like to extend our Branch-and-Bound inference algorithm to jointly recognize multiple human poses capturing the body part interaction across different individuals.

9.3 Scene Understanding

We propose two models for scene understanding which utilize the object properties extracted by our methods to improve the object segmentation and detection performance (Chapter VII and VIII). Both models incorporate the relationships among objects and the scene layout established through the extracted properties. As a result, our methods are able to infer the object category of every image region, the orientations of the supporting planes, and the location and 3D pose of the object instances within the scene. In the future, we would like to apply our methods to a video sequence instead of a single image. We believe that the ability to utilize the relationships among object and scene layout across the time domain will significantly improve the object segmentation and detection performance. One of the biggest challenge of extending our methods to video sequence is to keep the recognition process

tractable and efficient for practical usage.

In the future, we also would like to jointly solve general human behavior understanding problems, such as collective activity recognition (*Choi et al.*, 2011; *Lan et al.*, 2010), individual action recognition (*Yang et al.*, 2010; *Yao et al.*, 2011), and human pose estimation. We believe that it is also possible to combine human behavior understanding with general scene understanding problems so that a unified model incorporates human-object, human-human, and human-layout relationships to improve the understanding of both.

BIBLIOGRAPHY

BIBLIOGRAPHY

Agarwal, S., N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski (2009), Building rome in a day, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Aliaga, D. G., T. Funkhouser, D. Yanovsky, and I. Carlbom (2003), Sea of images: A dense sampling approach for rendering large indoor environments, *IEEE Comput. Graph. Appl.*, *23*(6), 22–30.

Andriluka, M., S. Roth, and B. Schiele (2009), Pictorial structures revisited:people detection and articulated pose estimation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Andriluka, M., S. Roth, and B. Schiele (2010), Monocular 3d pose estimation and tracking by detection, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Arie-Nachimson, M., and R. Basri (2009), Constructing implicit 3d shape models for pose estimation, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Ballard, D. H. (1981), Generalizing the hough transform to detect arbitrary shapes, *Pattern Recognition*, *13*(2), 111–122.

Bao, S. Y., M. Sun, and S. Savarese (2010a), Toward coherent object detection and scene layout understanding, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Bao, S. Y., M. Sun, and S. Savarese (2010b), Toward coherent object detection and scenelayout understanding, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Barinova, O., V. Lempitsky, and P. Kohli (2012), On detection of multiple object instances using hough transforms, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*.

Barrow, H. G., and J. M. Tenenbaum (1981), Interpreting line drawings as three-dimensional surfaces, *Artif. Intell.*, *17*(1-3), 75–116.

Batra, D., S. Nowozin, and P. Kohli (2011), Tighter relaxations for MAP-MRF inference: A local primal-dual gap based separation algorithm, in *Intern. Conf. on Artificial Intelligence and Statistics (AISTATS)*.

Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool (2008), Speeded-up robust features (surf), *Comput. Vis. Image Underst.*, *110*(3), 346–359.

Bergtholdt, M., J. Kappes, S. Schmidt, and C. Schnrr (2010), A study of parts-based object class detection using complete graphs, *International Journal of Computer Vision (IJCV)*, *87*, 93–117.

Berkman, O., and U. Vishkin (1993), Recursive star-tree parallel data structure, *SIAM Journal on Computing*, *22*, 221–242.

Biederman, I. (1985), Human image understanding: Recent research and theory, *Computer Vision, Graphics and Image Understanding*, *32*, 29–73.

Binford, T. (1971), Visual perception by computer, in *IEEE conference on Systems and Control*.

Blei, D. M. (2004), Variational methods for the dirichlet process, in *Intl. Conf. on Machine Learning (ICML)*.

Borenstein, E., and S. Ullman (2002), Class-specific, top-down segmentation, in *European Conference of Computer Vision (ECCV)*.

Boros, E., and P. Hammer (2002), Pseudo-boolean optimization., *Discrete Applied Mathematics*.

Bosch, X. B., J. M. Gonfaus, J. van de Weijer, A. D. Bagdanov, J. Serrat, and J. Gonz'alez (2010), Harmony potentials for joint classification and segmentation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Bouchard, G., and B. Triggs (2005), Hierarchical part-based visual object categorization, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Bouguet, J.-Y., and P. Perona (1995), Visual navigation using a single camera, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Bourdev, L., and J. Malik (2009), Poselets: Body part detectors trained using 3d human pose annotations, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Bourdev, L., S. Maji, T. Brox, and J. Malik (2010), Detecting people using mutually consistent poselet activations, in *European Conference of Computer Vision (ECCV)*.

Bowyer, K., and C. R. Dyer (1990), Aspect graphs: An introduction and survey of recent results, *International Journal of Imaging Systems and Technology*, *2*, 315–328.

Boykov, Y., O. Veksler, and R. Zabih (2001), Fast approximate energy minimization via graph cuts, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *23*, 1222–1239.

Branson, S., C. Wah, B. Babenko, F. Schroff, P. Welinder, P. Perona, and S. Belongie (2010), Visual recognition with humans in the loop, in *ECCV*.

Brostow, G. J., J. Shotton, J. Fauqueur, and R. Cipolla (2008), Segmentation and recognition using structure from motion point clouds, in *European Conference of Computer Vision (ECCV)*.

Brown, M., and D. G. Lowe (2005), Unsupervised 3d object recognition and reconstruction in unordered datasets, in *Conf. on 3D Imaging Modeling (3DIM)*.

Chen, X., S. B. Kang, Y.-Q. Xu, J. Dorsey, and H.-Y. Shum (2008), Sketching reality: Realistic interpretation of architectural designs, *ACM Trans. Graph.*, *27*(2), 1–15.

Chiu, H., L. Kaelbling, and T. Lozano-Perez (2007), Virtual training for multi-view object class recognition, in *Proc. Computer Vision and Pattern Recognition*.

Choi, W., K. Shahid, and S. Savarese (2011), Learning context for collective activity recognition, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Colombo, C., A. Del Bimbo, and F. Pernici (2005), Metric 3d reconstruction and texture acquisition of surfaces of revolution from a single uncalibrated view, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *27*(1), 99–114.

Cornelis, N., B. Leibe, K. Cornelis, and L. Van Gool (2006), 3d city modeling using cognitive loops, in *Conf. on 3D Processing Visualization Transmission (3DPVT)*.

Criminisi, A., P. Perez, and K. Toyama (2003), Object removal by exemplar-based inpainting, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Csurka, G., C. R. Dance, L. Fan, J. Willamowski, and C. Bray (2004), Visual categorization with bags of keypoints, in *European Conference of Computer Vision (ECCV), Workshop on Statistical Learning in Computer Vision*, pp. 1–22.

Dalal, N., and B. Triggs (2005), Histograms of oriented gradients for human detection, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

de Givry, S., F. Heras, J. Larrosa, and M. Zytnicki (2005), Existential arc consistenct: getting closer to full arc consistenct in weighted CSPs, in *Intern. Joint Conf. on Artificial Intelligence (IJCAI)*.

Debevec, P. E., C. J. Taylor, and J. Malik (1996), Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach, in *SIGGRAPH: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 11–20.

Dempster, A., N. Laird, and D. Rubin (1977), Maximum likelihood from incomplete data via the em algorithm, *Journal of the Royal Statistical Society*, *39*, 1–38.

Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei (2009), ImageNet: A Large-Scale Hierarchical Image Database, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Desai, C., D. Ramanan, and C. Fowlkes (2009), Discriminative models for multi-class object layout, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Dick, A. R., P. H. S. Torr, and R. Cipolla (2004), Modelling and interpretation of architecture from several images, *Intl. Journal of Computer Vision (IJCV)*, *60*(2), 111–134.

Duan, K., D. Parikh, D. Crandall, and K. Grauman (2012), Discovering localized attributes for fine-grained recognition, in *CVPR*.

Efros, A. A., and T. K. Leung (1999), Texture synthesis by non-parametric sampling, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Eichner, M., and V. Ferrari (2009), Better appearance models for pictorial structures, in *British Machine Vision Conference (BMVC)*.

Eichner, M., and V. Ferrari (2010), We are family: Joint pose estimation of multiple persons, in *European Conference of Computer Vision (ECCV)*.

Everingham, M., A. Zisserman, C. K. I. Williams, and L. Van Gool (2006), *The PASCAL Visual Object Classes Challenge 2006 (VOC2006) Results*.

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

Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman (2009), *The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results*.

Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman (2010), *The PASCAL VOC2010 Results*.

Farhadi, A., I. Endres, D. Hoiem, and D. Forsyth (2009a), Describing objects by their attributes, in *CVPR*.

Farhadi, A., M. K. Tabrizi, I. Endres, and D. Forsyth (2009b), A latent model of discriminative aspect, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Fei-Fei, L., R. Fergus, and P. Perona (2004), Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories, in *CVPR*.

Felzenszwalb, P., D. McAllester, and D. Ramanan (2008), A discriminatively trained, multiscale, deformable part model, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Felzenszwalb, P. F., and D. P. Huttenlocher (2004a), Efficient graph-based image segmentation, *Intl. Journal of Computer Vision (IJCV)*, *59*(2), 167–181.

Felzenszwalb, P. F., and D. P. Huttenlocher (2004b), Distance transforms of sampled functions, *Tech. rep.*, Cornell Computing and Information Science.

Felzenszwalb, P. F., and D. P. Huttenlocher (2005), Pictorial structures for object recognition, *Intl. Journal of Computer Vision (IJCV)*, *61*(1), 55–79.

Felzenszwalb, P. F., R. B. Girshick, D. McAllester, and D. Ramanan (2010), Object detection with discriminatively trained part-based models, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *32*, 1627–1645.

Fergus, R., P. Perona, and A. Zisserman (2003), Object class recognition by unsupervised scale-invariant learning, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Ferrari, V., and A. Zisserman (2007), Learning visual attributes, in *NIPS*.

Ferrari, V., T. Tuytelaars, and L. Gool (2006), Simultaneous object recognition and segmentation from single or multiple model views, *IJCV*, *67*, 159–188.

Ferrari, V., L. Fevrier, F. Jurie, and C. Schmid (2008a), Groups of adjacent contour segments for object detection, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *30*(1), 36–51.

Ferrari, V., M. M. Jimenez, and A. Zisserman (2008b), Progressive search space reduction for human pose estimation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Fischler, M. A., and R. C. Bolles (1981), Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, *Comm. of the ACM*, *24*, 381–395.

Flock, H. R. (1964), Three theoretical views of slant perception, *Psychological Bulletin*, *62*(2), 110–121.

Gall, J., and V. Lempitsky (2009), Class-specific hough forests for object detection, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Gertz, M., and S. Wright (2001), Object-oriented software for quadratic programming, *ACM Transactions on Mathematical Software*, *29*, 58–81.

Globerson, A., and T. Jaakkola (2008), Fixing max-product: Convergent message passing algorithms for MAP LP-relaxations, in *Advances in Neural Information Processing Systems (NIPS)*.

Gonfaus, J. M., X. Boix, J. van de Weijer, A. D. Bagdanov, J. Serrat, and J. Gonz'alez (2010), Harmony potentials for joint classification and segmentation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Gordon, I., and D. G. Lowe (2006), What and where: 3d object recognition with accurate pose, in *Toward Category-Level Object Recognition*.

Gould, S., R. Fulton, and D. Koller (2009a), Decomposing a scene into geometric and semantically consistent regions, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Gould, S., T. Gao, and D. Koller (2009b), Region-based segmentation and object detection, in *Advances in Neural Information Processing Systems (NIPS)*.

Gould, S., O. Russakovsky, I. Goodfellow, P. Baumstarck, A. Ng, and D. Koller (2009c), The stair vision library (v2.3).

Grauman, K., and T. Darrell (2005), The pyramid match kernel: discriminative classification with sets of image features, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Gupta, A., and L. Davis (2008a), Beyond nouns: Exploiting prepositions and comparators for learning visual classifiers, in *European Conference of Computer Vision (ECCV)*.

Gupta, A., and L. S. Davis (2008b), Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers, in *European Conference of Computer Vision (ECCV)*.

Gupta, A., A. A. Efros, and M. Hebert (2010), Blocks world revisited: Image understanding using qualitative geometry and mechanics, in *European Conference of Computer Vision (ECCV)*.

Han, F., and S. Zhu (2005), Bottom-up/top-down image parsing by attr ibute graph grammar, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Hays, J., and A. A. Efros (2007), Scene completion using millions of photographs, *ACM Trans. Graph.*, *26*(3).

He, X., R. S. Zemel, and M. Á. Carreira-Perpiñán (2004), Multiscale conditional random fields for image labeling, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Hedau, V., D. Hoiem, and D. Forsyth (2009), Recovering the spatial layout of cluttered rooms, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Hedau, V., D. Hoiem, and D. Forsyth (2010), Thinking inside the box: Using appearance models and context based on room geometry, in *European Conference of Computer Vision (ECCV)*.

Heitz, G., and D. Koller (2008), Learning spatial context: Using stuff to find things, in *European Conference of Computer Vision (ECCV)*.

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

Hoeim, D., and S. Savarese (2011), *Representations and Techniques for 3D Object Recognition and Scene Interpretation*, Morgan and Claypool.

Hoeim, D., C. Rother, and J. Winn (2007), 3d layoutcrf for multi-view object class recognition and segmentation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Hoiem, D., A. A. Efros, and M. Hebert (2005a), Geometric context from a single image, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Hoiem, D., A. A. Efros, and M. Hebert (2005b), Automatic photo pop-up, *ACM Trans. Graph.*, *24*(3), 577–584.

Hoiem, D., A. A. Efros, and M. Hebert (2006), Putting objects in perspective, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Hoiem, D., A. Efros, and M. Hebert (2007), Recovering surface layout from an image, *Intl. Journal of Computer Vision (IJCV)*.

Hoiem, D., A. A. Efros, and M. Hebert (2008), Closing the loop on scene interpretation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Hong, E.-J., and T. Lozano-Perez (2006), Protein side-chain placement through MAP estimation and problem-size reduction, in *Workshop on Algorithms in Bioinformatics (WABI)*.

Horry, Y., K.-I. Anjyo, and K. Arai (1997), Tour into the picture: using a spidery mesh interface to make animation from a single image, in *SIGGRAPH: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 225–232.

Hsiao, E., A. Collet, and M. Hebert (2010), Making specific features less discriminative to improve point-based 3d object recognition, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Huttenlocher, D. P., and S. Ullman (1987), Object recognition using alignment, in *ICCV*.

Ionescu, C., L. Bo, and C. Sminchisescu (2009), Structural svm for visual localization and continuous state estimation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Jacobs, D., and R. Basri (1999), 3d to 2d pose determination with regions, *International Journal of Computer Vision*, *2/3*(34), 123–145.

Jiang, H., and D. R. Martin (2008), Global pose estimation using non-tree models, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Jiang, N., P. Tan, and L.-F. Cheong (2009), Symmetric architecture modeling with a single image, *ACM Trans. Graph.*, *28*(5), 113:1–113:8.

Kanade, T. (1981), Recovery of the three-dimensional shape of an object from a single view, *Artificial Intelligence*, *17*, 409 – 460.

Karpenko, O. A., and J. F. Hughes (2006), Smoothsketch: 3d free-form shapes from complex sketches, *ACM Trans. Graph.*, *25/3*, 589–598.

Koenderink, J., and A. V. Doorn (1979), The internal representation of solid shape with respect to vision, *Biol. Cybern*, *32*, 211–216.

Kohli, P., L. Ladicky, and P. H. Torr (2008), Robust higher order potentials for enforcing label consistency, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Koller, D., and N. Friedman (2009), Probabilistic graphical models: Principles and techniques, *MIT Press*.

Kolmogorov, V. (2006), Convergent tree-reweighted message passing for energy minimization, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *28*(10), 1568 –1583.

Kolmogorov, V., and R. Zabih (2004), What energy functions can be minimizedvia graph cuts?, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *26*, 147–159.

Komodakis, N., and N. Paragios (2008), Beyond loose LP-relaxations: Optimizing MRFs by repairing cycles, in *European Conference of Computer Vision (ECCV)*.

Kopf, J., B. Neubert, B. Chen, M. Cohen, D. Cohen-Or, O. Deussen, M. Uyttendaele, and D. Lischinski (2008), Deep photo: model-based photograph enhancement and viewing, *ACM Trans. Graph.*, *27*(5), 116:1–116:10.

Koster, A., C. P. M. van Hoesel, and A. W. J. Kolen (1998), The partial constraint satisfaction problem: Facets and lifting theorems, *Oper. Res. Lett.*, *23*, 89–97.

Kushal, A., and J. Ponce (2006), Modeling 3d objects from stereo views and recognizing them in photographs, in *European Conference of Computer Vision (ECCV)*.

Kushal, A., C. Schmid, , and J. Ponce (2007), Flexible object models for category-level 3d object recognition, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Kutulakos, K. N., and S. M. Seitz (2000), A theory of shape by space carving, *Intl. Journal of Computer Vision (IJCV)*, *38*(3), 199–218.

Ladicky, L., C. Russell, P. Kohli, and P. Torr (2010a), Graph cut based inference with co-occurrence statistics, in *European Conference of Computer Vision (ECCV)*.

Ladicky, L., C. Russell, P. Kohli, and P. H. Torr (2010b), Graph cut based inference with co-occurrence statistics, in *European Conference of Computer Vision (ECCV)*.

Ladicky, L., P. Sturgess, K. Alahari, C. Russell, and P. H. Torr (2010c), What,where & how many? combining object detectors and CRFs, in *European Conference of Computer Vision (ECCV)*.

Lafferty, J., A. McCallum, and F. Pereira (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in *ICML*.

Lampert, C. H., H. Nickisch, and S. Harmeling (2009), Learning to detect unseen object classes by between-class attribute transfer, in *CVPR*.

Lan, T., Y. Wang, W. Yang, and G. Mori (2010), Beyond actions: Discriminative models for contextual group activities, in *Advances in Neural Information Processing Systems (NIPS)*.

Lan, X., and D. P. Huttenlocher (2005), Beyond trees: Common factor models for 2d human pose recovery, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Land, A. H., and A. G. Doig (1960), An automatic method of solving discrete programming problems, *Econometrica*, pp. 497–520.

Laurentini, A. (1994), The visual hull concept for silhouette-based image understanding, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *16*(2), 150–162.

Lazebnik, S., C. Schmid, and J. Ponce (2006), Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Lee, D. C., M. Hebert, and T. Kanade (2009), Geometric reasoning for single image structure recovery, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Leibe, B., A. Leonardis, and B. Schiele (2004), Combined object categorization and segmentation with an implicit shape model, in *European Conference of Computer Vision (ECCV) workshop on statistical learning in computer vision*.

Levoy, M., and P. Hanrahan (1996), Light field rendering, in *SIGGRAPH: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 31–42.

Levoy, M., et al. (2000), The digital michelangelo project: 3d scanning of large statues, in *SIGGRAPH: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 131–144.

Li, C., A. Kowdle, A. Saxena, and T. Chen (2010), Towards holistic scene understanding: Feedback enabled cascaded classification models, in *Advances in Neural Information Processing Systems (NIPS)*.

Li, L.-J., and L. Fei-Fei (2007), What, where and who? classifying event by scene and object recognition, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Li, L.-J., G. Wang, and L. Fei-Feie (2007), Optimol: automatic online picture collection via incremental model learning, in *CVPr*.

Li, L.-J., R. Socher, and L. Fei-Fei (2009a), Towards total scene understanding:classification, annotation and segmentation in an automatic framework, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Li, Y., L. Gu, and T. Kanade (2009b), A robust shape model for multi-view car alignment, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Liebelt, J., and C. Schmid (2010), Multi-view object class detection with a 3D geometric model, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Liebelt, J., C. Schmid, and K. Schertler (2008), Viewpoint-independent object class detection using 3d feature maps, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Liebowitz, D., A. Criminisi, and A. Zisserman (1999), Creating architectural models from images, in *Annual Conference of the European Association for Computer Graphics (Eurographics)*, vol. 18, pp. 39–50.

Lowe, D. (1999), Object recognition from local scale-invariant features, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Lowe, D., and T. Binford (1985), The recovery of three-dimensional structure from image curves, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *7*, 320–326.

Lucas, B. D., and T. Kanade (1981), An iterative image registration technique with an application to stereo vision, in *Intern. Joint Conf. on Artificial Intelligence (IJCAI)*.

Maji, S., and J. Malik (2009), Object detection using a max-margin hough tranform, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Marinescu, R., and R. Dechter (2007), Best-first and/or search for graphical models, in *National Conf. on Artificial intelligence (AAAI)*.

Marr, D. (1978), Representing visual information, in *Computer Vision Systems*.

Marr, D. (1982), *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*, Henry Holt and Co., Inc.

Matas, J., O. Chum, M. Urban, and T. Pajdla (2002), Robust wide baseline stereo from maximally stable extremal regions, in *British Machine Vision Conference (BMVC)*.

McMillan, L., and G. Bishop (1995), Plenoptic modeling: an image-based rendering system, in *SIGGRAPH: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pp. 39–46.

Meltzer, T., C. Yanover, and Y. Weiss (2005), Globally optimal solutions for energy minimization in stereo vision using reweighted belief propagation, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Mendonça, P. R. S., K.-Y. K. Wong, and R. Cipolla (2000), Camera pose estimation and reconstruction from image profiles under circular motion, in *European Conference of Computer Vision (ECCV)*.

Microsoft Corp. Redmond WA (2010), *Kinect for Xbox 360*.

Mikolajczyk, K., and C. Schmid (2002), An affine invariant interest point detector, in *International Journal of Computer Vision*, pp. 128–142.

Mitra, N. J., L. J. Guibas, and M. Pauly (2006), Partial and approximate symmetry detection for 3d geometry, *ACM Trans. Graph.*, *25*(3), 560–568.

Munoz, D., J. A. Bagnell, and M. Hebert (2010), Stacked hierarchical labeling, in *European Conference of Computer Vision (ECCV)*.

Ng, J., and S. Gong (1999), Multi-view face detection and pose estimation using a composite support vector machine across the view sphere, in *RATFG-RTS*.

Obdrzalek, S., and J. Matas (2002), Object recognition using local affine frames on distinguished regions, in *British Machine Vision Conference (BMVC)*.

Ommer, B., and J. Malik (2009), Multi-scale object detection by clustering lines, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Oswald, M. R., E. Töppe, K. Kolev, and D. Cremers (2009), Non-parametric single view reconstruction of curved objects using convex optimization, in *DAGM Symposium on Pattern Recognition*.

Palmer, S. (1975), Visual perception and world knowledge: notes on a model of sensory-cognitive interaction, in *Explorations in Cognition*.

Palmer, S., E. Rosch, and P. Chase (1981), Canonical perspective and the perception of objects, *Attention and Performance*, *9*, 135–151.

Palmer, S. E. (1999), Vision science-photons to phenomenology, *MIT Press*.

Pauly, M., N. J. Mitra, J. Giesen, M. Gross, and L. J. Guibas (2005), Example-based 3d scan completion, in *SGP: Proceedings of the third Eurographics symposium on Geometry processing*.

Payet, N., and S. Todorovic (2011a), From contours to 3d object detection and pose estimation, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Payet, N., and S. Todorovic (2011b), Scene shape from textures of objects, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Pérez, P., M. Gangnet, and A. Blake (2003), Poisson image editing, *ACM Trans. Graph.*, *22*(3), 313–318.

Perona, P. (2010), Visions of a visipedia, *Proceedings of the IEEE*, *98*, 1526–1534.

Pingkun Yan, S. M. K., and M. Shah (2007), 3d model based object class detection in an arbitrary view, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Poggio, T., and S. Edelman (1990), A neural network that learns to recognize three-dimensional objects, *Nature*, *343*, 263–266.

Pollefeys, M., L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch (2004), Visual modeling with a hand-held camera, *Intl. Journal of Computer Vision (IJCV)*, *59*(3), 207–232.

Prasad, M., and A. Fitzgibbon (2006), Single view reconstruction of curved surfaces, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Prasad, M., A. W. Fitzgibbon, A. Zisserman, and L. J. V. Gool (2010), Finding nemo: Deformable object class modelling using curve matching, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

P.Winston (1975), Learning structural descriptions from examples, in *The Psycology of Computer Vision*.

Rabinovich, A., A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie (2007), Objects in context, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Ramanan, D. (2006), Learning to parse images of articulated bodies, in *Advances in Neural Information Processing Systems (NIPS)*.

Ren, X., A. C. Berg, and J. Malik (2005), Recovering human body configurations using pairwise constraints between parts, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Romea, A. C., S. S. D. Berenson, and D. Ferguson (2009), Object recognition and full pose registration from a single image for robotic manipulation, in *IEEE International Conference on Robotics and Automation (ICRA)*.

Rother, C., V. Kolmogorov, and A. Blake (2004), "grabcut": interactive foreground extraction using iterated graph cuts, *ACM Trans. Graph.*, *23*(3), 309–314.

Rother, C., V. Kolmogorov, V. Lempitsky, and M. Szummer (2007), Optimizing binary mrfs via extended roof duality, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Rothganger, F., S. Lazebnik, C. Schmid, and J. Ponce (2003), 3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints., in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Rusinkiewicz, S., O. Hall-Holt, and M. Levoy (2002), Real-time 3d model acquisition, *ACM Trans. Graph.*, *21*(3), 438–446.

Russell, B. C., A. Torralba, K. P. Murphy, and W. T. Freeman (2008), Labelme: A database and web-based tool for image annotation, *Intl. Journal of Computer Vision (IJCV)*, *77*(1-3), 157–173.

Sapp, B., C. Jordan, and B. Taskar (2010a), Adaptive pose priors for pictorial structures, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Sapp, B., A. Toshev, and B. Taskar (2010b), Cascaded models for articulated pose estimation., in *European Conference of Computer Vision (ECCV)*.

Sapp, B., D. Weiss, and B. Taskar (2010c), Sidestepping intractable inference with structured ensemble cascades, in *Advances in Neural Information Processing Systems (NIPS)*.

Sapp, B., D. Weiss, and B. Taskar (2011), Parsing human motion with stretchable models, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Savarese, S., and L. Fei-Fei (2007), 3d generic object categorization, localization and pose estimation, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Savarese, S., and L. Fei-Fei (2008), View synthesis for recognizing unseen poses of object classes, in *European Conference of Computer Vision (ECCV)*.

Savarese, S., M. Andreetto, H. Rushmeier, F. Bernardin, and P. Perona (2006a), 3d reconstruction by shadow carving: Theory and practical evaluation, *Intl. Journal of Computer Vision (IJCV)*, *71*(3), 305–336.

Savarese, S., J. Winn, and A. Criminisi (2006b), Discriminative object class models of appearance and shape by correlatons, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Saxena, A., M. Sun, and A. Y. Ng (2009), Make3d: Learning 3d scene structure from a single still image, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *31*(5), 824–840.

Schmid, C. (2006), Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Schneiderman, H., and T. Kanade (2000), A statistical approach to 3D object detection applied to faces and cars, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Schwing, A., T. Hazan, M. Pollefeys, and R. Urtasun (2012), Efficient structured prediction for 3d indoor scene understanding, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Seitz, S. M., and C. R. Dyer (1996), View morphing, in *SIGGRAPH: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*.

Seitz, S. M., B. Curless, J. Diebel, D. Scharstein, and R. Szeliski (2006), A comparison and evaluation of multi-view stereo reconstruction algorithms, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Shamir, A., and S. Avidan (2009), Seam carving for media retargeting, *Commun. ACM*, *52*(1), 77–85.

Shilane, P., P. Min, M. Kazhdan, and T. Funkhouser (2004), The princeton shape benchmark, in *SMI '04: Proceedings of the Shape Modeling International 2004*.

Shimony, S. E. (1994), Finding MAPs for belief networks is NP-hard, *Artificial Intelligence*, *68*, 399–410.

Shlezinger, M. I. (1976), Syntactic analysis of two-dimensional visual signals in the presence of noise, *Cybernetics and Systems Analysis*, *12*, 612–628.

Shotton, J., J. Winn, C. Rother, and A. Criminisi (2006), Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation, in *European Conference of Computer Vision (ECCV)*.

Shotton, J., A. Blake, and R. Cipolla. (2008), Semantic texton forests for image categorization and segmentation., in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Sigal, L., and M. J. Black (2006), Measure locally, reason globally: Occlusion-sensitive articulated pose estimation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Snavely, N., S. M. Seitz, and R. Szeliski (2006), Photo tourism: exploring photo collections in 3d, *ACM Trans. Graph.*, *25*(3), 835–846.

Sontag, D., and T. Jaakkola (2009), Tree block coordinate descent for MAP in graphical models, in *Intern. Conf. on Artificial Intelligence and Statistics (AISTATS)*.

Sontag, D., A. Globerson, and T. Jaakkola (2008a), Clusters and coarse partitions in LP relaxations, in *Advances in Neural Information Processing Systems (NIPS)*.

Sontag, D., T. Meltzer, A. Globerson, Y. Weiss, and T. Jaakkola (2008b), Tightening LP relaxations for MAP using message-passing, in *Conf. on Uncertainty in Artificial Intelligence (UAI)*.

Stark, M., M. Goesele, and B. Schiele (2010), Back to the future: Learning shape models from 3d cad data, in *British Machine Vision Conference (BMVC)*.

Sudderth, E. B., A. Torralba, W. T. Freeman, and A. S. Willsky (2008), Describing visual scenes using transformed objects and parts, *Intl. Journal of Computer Vision (IJCV)*, *77*(1-3), 291–330.

Sun, M., and S. Savarese (2011), Articulated part-based model for joint object detection and pose estimation, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Sun, M., H. Su, S. Savarese, and L. Fei-Fei (2009), A multi-view probabilistic model for 3d object classes, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Sun, M., S. Y.-Z. Bao, and S. Savarese (2010a), Object detection with geometrical context feedback loop, in *British Machine Vision Conference (BMVC)*.

Sun, M., G. Bradski, B.-X. Xu, and S. Savarese (2010b), Depth-encoded hough voting for coherent object detection, pose estimation, and shape recovery, in *European Conference of Computer Vision (ECCV)*.

Sun, M., B.-s. Kim, P. Kolih, and S. Savarese (2012a), Relating things and stuff via object property interactions, in *European Conference of Computer Vision (ECCV) workshop on Higher-Order Models and Global Constraints in Computer Vision*.

Sun, M., M. Telaprolu, H. Lee, and S. Savarese (2012b), Efficient and exact MAP-MRF inference using branch and bound, in *Intern. Conf. on Artificial Intelligence and Statistics (AISTATS)*.

Sun, M., M. Telaprolu, H. Lee, and S. Savarese (2012c), An efficient branch-and-bound algorithm for optimal human pose estimation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Tao, M. W., M. K. Johnson, and S. Paris (2010), Error-tolerant image compositing, in *European Conference of Computer Vision (ECCV)*.

Tarr, M., and S. Pinker (1989), Mental rotation and orientation-dependence in shape recognition, *Cognitive Phycology*, *21*, 233–282.

Teller, S., M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master (2003), Calibrated, registered images of an extended urban area, *Intl. Journal of Computer Vision (IJCV)*, *53*(1), 93–107.

Thomas, A., V. Ferrari, B. Leibe, T. Tuytelaars, B. Schiele, and L. V. Goo (2006), Towards multi-view object class detection, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Thomas, A., V. Ferrari, B. Leibe, T. Tuytelaars, and L. J. V. Gool (2007), Depth-from-recognition: Inferring meta-data by cognitive feedback, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Tian, T.-P., and S. Sclaroff (2010), Fast globally optimal 2D human detection with loopy graph models, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Tighe, J., and S. Lazebnik (2010), Superparsing: Scalable nonparametric image parsing with superpixels, in *European Conference of Computer Vision (ECCV)*.

Toldo, R., and A. Fusiello (2008), Robust multiple structures estimation with j-linkage, in *European Conference of Computer Vision (ECCV)*.

Torralba, A., K. P. Murphy, W. T. Freeman, and M. A. Rubin (2003), Context-based vision system for place and object recognition, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Tran, D., and D. Forsyth (2010), Improved human parsing with a full relational model, in *European Conference of Computer Vision (ECCV)*.

Tsai, G., C. Xu, J. Liu, and B. Kuipers (2011), Real-time indoor scene understanding using bayesian filtering with motion cues, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Tsochantaridis, I., T. Hofmann, T. Joachims, and Y. Altun (2004), Support vector machine learning for interdependent and structured output spaces, in *Intl. Conf. on Machine Learning (ICML)*.

Ullman, S., and R. Basri (1991), Recognition by linear combinations of models, *TPAMI*, *13*, 992–1006.

Viola, P., and M. J. Jones (2004), Robust real-time face detection, *Intl. Journal of Computer Vision (IJCV)*, *57*(2), 137–154.

Wainwright, M. J., and M. I. Jordan (2008), Graphical models, exponential families, and variational inference, *Found. Trends Mach. Learn.*, *1*, 1–305.

Wainwright, M. J., T. S. Jaakkola, and A. S. Willsky (2005), MAP estimation via agreement on trees: message-passing and linear programming, *IEEE Trans. Information Theory*, *51*(11), 3697 – 3717.

Wang, H., S. Gould, and D. Koller (2010), Discriminative learning with latent variables for cluttered indoor scene understanding, in *European Conference of Computer Vision (ECCV)*.

Wang, Y., and G. Mori (2008), Multiple tree models for occlusion and spatial constraints in human pose estimation, in *European Conference of Computer Vision (ECCV)*.

Wang, Y., D. Tran, and Z. Liao (2011), Learning hierarchical poselets for human parsing, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Weber, M., W. Einhaeuser, M. Welling, and P. Perona (2000a), Viewpoint-invariant learning and detection of human heads, in *Proc. 4th Int. Conf. Autom. Face and Gesture Rec.*, pp. 20–27.

Weber, M., M. Welling, and P. Perona (2000b), Unsupervised learning of models for recognition, in *European Conference of Computer Vision (ECCV)*.

Werner, T. (2007), A linear programming approach to max-sum problem: A review, *IEEE Trans. Pattern Analysis and Machine Intelligence (TPAMI)*, *29*, 1165–1179.

Werner, T. (2008), High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF), in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Winn, J., and J. Shotton (2006), The layout consistent random field for recognizing and segmenting partially occluded objects, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Xiang, Y., and S. Savarese (2012), Estimating the aspect layout of object categories, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Xiao, J., and M. Shah (2004), Tri-view morphing, *Comput. Vis. Image Underst. (CVIU)*, *96*(3), 345–366.

Xu, C., B. Kuipers, and A. Murarka (2009), 3d pose estimation for planes, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV) workshop on 3D Representation for Recognition (3dRR-09)*.

Yan, P., D. Khan, and M. Shah (2007), 3d model based object class detection in an arbitrary view., in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Yang, W., Y. Wang, and G. Mori (2010), Recognizing human actions from still images with latent poses, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Y., and D. Ramanan (2011), Articulated pose estimation using flexible mixtures of parts, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Yang, Y., S. Baker, A. Kannan, and D. Ramanan (2012), Recognizing proxemics in personal photos, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

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

Yao, B., and L. Fei-Fei (2010), Modeling mutual context of object and human pose in human-object interaction activities, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Yao, B., X. Jiang, A. Khosla, A. L. Lin, L. J. Guibas, and L. Fei-Fei (2011), Action recognition by learning bases of action attributes and parts, in *Proc. of IEEE Intern. Conf. in Computer Vision (ICCV)*.

Yao, B., G. Bradski, and L. Fei-Fei (2012a), A codebook-free and annotation-free approach for fine-grained image categorization, in *CVPR*.

Yao, J., S. Fidler, and R. Urtasun (2012b), Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Zhang, Z. (2004), Floatboost learning and statistical face detection, *TPAMI*, *26*, 1112–1123.

Zhu, L. L., Y. Chen, Y. Lu, C. Lin, and A. Yuille (2008), Max margin and/or graph learning for parsing the human body, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Zhu, L. L., Y. Chen, A. Yuille, and W. Freeman (2010), Latent hierarchical structural learning for object detection, in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*.

Zitnick, C. L., S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski (2004), High-quality video view interpolation using a layered representation, *ACM Trans. Graph.*, *23*(3), 600–608.