TOWARDS OPEN-UNIVERSE IMAGE PARSING WITH BROAD COVERAGE

Joseph P. Tighe

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Computer Science.

Chapel Hill
2013

Approved by:

Svetlana Lazebnik

Jan-Michael Frahm

Stephen Pizer

Marc Niethammer

Derek Hoiem

ABSTRACT

JOSEPH P. TIGHE: TOWARDS OPEN-UNIVERSE IMAGE PARSING WITH BROAD COVERAGE.
(Under the direction of Svetlana Lazebnik .)

One of the main goals of computer vision is to develop algorithms that allow the computer to interpret an image not as a pattern of colors but as the semantic relationships that make up a real world three-dimensional scene. In this dissertation, I present a system for image parsing, or labeling the regions of an image with their semantic categories, as a means of scene understanding. Most existing image parsing systems use a fixed set of a few hundred hand-labeled images as examples from which they learn how to label image regions, but our world cannot be adequately described with only a few hundred images. A new breed of "open universe" datasets have recently started to emerge. These datasets not only have more images but are constantly expanding, with new images and labels assigned by users on the web.

Here I present a system that is able to both learn from these larger datasets of labeled images and scale as the dataset expands, thus greatly broadening the number of class labels that can correctly be identified in an image. Throughout this work I employ a *retrieval-based* methodology: I first retrieve images similar to the query and then match image regions from this set of retrieved images. My system can assign to each image region multiple forms of meaning: for example, it can simultaneously label the wing of a crow as an animal, crow, wing, and feather. I also broaden the label coverage by using

both region and detector based similarity measures to effectively match a broad range to label types. This work shows the power of retrieval-based systems and the importance of having a diverse set of image cues and interpretations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

In the last decade there has been great success in specific computer vision tasks, such as face recognition, object tracking, camera pose estimation, and 3D reconstruction from multiple images. However, the ultimate goal of computer vision research is to have the computer fully "understand" the visual world, which many believe will require truly intelligent machines, or artificial intelligence, to be invented. Before we can start to talk about how to achieve such a goal, we must define what it means for the computer to "understand" visual data. I use the notion of parsing, borrowed from natural language processing, as a model of "understanding"; thus, this work focuses on the problem of "image parsing."

The term "image parsing" can be interpreted a number of ways. "Image" for our purposes refers to a color image taken with a standard digital camera. Images are represented by a 2D grid of pixels in some color space, typically RGB (red, green, blue). More generally an "image" can vary not only by dimension (a 3D image that represents 3D space, a 2D video where the 3rd dimension is time, or even a 3D video where time is the 4th dimension) but also by modality (medical images such as ultrasound or magnetic-resonance imaging, or hyper-spectral imaging).

The term "parsing" typically refers to the process of breaking a string of symbols into semantically meaningful parts (words) and then deriving a meaning from the string by relating the semantic parts to one another. To parse an image then, we must determine

what the semantically meaningful parts are and relate them to one another. Again there are a wide range of choices that can be made here. Image parts can be defined either as pixels or as regions (groups of pixels) in the image. Then some semantic meaning must be assigned to each image part. Typically a single semantic label is assigned, but labels of various types may be used. For example, an image part could be assigned both a class label, such as "bird," and a material label, such as "feather." Finally, the parts should be related to each other in some manner. The relationship between the parts may be simple; in a 2D image, for example, the parts above, below, and next to one another could simply be connected to form this relationship. However, more complex relationships can also be considered, such as hierarchies, depth ordering, or a full 3D environment.

In this work I define image parsing using the following criteria:

1. Image parts are defined as either single pixels or regions (groups of pixels) in the image.

2. Image parts are assigned different forms of semantic meaning: a semantic label representing an object category as well as other label modalities, such as material type or surface orientation.

3. Image parts are related in 2D by examining the likelihood for two classes to share a boundary with one another.

I use this definition and focus my efforts on designing a simple and scalable system that parses images from a diverse set of scenes with a broad range of label types.

A common approach to building an image parsing system as I have defined it is to train

a local appearance model from a corpus of labeled images (known as the training set) to return a score for each label at each image part in a query (test) image. Such a model often takes the form of a classifier. Then the scores returned by the local model are plugged into a global contextual model, such as a Markov random field (MRF) or conditional random field (CRF), to infer a global labeling for the entire test image. This approach works well when the training set is small and fixed, but it can be computationally infeasible as the training set size increases. Recently a new generation of image parsing datasets has emerged; these datasets have no pre-defined set of class labels and are constantly expanding as people upload new photos or add annotations to current ones. I refer to these new datasets as "open-universe" and the traditional small, fixed datasets as "closed-universe". An example of an open-universe dataset is LabelMe (Russell et al., 2008), which consists of complex, real-world scene images that have been segmented and labeled by multiple users (sometimes incompletely or noisily). To cope with such datasets, vision algorithms must have much faster training and testing times, and they must make it easy to continuously update the visual models with new classes and/or new images.

The main contribution of this thesis is an image parsing system that is designed to be simple and scalable, which can serve as a baseline as more researchers start working on open-universe datasets. My approach provides a rich form of scene understanding that infers multiple types of labels with broad coverage of classes.

1.1 Outline of Contributions

My work on image parsing focuses on systems that can handle large scale data sets. In Chapter 3, I discuss my core parsing system that uses a retrieval-based methodology to assign a single semantic label to each region in the image (Tighe and Lazebnik, 2010, 2013b). An overview of my core parsing system is show in Figure 1.1. The motivation behind this system is to provide a nonparametric solution to image parsing that is as straightforward and efficient as possible and that relies only on operations that can easily be scaled to large image collections and sets of labels. My system requires no training (just some basic computation of dataset statistics) and uses a retrieval set of scenes whose content is used to interpret the test image. While the idea of a retrieval set has been used for various tasks, such as scene completion (Hays and Efros, 2007) and even image parsing (Liu et al., 2011a), my system is the first to use the retrieval set to transfer labels at the level of *superpixels* (Ren and Malik, 2003), or coherent image regions produced by a bottom-up segmentation method. The label transfer is accomplished with a fast and simple nearest-neighbor search algorithm, and it allows for more variation between the layout of the test image and the images in the retrieval set.

The system relates the neighboring superpixels by the label co-occurrences in the training set and performs inference using a Markov Random Field (MRF). I then extend this system to label regions simultaneously with two label types: semantic and geometric classes. Here, semantic classes refer to basic-level category names, such as grass, mountain, car, and building, while geometric classes are the surface orientations from Hoiem

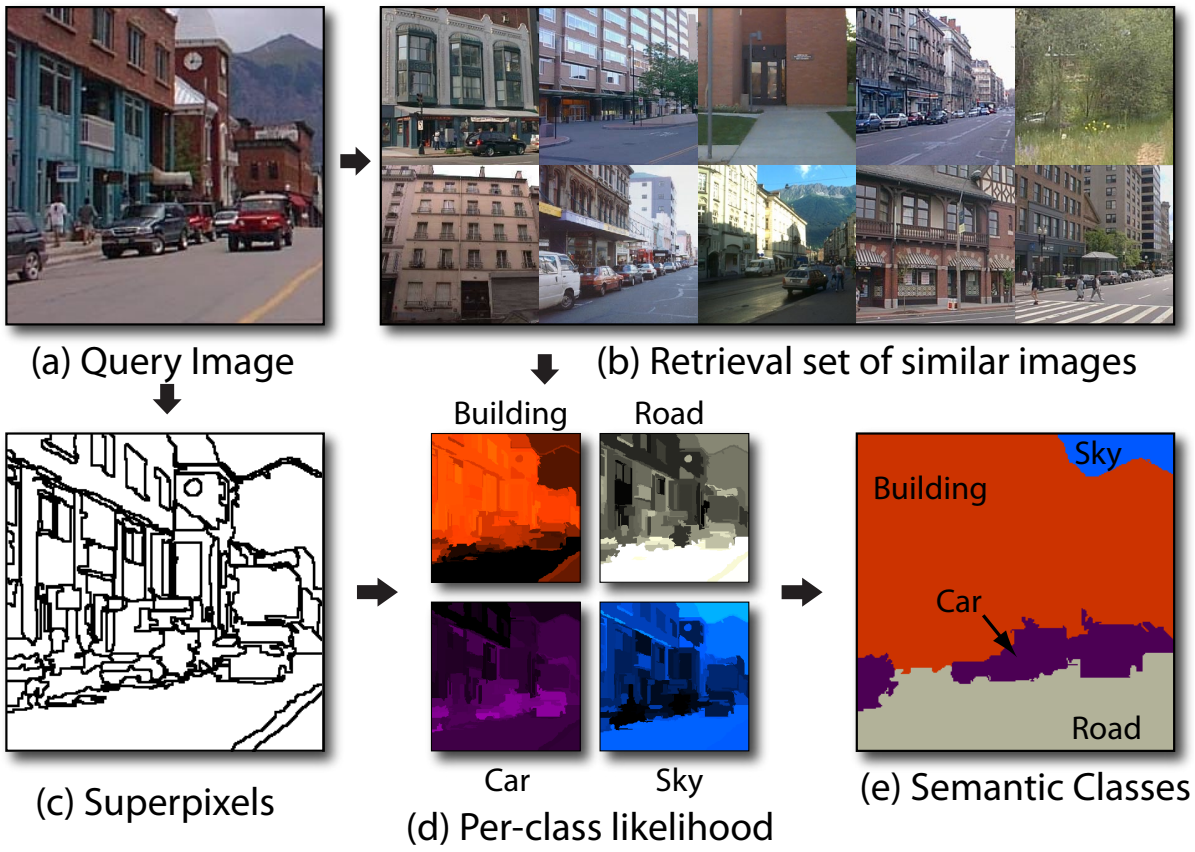

Figure 1.1: Overview of my core image parsing system, which is described in Chapter 3. Given a query image (a): retrieve similar images (b) from the dataset. Next, divide the query into superpixels (c) and compute a per-superpixel score (d) for each class based on matches from the retrieval set. These scores are smoothed with a MRF model to give a dense labeling of the query image (e).

et al. (2007): horizontal, vertical, and sky. Gould et al. (2009) performed a similar simultaneous labeling but did not show that it improved classifications accuracy. I am the first to show performing semantic and geometric labeling simultaneously to be a significant form of context, with each label type correcting errors in the other.

In Chapter 4, I extend my core system to work on an arbitrary number of label types with arbitrary relationships (Tighe and Lazebnik, 2011). The geometric and semantic label types represent a strict hierarchy, but more complex relationships exist. For exam-

ple, a car can be made of glass, painted metal, or rubber, but these materials are not exclusive to cars. My multi-level parsing system learns these many-to-many relationships in a novel way from simple dataset statistics. I then propose an novel, efficient multi-level inference system to infer labels simultaneously across multiple label types.

Finally, in Chapter 5, I return to the problem of assigning a single semantic label to each pixel but focus on achieving broader coverage of class labels—the ability to recognize hundreds or thousands of object classes that commonly occur in everyday street scenes and indoor environments. While the parsing system I describe in Chapter 3 performs quite well, it tends to be more accurate for "stuff" classes (Adelson, 2001) that are characterized by local appearance rather than overall shape – classes such as road, sky, tree, and building. To improve performance on "thing" classes such as car, cat, person, and vase, I incorporate the *per-exemplar detectors* of Malisiewicz et al. (2011) that model the overall object shape with my region-based system. Detectors are trained on individual object instances, and the same retrieval-based methodology is used to select which exemplar detectors are run on a query image. While detectors have been shown to increase the accuracy for parsing systems (see section 2.2 for examples), my system is the first to transfer detection masks rather than bounding boxes, providing greater pixel level accuracy, while maintaining its open-universe compatibility. By incorporating detectors, this parsing system greatly increases the coverage of correctly classified classes.

In summary the novel contributions presented in this thesis are as follows:

1. An efficient, retrieval-based parsing system capable of working with open-universe datasets (Chapter 3);

2. A method for joint inference of both semantic and geometric labels that is shown to improve parsing performance for both label types (Section 3.1.5);

3. A framework to learn relationships and simultaneously infer labels across multi-label sets that have many-to-many relationships (Chapter 4);

4. A parsing system that incorporates detectors in an elegant and scalable manner by transferring object masks from per-exemplar detectors (Chapter 5).

CHAPTER 2: RELATED WORK AND MY WAY FORWARD

2.1 Early work using grammars

Early vision researchers attempted to build image parsing systems that were based on grammars. This work, known as *syntactic pattern recognition*, was an active area of research from the late 1970s to early 1980s (Fu and Albus, 1982). Syntactic approaches were often built on a block world (Roberts, 1965), depicted in Figure 2.1(a). The parsing problem in this scenario is to first break an image into regions representing the individual faces of the blocks in the scene and then to relate the faces to each other using a formal grammar. The grammar used by Fu and Albus (1982) is depicted in Figure 2.1(b). The vertical edges represent a scene decomposition from the full scene down to the individual block faces. Other relationships, such as support between objects and surfaces as well as face adjacency, could also be represented using horizontal edges.

Other work at the time used similar grammar-based approaches on natural scenes (Ohta et al., 1978; Hanson and Riseman, 1978; Ohta, 1985). Unfortunately, accurately splitting a natural scene into its rudimentary parts proved to be very challenging, and without accurate segmentation the parsing grammar was not very effective. Generating robust label predictions in the absence of powerful appearance descriptors or statistical learning techniques also presented insurmountable difficulties. In the mid-1980s, this line of research mostly stopped as researchers focused on sub-problems like segmentation and

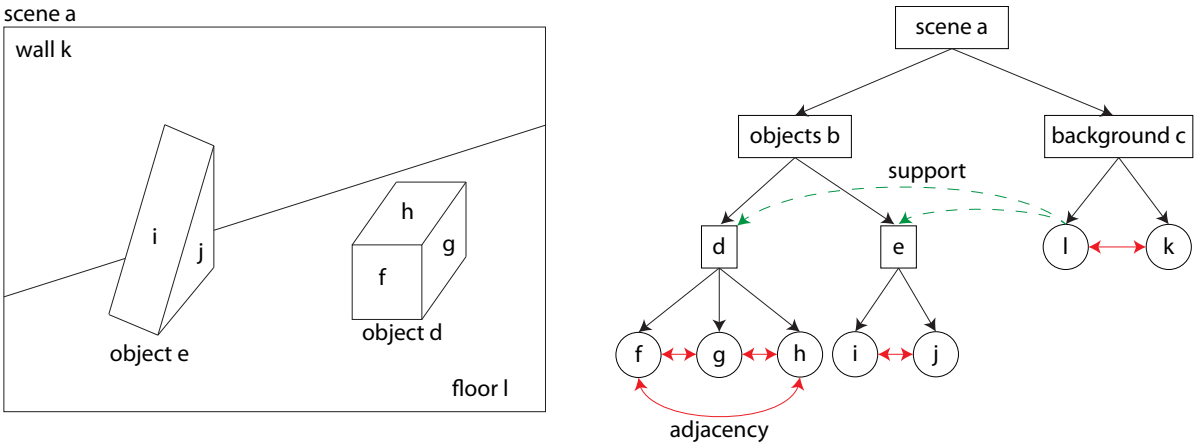

Figure 2.1: The parsing grammar tree for a block world scene as proposed by (Fu and Albus, 1982). Circles are the faces of the blocks in the scene, squares represent block objects, and rectangles larger groupings. Horizontal connections represent relationships such as face adjacency or object support, while vertical connections represent the scene decomposition in to object and parts.

feature extraction that were required before accurate parsing could be feasible.

In the early 2000s, as computer vision and machine learning techniques began to effectively tackle some of these simpler tasks, researchers again started to look at the problem of grammar-based image parsing (Zhu and Mumford, 2006; Zhao and Zhu, 2011; Socher et al., 2011). Tu et al. (2005) even proposed a system that explicitly combined many of the advances developed since the grammar-based image parsing was abandoned in the late 1980s, including segmentation, detection, feature extraction, and classification.

Much of the work on grammar-based parsing involves learning or encoding and then enforcing these grammatical relationships, which can be difficult if the grammatical structure is not known or there are too few samples for each class, which is common with open-universe datasets. In this thesis, I focus first on labeling the regions of the image and only learn simple, pairwise relationships with soft constraints. To this end I model the image parsing problem using a random field model, rather than a grammar-based

model.

2.2 Image parsing methods based on random fields

A common framework emerged in the early 2000s for image parsing with Markov random fields (MRFs) or conditional random fields (CRFs). Most systems roughly followed the same pipeline, borrowed from other popular tasks at the time, such as image restoration (Geman and Geman, 1984):

1. For each pixel of each image in the training set extract features from a patch centered around that pixel.

2. Given this set of features and ground truth labels, train a local model to produce a compatibility score for each feature and each label.

3. Run the trained classifier on a query image and use the output as the unary term of an MRF or CRF.

4. Define a smoothing prior as the binary term of the MRF, where typically a 4- or 8-connected graph is defined over the pixels of the image, or optionally train parameter of the MRF or CRF.

5. Perform maximum a posteriori (MAP) inference on the MRF.

Following this pipeline, He et al. (2004) expanded the domain of the classifier to include both region and global image classification. Shotton et al. (2008) proposed a

parsing system that used boosted classifiers to represent the local spatial layout of textures corresponding to different classes. It also incorporated a per-image color model as well as location priors to improve the discriminative power of the local model. Later, Shotton et al. (2009) showed how to adapt this technique to random forest classifiers that were able to parse a scene in real time. More recently, convolutional neural nets have also been shown to achieve real-time speed but with higher accuracy and without requiring features to be engineered by hand (Farabet et al., 2012).

Context

Going beyond simple smoothing in the MRF model, context emerged as an important factor. The idea was to learn contextual relationships; for instance, cars are supported by the road, and bikes do not float in the sky. Because these relationships require understanding the 3D structure of the scene, they are hard to encode, so systems often settled for simpler relationships. Rabinovich et al. (2007) learn contextual relationships of the dataset, like boats occur with water and cars with roads, by simple co-occurrence statistics in the training set. These relationships are then incorporated into a CRF as penalty terms. This trend continues with systems using multiple forms of context based on co-occurrence, location, and appearance (Galleguillos et al., 2008, 2010). Ladický et al. (2010b) even showed how to efficiently incorporate a global context penalty, i.e., penalizing unlikely pairs of labels from being assigned anywhere in a image.

The "Auto-context" work of Tu (2008) avoided using a heuristic by learning a cascade of classifiers, where the output of the classifier from one round was sparsely sampled

around the pixel being classified and fed as the input to the classifier for the next round. The classifier at each round would choose which sparse locations were informative for the classification task and thus learn positional context and smoothing. Lazebnik and Raginsky (2009) presented work in a similar vein, learning contextual smoothing directly from the images.

In Chapter 3 I explore simple contextual constraints based on label co-occurrence counts. I use a pairwise term in my MRF formulation to model simple "next-to" relationships between labels. This term penalizes classes being assigned to adjacent regions if they are rarely or never seen next to one another.

Region-based approaches

As bottom-up segmentation methods became more accurate and efficient, researchers started to use these methods to generate an oversegmentation of the image. Then the regions from this oversegmentation were used as the rudimentary parts instead of individual pixels . I refer to these methods as region-based methods. Since the number of regions, or superpixels, is much smaller than the total number of image pixels, more computationally complex inference methods could be used. Hoiem et al. (2007) tested multiple segmentations to find the most feasible configuration. Malisiewicz and Efros (2008) learned a separate distance function for each region in the training set, which would be computationally infeasible if using pixels. Regions were not only used for efficiency; Ladický et al. (2009) also combined pixel and region classifier outputs in a hierarchal CRF to generate more plausible object boundaries. Because the regions were generated by advanced

bottom-up segmentation, they tended to have more natural boundaries than the ones generated by simple smoothing heuristics in an MRF.

My core parsing system (described in Chapter 3) is a region-based method. I have found that not only does matching regions instead of individual pixels greatly reduce the computational complexity but matching regions improves the classification accuracy versus matching smaller patches around each pixel.

Three-dimensional scene interpretation

If our data is in the form of image sequences taken by a video camera, structure from motion techniques can be used to estimate the 3D geometry of the scene, and this geometry can be used to help in image parsing. Brostow et al. (2008) compute a sparse point cloud from a video sequence and then extract features based on both this point cloud and on appearance features derived from the color frames. They demonstrate that spatial features can provide complementary cues to appearance-based features, boosting the parsing accuracy. Sturgess et al. (2009) continue this line of research with a better inference method, and Zhang et al. (2010) demonstrate that dense depth maps can generate even more powerful cues. With Microsoft Kinect popularizing depth sensors, systems started to extract additional features from dense depth maps directly produced by such devices (Silberman and Fergus, 2011). Silberman et al. (2012) use the depth cues not only to boost their parsing system's pixel level performance but also to infer support relationships in 3D.

Other image parsing methods attempt to reason in the scene's 3D space rather than

just the 2D image plane. Work by Hoiem et al. (2005, 2007) estimates both the 3D orientations of surfaces in an outdoor scene and the basic support relationships. Gupta et al. (2010) returns to the 3D block worlds of Roberts (1965), but instead of working in a simplistic world built of only wooden blocks, they decompose an outdoor scene into block primitives and infer 3 dimensional relationships between these primitives. A similar line of research focuses on indoor scenes. Hedau et al. (2010) estimate a 3D box that defines the gross structure of a room. Lee et al. (2010) build on this notion, adding volumes for objects that occupy space in the estimated 3D box, while Gupta et al. (2011) predict both likely objects with which people could interact and likely positions a person would take in the room.

In this work, I do not take advantage of 3D geometric cues, though the parsing system of Chapter 3 would make it straightforward to add depth features if they were available, say, from RGBD images.

Nonparametric approaches

Recently, a few researchers have begun advocating nonparametric, data-driven approaches suitable for open-universe datasets (Hays and Efros, 2008; Torralba et al., 2008; Liu et al., 2011a,b). Such approaches avoid training altogether. Instead, for each new test image, they try to retrieve the most similar training images and transfer the desired information from the training images to the query. Liu et al. (2011b) have proposed a nonparametric label transfer method based on estimating "SIFT flow," i.e., a dense deformation field between images. Unfortunately, the optimization problem for finding the SIFT flow is

fairly complex and expensive to solve. Moreover, matching scenes by estimating a dense per-pixel flow field may not necessarily match our intuitive understanding of scenes as collections of discrete objects defined by their spatial support and class identity.

In Chapter 3 I describe my nonparametric parsing system, the base system for all of the work presented herein. My system leverages the concept of a retrieval set to efficiently and accurately parse scenes while easily scaling with an expanding open-universe dataset. This system has already been extended by a number of other researchers (Eigen and Fergus, 2012; Singh and Košecká, 2013).

Multiple labelings

Most existing image parsing systems (He et al., 2004; Liu et al., 2011a; Rabinovich et al., 2007; Shotton et al., 2009) perform a single-level image labeling according to basic-level category names. Coarser labelings, such as natural/manmade, are sometimes considered, as in Kumar and Hebert (2006). Hoiem et al. (2007) were the first to propose geometric labeling. Having a geometric label assigned to each pixel is valuable as it enables tasks like single-view reconstruction. Gould et al. (2009) have introduced the idea of using two different label types; they assign one geometric and one semantic label to each pixel. I also investigate geometric/semantic context in a similar manner. Namely, for each superpixel in the image, I simultaneously estimate two different label types—a *semantic* label (e.g., building, car, person) and a *geometric* label (sky, ground, or vertical surface)—while ensuring the consistency of both labels assigned to the same region (e.g., a building has to be vertical, a road horizontal, and so on). My experiments show that enforcing this

coherence improves the performance of both labeling tasks.

In Chapter 4 I generalize this idea to handle an arbitrary number of label types. There has been a lot of interest in visual representations that allow for richer forms of image understanding by incorporating context (Divvala et al., 2009; Rabinovich et al., 2007), geometry (Gould et al., 2009; Gupta et al., 2010; Hoiem et al., 2007), attributes (Farhadi et al., 2010; Kumar et al., 2011; Lampert et al., 2009), hierarchies (Deng et al., 2010; Griffin and Perona, 2008; Marszalek and Schmid, 2007), and language (Gupta and Davis, 2008; Kulkarni et al., 2013; Ordonez et al., 2011). My work follows in this vein by incorporating a new type of semantic cue: the consistency between different types of labels for the same region. In my framework, relationships between two different types of labels may be hierarchical (e.g., a car is a vehicle) or many-to-many (e.g., a wheel may belong to multiple types of vehicles, while a vehicle may have many other parts besides a wheel). I show how to formulate the inference problem to enforce agreement between different types of labels. My results show that simultaneous multi-level inference performs better than treating each label set in isolation.

Parsing with detectors

As discussed in the introduction, to cover a broader range of labels one must focus on correctly classifying the "thing" classes—people, cars, dogs, mailboxes, vases, and stop signs. To improve performance on "things," a few recent image parsing approaches (Arbelaez et al., 2012; Floros et al., 2011; Guo and Hoiem, 2012; Heitz and Koller, 2008; Ladický et al., 2010b) have attempted to incorporate sliding window detectors. Many

of these approaches rely on detectors like histogram of oriented gradients (HOG) templates (Dalal and Triggs, 2005) and deformable part-based models (DPMs) (Felzenszwalb et al., 2008), which produce only bounding box hypotheses. However, attempting to infer a pixel-level segmentation from a bounding box is a complex and error-prone process. More sophisticated detection frameworks like implicit shape models (Leibe et al., 2008) and poselets (Bourdev and Malik, 2009) provide a better way to do per-pixel reasoning, but they tend to require a lot of extensively annotated positive training examples. Ladický et al. (2010b) overcame this limitation by inferring a mask from a bounding box detection using GrabCut (Rother et al., 2004). This automatic segmentation step can sometimes fail; moreover, it does not leverage the learned detection model, only the final bounding box. Floros et al. (2011) extended the work of Ladický et al. (2010b) by replacing the GrabCut step with the Implicit Shape Model top-down segmentation system (Leibe et al., 2008), improving their bounding box to segment conversion but failing to achieve state-of-the-art performance. Arbelaez et al. (2012) leverage poselet detectors that are trained on semantically meaningful parts and have a fairly stable average segmentation mask. They directly transfer the mean poselet masks and thus avoid the automatic segmentation step. Guo and Hoiem (2012) do not predict a mask from a bounding box but instead use auto-context (Tu, 2008) to directly incorporate the detector responses into their pixel-level parsing system. None of these schemes are well suited for handling large numbers of sparsely-sampled classes with high intra-class variation.

Instead, in Chapter 5 I build an image parsing system that integrates region-based cues with the promising novel framework of *per-exemplar detectors* or *exemplar-SVMs* (Mal-

isiewicz et al., 2011). Per-exemplar detectors are more appropriate than traditional sliding window detectors for classes with few training samples and wide variability. They also meet my need for pixel-level localization: when a per-exemplar detector fires on a test image, I can take the segmentation mask from the corresponding training exemplar and transfer it into the test image to form a segmentation hypothesis.

In summary, my core parsing system is most closely related to the non-parametric method of Liu et al. (2011b), but my system performs matching at the level of superpixels and simultaneously infers multiple types of labels. My detector-based component is most similar to the work of Ladický et al. (2010b), but my method can scale to hundreds of classes with relatively few training examples and does not rely on automatic segmentation to obtain pixel-level localization.

CHAPTER 3: SCALABLE NONPARAMETRIC IMAGE PARSING WITH SUPERPIXELS

This chapter details my retrieval-based image parsing system, which will be basis for all work presented in this dissertation. Figure 3.1 gives an overview of my system.

My parsing system is based on a *lazy learning* philosophy, meaning that (almost) no training takes place offline; given a test image to be interpreted, my system dynamically selects the training exemplars that appear to be the most relevant and proceeds to transfer labels from them to the query. The following is a summary of the steps taken by the system for every query image.

1. Find a retrieval set of images similar to the query image (Section 3.1.1, Figure 3.1b).

2. Segment the query image into superpixels and compute feature vectors for each superpixel (Section 3.1.2, Figure 3.1c).

3. For each superpixel and each feature type, find the nearest-neighbor superpixels in the retrieval set according to that feature. Compute a likelihood score for each class based on the superpixel matches (Section 3.1.3, Figure 3.1d).

4. Use the computed likelihoods together with pairwise co-occurrence energies in a Markov Random Field (MRF) framework to compute a global labeling of the image (Section 3.1.4, Figure 3.1e). Alternatively, with modifications, the MRF framework

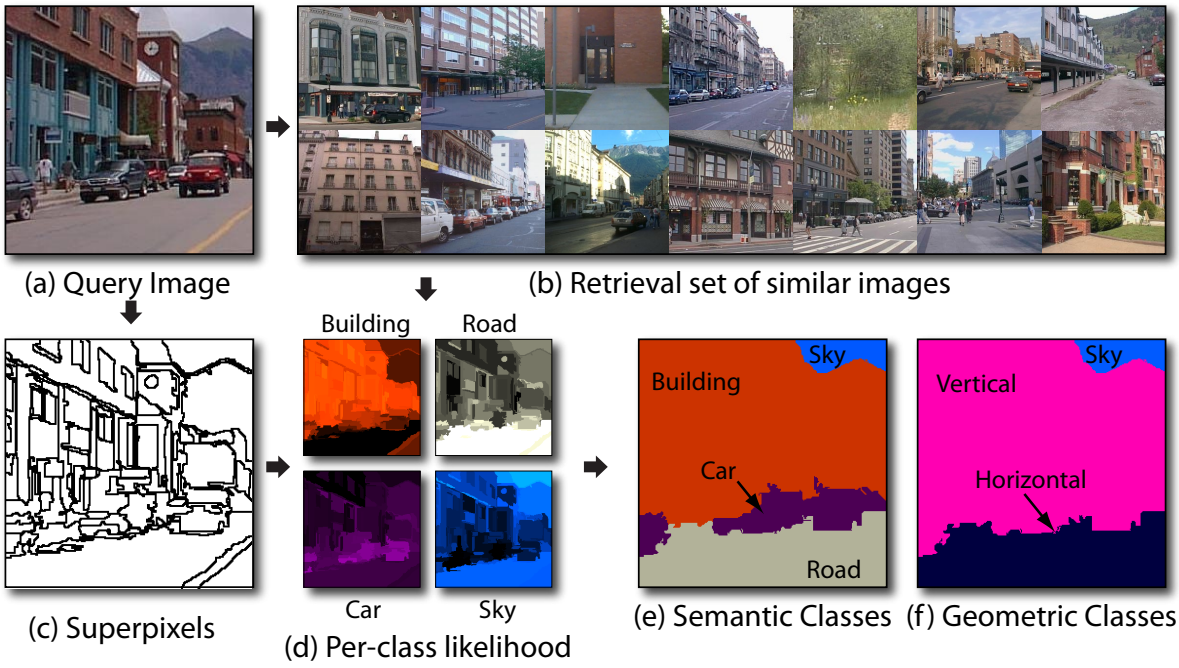

Figure 3.1: System overview. Given a query image (a): retrieve similar images (b) from the dataset using several global features. Next, divide the query into superpixels (c) and compute a per-superpixel likelihood ratio score (d) for each class based on nearest-neighbor superpixel matches from the retrieval set. These scores, in combination with a contextual MRF model, give a dense labeling of the query image in terms of *semantic* (e) and *geometric* (f) labels.

can simultaneously solve for both semantic and geometric class labels (Section 3.1.5, Figure 3.1f).

My system exceeds the results reported in Liu et al. (2011b), which was the state-of-the-art at the time, on a dataset of 2,688 images and 33 labels. Moreover, to demonstrate the scalability of my method, I present results on a subset of the LabelMe (Russell et al., 2008) and SUN (Xiao et al., 2010) datasets totaling 45,676 images and 232 labels. To my knowledge, I was the first to report complete recognition results on a dataset of this size. Thus, one of the contributions of this work was to establish a new benchmark for

large-scale image parsing. Note that unlike other popular benchmarks for image parsing (e.g., Gould et al. (2009); Hoiem et al. (2007); Liu et al. (2011b); Shotton et al. (2009)), my LabelMe+SUN dataset contains both outdoor and indoor images. As will be discussed in Section 3.2.3, indoor imagery currently appears to be much more challenging for general-purpose image parsing systems than outdoor imagery, due in part to the greater diversity of indoor scenes, as well as to the smaller amount of training data available for them.

As another contribution, I extend my parsing approach to video and show how to take advantage of motion cues and temporal consistency to improve performance. Existing video parsing approaches (Brostow et al., 2008; Zhang et al., 2010) use structure from motion to obtain either sparse point clouds or dense depth maps, and extract geometry-based features that can be combined with appearance-based features or used on their own to achieve greater accuracy. I take a simpler approach and only use motion cues to segment the video into temporally consistent regions (Grundmann et al., 2010), or *supervoxels*. This helps to better separate moving objects from one another especially when there is no high-contrast edge between them. My results in Section 3.3 show that the incorporation of motion cues from video can significantly help parsing performance even without the explicit reconstruction of scene geometry. This work was originally published in Tighe and Lazebnik (2010) and Tighe and Lazebnik (2013b).

3.1 System Description

This section presents the details of all the components of my parsing system.

3.1.1 Retrieval Set

Similarly to several other data-driven methods (Hays and Efros, 2008; Liu et al., 2011b,a; Russell et al., 2007), the first step in parsing a query test image is to find a relatively small *retrieval set* of training images that will serve as the source of candidate superpixel-level matches. This is done not only for computational efficiency, but also to provide scene-level context for the subsequent superpixel matching step. A good retrieval set will contain images that have similar scene types, objects, and spatial layouts to the query image. In the attempt to indirectly capture this kind of similarity, three types of global image features are used (Table 3.1(a)): spatial pyramid (Lazebnik et al., 2006), gist (Oliva and Torralba, 2006), and color histogram. For each feature type, all training images are ranked in increasing order of Euclidean distance from the query. Then the minimum of the per-feature ranks is taken to get a single ranking for each image, and the top-ranking K images are used as the retrieval set (a typical value of K in my experiments is 200). Empirically, this method gives an improvement of 1-2% over other schemes, such as simply averaging the ranks. Intuitively, taking the best scene matches from each of the global descriptors leads to better superpixel-based matches for region-based features that capture similar types of cues as the global features (Table 3.1b).

I examine the contributions of different global features and the effect of changing the retrieval set size K in the experiments of section 3.2.3.

(a) Global features for retrieval set computation (Section 3.1.1)		
Type	Name	Dimension
Global	Spatial pyramid (3 levels, SIFT dictionary of size 200)	4200
	Gist (3-channel RGB, 3 scales with 8, 8, & 4 orientations)	960
	Color histogram (3-channel RGB, 8 bins per channel)	24
(b) Superpixel features (Section 3.1.2)		
Shape	Mask of superpixel shape over its bounding box (8×8)	64
	Relative bounding box width/height	2
	Superpixel area relative to the area of the image	1
Location	Mask of superpixel shape over the image	64
	Top height of bounding box relative to image height	1
Texture	Texton histogram, dilated by 10 pix texton histogram	100×2
	Quantized SIFT histogram, dilated by 10 pix.	100×2
	Left/right/top/bottom boundary quantized SIFT hist.	100×4
Color	RGB color mean and std. dev.	3×2
	Color hist. (RGB, 11 bins per channel), dilated by 10 pix.	33×2
Appearance	Color thumbnail (8×8)	192
	Masked color thumbnail	192
	Grayscale gist over superpixel bounding box	320

Table 3.1: A complete list of features used in my system

3.1.2 Superpixel Features

I wish to label the query image based on the content of the retrieval set, but assigning labels on a per-pixel basis as in He et al. (2004) and Liu et al. (2011a,b) tends to be too inefficient. Instead, like Hoiem et al. (2007), Malisiewicz and Efros (2008) and Rabinovich et al. (2007), I choose to assign labels to superpixels, or regions produced by bottom-up segmentation. This not only reduces the complexity of the problem, but also gives better spatial support for aggregating features that could belong to a single object than, say, fixed-size square windows centered on every pixel in the image. Superpixels are computed by using the fast graph-based segmentation algorithm of Felzenszwalb and Huttenlocher

(2004) [1] and describe their appearance using 20 different features similar to those of Malisiewicz and Efros (2008), with some modifications and additions. A complete list of the features is given in Table 3.1(b). In particular, the system computes histograms of textons[2] and dense SIFT descriptors over the superpixel region, as well as a version of that region dilated by 10 pixels. For SIFT features, which are more powerful than textons, I have found it useful to compute left, right, top, and bottom boundary histograms. To do this, I find the boundary region as the difference between the superpixel dilated and eroded by 5 pixels, and then obtain the left/right/top/bottom parts of the boundary by cutting it with an "X" drawn over the superpixel bounding box. All of the features are computed for each superpixel in the training set and stored together with their class labels. A class label is assigned to a training superpixel if 50% or more of the superpixel overlaps with a ground truth segment mask of that label.

3.1.3 Local Superpixel Labeling

Having segmented the test image and extracted the features of all its superpixels, the system computes a log likelihood ratio score for each test superpixel (s_i) and each class (c) that is present in the retrieval set. Making the Naive Bayes assumption that features

[1] I set $K = 200$ and $\sigma = .8$

[2] Code: http://www.robots.ox.ac.uk/~vgg/research/texclass /filters.html

(f_i^k) are independent of each other given the class, the log likelihood ratio is defined as

$$\begin{aligned} L(s_i, c) = \log \frac{P(s_i|c)}{P(s_i|\bar{c})} &= \log \prod_k \frac{P(f_i^k|c)}{P(f_i^k|\bar{c})} \\ &= \sum_k \log \frac{P(f_i^k|c)}{P(f_i^k|\bar{c})}\,, \end{aligned} \tag{3.1}$$

where $\bar{c}$ is the set of all classes excluding c. Each likelihood ratio $P(f_i^k|c)/P(f_i^k|\bar{c})$ is computed with the help of nonparametric density estimates of features from the required class(es) in the neighborhood of f_i^k. Specifically, let $\mathcal{D}$ denote the set of all superpixels in the training set, and $\mathcal{N}_i^k$ denote the set of all superpixels in the retrieval set whose kth feature distance from f_i^k is below a fixed threshold t_k. Then we have

$$\begin{aligned} \frac{P(f_i^k \mid c)}{P(f_i^k \mid \bar{c})} &= \frac{(n(c, \mathcal{N}_i^k) + \epsilon)/n(c, \mathcal{D})}{(n(\bar{c}, \mathcal{N}_i^k) + \epsilon)/n(\bar{c}, \mathcal{D})} \\ &= \frac{n(c, \mathcal{N}_i^k) + \epsilon}{n(\bar{c}, \mathcal{N}_i^k) + \epsilon} \times \frac{n(\bar{c}, \mathcal{D})}{n(c, \mathcal{D})}\,, \end{aligned} \tag{3.2}$$

where $n(c, \mathcal{S})$ is the number of superpixels in set $\mathcal{S}$ with class label c, $n(\bar{c}, \mathcal{S})$ is the number of superpixels in set $\mathcal{S}$ with class label not c, and ϵ is a constant added to prevent zero likelihoods and smooth the counts. In my implementation, I use the ℓ_2 distance for all features, and set each threshold t_k to the median distance to the Tth nearest neighbor for the kth feature type over the dataset. Interestingly, the radius threshold t_k does not seem to have a large influence on the performance of my system, though using a radius instead of taking a fixed number of nearest neighbors was very important to achieve high performance. I use a target number of near neighbors $T = 80$ for all experiments in this chapter. I examine the effect of changing the smoothing constant (ϵ) in section 3.2.3.

The superpixel neighbors $\mathcal{N}_i^k$ are found by linear search through the retrieval set. While approximate nearest neighbor techniques could be used to speed up this search, at my current scale this is not the computational bottleneck of my system as will be discussed in Section 3.2.3.

Note that my density estimates are normalized by counts over the entire training set ($\mathcal{D}$), instead of over the retrieval set, as might perhaps be expected. This is because the majority of feature points removed from the retrieval set are far from the query point f_i^k and nearby points are removed roughly in proportion to the density of that class. Therefore, my likelihood estimates are proportional to the true likelihoods across the whole dataset.

At this point, a labeling of the image can be obtained by simply assigning to each superpixel the class that maximizes equation 3.1. As shown in Table 3.2, the resulting classification rates already come within 2.5% of those of Liu et al. (2011b). I am not aware of any comparably simple scoring scheme reporting such encouraging results for image parsing problems with highly, unequally distributed labels. In particular, my score successfully combines multiple features without relying on learned feature weights or distance functions, which are often held to be crucial for integrating heterogeneous cues to recognize many diverse classes (Malisiewicz and Efros, 2008).

3.1.4 Contextual Inference

Next, I would like to enforce contextual constraints on the image labeling – for example, a labeling that assigns "water" to a superpixel completely surrounded by "sky" is not

very plausible. Many state-of-the-art approaches encode such constraints with the help of conditional random field (CRF) models (Galleguillos and Belongie, 2010; Gould et al., 2009; He et al., 2004; Nowozin et al., 2011; Rabinovich et al., 2007). However, CRFs tend to be very costly both in terms of learning and inference. In keeping with my nonparametric philosophy and emphasis on scalability, I restrict myself to contextual models that require minimal training and that can be solved efficiently. Therefore, I formulate the global image labeling problem as minimization of a standard MRF energy function, which I have found to be comparable to CRF models in terms of accuracy as long as the unary term is strong enough. The MRF is defined over the field of superpixel labels $\mathbf{c} = \{c_i\}$:

$$J(\mathbf{c}) = \sum_{s_i \in SP} E_{\text{data}}(s_i, c_i) + \lambda \sum_{(s_i,s_j)\in A} E_{\text{smooth}}(c_i, c_j)\,, \tag{3.3}$$

where SP is the set of superpixels, A is the set of pairs of adjacent superpixels and λ is the smoothing constant. I define the data term as:

$$E_{\text{data}}(s_i, c_i) = -w_i \sigma(L(s_i, c_i)), \tag{3.4}$$

where $L(s_i, c_i)$ is the likelihood ratio score from equation 3.1, $\sigma(t) = 1/(1+\exp(-\gamma t))$ is the sigmoid function[3] and w_i is the superpixel weight (the size of s_i in pixels divided by

[3]Note that my original system (Tighe and Lazebnik, 2010) did not use the sigmoid nonlinearity, but in my subsequent work (Tighe and Lazebnik, 2011) I found it necessary to successfully perform more complex multi-level inference. Without the sigmoid, extremely negative classifier outputs, which often occur on the more rare and difficult classes, end up dominating the multi-level inference, "converting" correct labels on other levels to incorrect ones. I have also found that the sigmoid is a good way of making the output of the nonparametric classifier comparable to that of other classifiers, for example, boosted decision trees (see Section 3.2.1).

the mean superpixel size). The smoothing term E_{smooth} is defined based on probabilities of label co-occurrence:

$$E_{\text{smooth}}(c_i, c_j) = -\log \frac{(P(c_i|c_j) + P(c_j|c_i))}{2} \times \delta[c_i \neq c_j]\,, \tag{3.5}$$

where $P(c|c')$ is the conditional probability of one superpixel having label c given that its neighbor has label c', estimated by counts from the training set. I use the two conditionals probabilities $P(c|c')$ and $P(c'|c)$ instead of the joint probability $P(c, c')$ because they have better numerical scaling. I average them to obtain a symmetric quantity. Multiplication by $\delta[c_i \neq c_j]$ is necessary to ensure that this energy term is semi-metric as required by graph cut inference (Boykov et al., 2001). Qualitatively, I have found equation 3.5 to produce very intuitive edge penalties. As can be seen from the examples in Figure 3.2, it successfully flags improbable boundaries. Quantitatively, results with equation 3.5 tend to be about 1% more accurate than with the constant Potts penalty $\delta[c_i \neq c_j]$. MRF inference is performed using the efficient graph cut optimization code of Boykov and Kolmogorov (2004); Boykov et al. (2001); Kolmogorov and Zabih (2004). On my large datasets, the resulting labelings improve the accuracy by 2-4% (Tables 3.2 and 3.3).

I have also experimented with a contrast-sensitive per-pixel MRF similar to that of Liu et al. (2011b), but have found that the per-superpixel formulation is faster, and achieves the same per-pixel and per-class performance. One reason for this may be that the per-superpixel MRF makes it easier to converge to a better minimum by flipping labels over larger areas of the image. A per-pixel MRF does however produce more visually pleasing

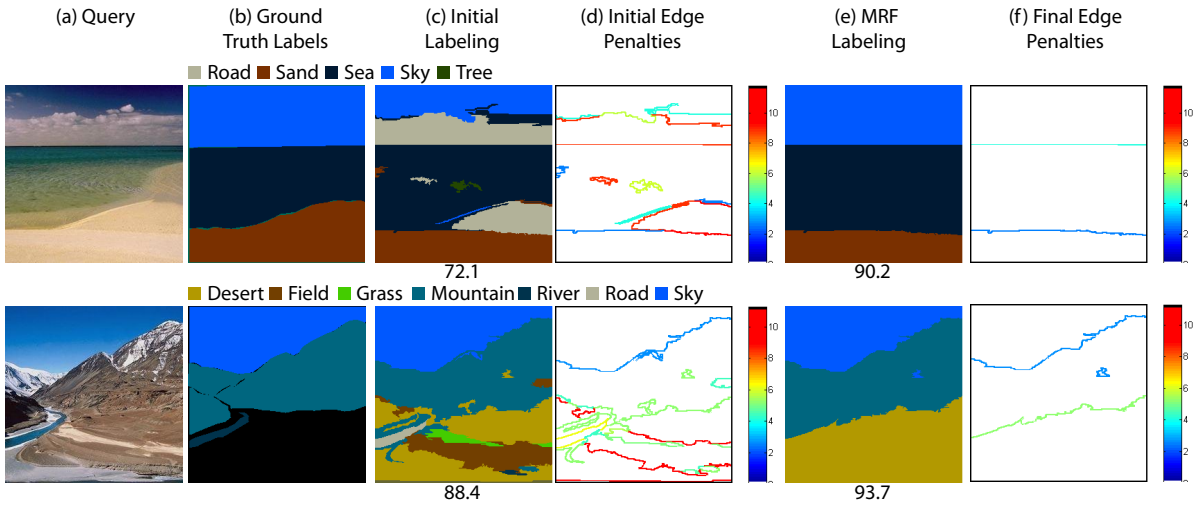

Figure 3.2: Contextual edge penalty before and after the MRF optimization. The top row shows my contextual model successfully flags improbable boundaries between "sea" and "road" and the second row shows it flags "Desert" and "Field" since in the data set only they never occur next to each other.

labelings, but I chose to use the superpixel-based MRF due to its superior speed.

3.1.5 Simultaneous Classification of Semantic and Geometric Classes

To achieve more comprehensive image understanding and to explore a higher-level form of context, I consider the task of simultaneously labeling regions into two types of classes: semantic and geometric (Gould et al., 2009). The notion of parsing an image into geometric primitives was introduced by Hoiem et al. (2007) and shown to be useful for a variety of tasks, such as rough 3D modeling (Hoiem et al., 2005) and object location prediction (Hoiem et al., 2006). Like Hoiem et al. (2007) and Gould et al. (2009), I use three geometric labels – sky, horizontal, and vertical – although the set of semantic labels in my datasets are much larger. In this chapter, I make the assumption that each semantic class is associated with a unique geometric class (e.g., "building" is "vertical,"

"river" is "horizontal," and so on) and specify this mapping by hand. This is a bit restrictive for a few classes (e.g., I force "rock" and "mountain" to be vertical), but for the vast majority of semantic classes, a unique geometric label makes sense. In chapter 4 I generalize this idea and learn the relationship from the data. I jointly solve for the fields of semantic labels ($\mathbf{c}$) and geometric labels ($\mathbf{g}$) by minimizing a cost function that is a simple extension of equation 3.5:

$$H(\mathbf{c}, \mathbf{g}) = J(\mathbf{c}) + J(\mathbf{g}) + \mu \sum_{s_i \in SP} \varphi(c_i, g_i), \tag{3.6}$$

where φ is the term that enforces coherence between the geometric and semantic labels. It is 0 when the semantic class c_i is of the geometric class type g_i and 1 otherwise. The constant μ controls how strictly the coherence is enforced (I use $\mu = \lambda = 1$ in all experiments). Note that it is possible to enforce the semantic/geometric consistency in a hard manner by effectively setting $\mu = \infty$, but I have found that allowing some trade-off produces better results. Equation 3.6 is in a form that can be optimized by the α/β-swap algorithm (Boykov and Kolmogorov, 2004; Boykov et al., 2001; Kolmogorov and Zabih, 2004). The inference takes nearly the same amount of time as for the MRF setup of the previous section. Figure 3.3 shows an example where joint inference over semantic and geometric labels improves the accuracy of the semantic labeling. More generally, as will be shown by the quantitative results of Section 3.2, joint inference tends to improve both labelings simultaneously.

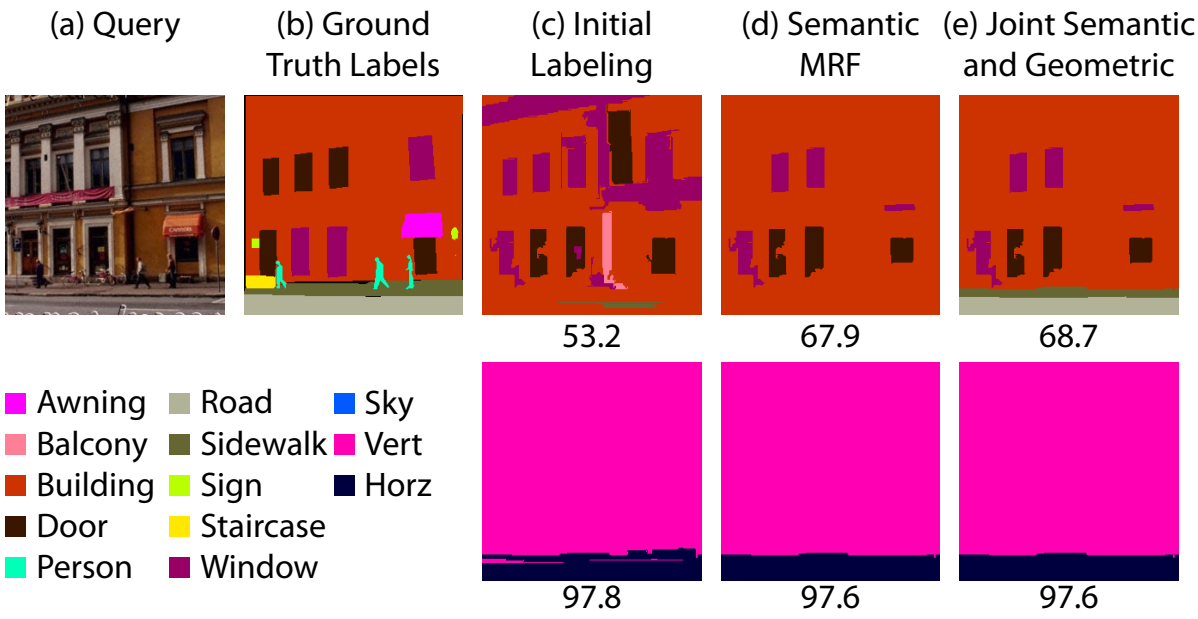

Figure 3.3: In the contextual MRF classification, the road gets replaced by "building," while "horizontal" is correctly classified. By jointly solving for the two kinds of labels, the system manages to recover some of the "road" and "sidewalk" in the semantic labeling. Note also that in this example, my method correctly classifies some of the windows that are mislabeled as doors in the ground truth, and incorrectly but plausibly classifies the windows on the lower level as doors.

3.2 Image Parsing Results

Sections 3.2.1 and 3.2.2 give an overview of results on the two large datasets, SIFT Flow and LM+SUN. State-of-the-art performance is shown in Tables 3.2. Section 3.2.3 gives a thorough evaluation of all the major components of the system. The retrieval set selection is evaluated by looking at the features used (Table 3.4), the size of the retrieval set (Table 3.5), and the scene type in the retrieval set (Figure 3.8,Table 3.7). Then the superpixle classification is evaluated by looking at the segmentation used (Figure 3.9), the features (Figure 3.10), and the classifier (Table 3.8). Finally runtime is broken down by the individual components of the system (Table 3.9)

3.2.1 SIFT Flow Dataset

The first dataset used to test the system, referred to as "SIFT Flow dataset" in the following, comes from Liu et al. (2011b). It is composed of the 2,688 images that have been thoroughly labeled by LabelMe users. Liu et al. (2011b) have split this dataset into 2,488 training images and 200 test images and used synonym correction to obtain 33 semantic labels; this training/test split is used in all experiments.

The frequencies of different labels on this dataset are shown in Figure 3.4(a). It is clear that they are very non-uniform: a few classes like building, mountain, tree, and sky are very common, but there is also a "long tail" of relatively rare classes like person, sign, boat, and bus. To give a fair idea of my system's performance on such unbalanced data, I evaluate accuracy using not only the per-pixel classification rate, which is mainly determined by how well the system can label the few largest classes, but also the unweighted average of the per-pixel rates over each of the classes, which is referred to as the average per-class rate. As will be shown in Table 3.6, a system that classifies all pixels into the top few most common classes would have a relatively high per-pixel rate, but a catastrophically low average per-class rate.

As explained in Section 3.1.5, my system labels each superpixel by a semantic class (the original 33 labels) and a geometric class of sky, horizontal, or vertical. Because the number of geometric classes is small and fixed, I have trained a boosted decision tree (BDT) classifier as in Hoiem et al. (2007) to distinguish between them. I use a tree depth of 8 and train 100 trees for each class. This classifier outputs a likelihood ratio

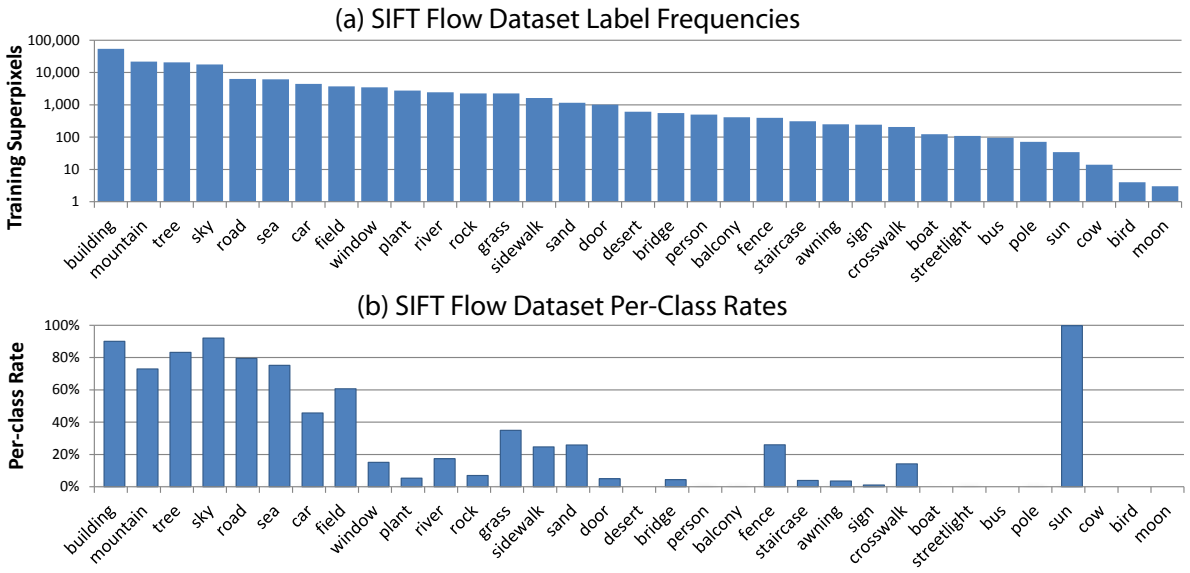

Figure 3.4: Label frequencies for the superpixels in the training set and the classification rate broken down by class for my full system on the SIFT Flow dataset.

score that is comparable to the one produced by my nonparametric scheme (equation 3.1), but that gets about 2% higher accuracy for geometric classification (Section 3.2.3 will present a detailed comparison of nearest-neighbor classifiers and BDT). Apart from this, Local and MRF classification for geometric classes proceeds as described in Sections 3.1.3 and 3.1.4, and I also put the semantic and geometric likelihood ratios into a joint contextual classification framework as described in Section 3.1.5.

Table 3.2 reports per-pixel and average per-class rates for semantic and geometric classification of local superpixel labeling (section 3.1.3), separate semantic and geometric MRF (section 3.1.4), and joint semantic/geometric MRF (section 3.1.5). As compared to the local baseline, the contextual MRF improves overall per-pixel rates on the SIFT Flow dataset by about 2%. The average per-class rate for the MRF drops due to "smoothing away" some of the smaller classes, while joint semantic/geometric MRF improves the

	Semantic	Geometric
Local Labeling	74.1 (30.2)	90.2 (88.7)
MRF	76.2 (29.1)	90.6 (88.9)
Joint	77.0 (30.1)	90.8 (89.2)
Liu et al. (2011b)	76.7	NA
Farabet et al. (2012)	78.5 (29.6)	NA
Eigen and Fergus (2012)	77.1 (32.5)	NA

Table 3.2: Performance on the SIFTFlow dataset for my system and three state-of-the-art approaches. Per-pixel classification rate is listed first, followed by the average per-class rate in parentheses.

results for both per-pixel and average per-class rates. Figure 3.4(b) shows classification rates for the 33 individual classes. Similarly to most other image labeling approaches that do not rely on object detectors, my system gets much weaker performance on "things" (people, cars, signs) than on "stuff" (sky, road, trees).

My final system on the SIFT Flow dataset achieves a classification rate of 77.0%. Thus, it outperformed the state-of-the-art at the time Liu et al. (2011b), who report a rate of 76.7% on the same test set with a more complex pixel-wise MRF (without the pixel-wise MRF, their rate is 66.24%). Liu et al. (2011b) also cite a rate of 82.72% for the top seven object categories; my corresponding rate is 84.7%. Table 3.2 also reports results of two other approaches (Eigen and Fergus, 2012; Farabet et al., 2012) that build on and compare to the earlier version of my system (Tighe and Lazebnik, 2010). Eigen and Fergus (2012) are able to improve on my average per-class rate, while Farabet et al. (2012) are able to improve on the overall rate through the use of more sophisticated learning techniques.

Sample output of my system on several SIFT Flow test images can be seen in Figure

3.5.

3.2.2 LM+SUN Dataset

My second dataset ("LM+SUN" in the following) is derived by combining the SUN dataset (Xiao et al., 2010) with a complete download of LabelMe (Russell et al., 2008) as of July 2011. I culled from this dataset any duplicate images and any images from video surveillance (about 10,000), and use manual synonym correction to obtain 232 labels. This results in 45,676 images of which 21,182 are indoor and 24,494 are outdoor. I split the dataset into 45,176 training images and 500 test images by selecting test images at random that have at least 90% of their pixels labeled and at least 3 unique labels (a total of 13,839 images in the dataset meet this criteria). Apart from its bigger size, the inclusion of indoor images makes this dataset very challenging.

As shown in Figure 3.6(a), the LM+SUN dataset has unbalanced label frequencies just like SIFT Flow. Table 3.3 shows the performance for the three versions of my system (local maximum likelihood labeling, separate semantic and geometric MRF, and joint semantic/geometric MRF) on the entire dataset, as well as on outdoor and indoor images separately. The overall trend is the same as for SIFT Flow: separate MRF inference always increases the overall accuracy over the local baseline though it can sometimes over-smooth, decreasing the average per-class rate. As for joint semantic/geometric inference, it not only gives the highest overall accuracy in all cases, but is also much less prone to over-smoothing.

The final system achieves a classification rate of 54.9% across all scene types (as

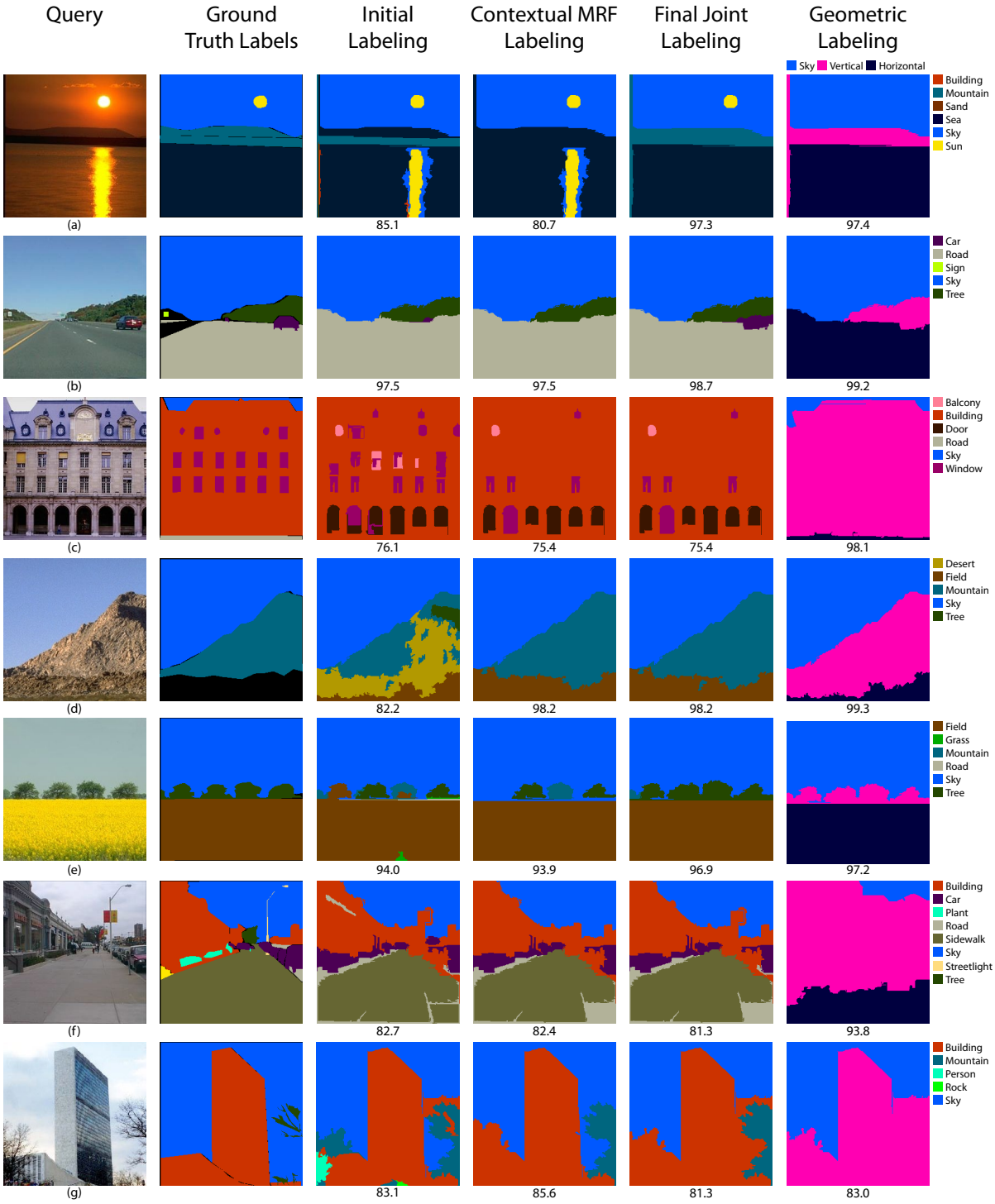

Figure 3.5: Example results from the SIFT Flow test set (best viewed in color). The number under each result image is the percentage of pixels labelled correctly. In (a), joint geometric/semantic inference removes the spurious classification of the sun's reflection in the water. In (c), the system finds some windows (some of which are smoothed away by the MRF) and plausibly classify the arches at the bottom of the building as doors. In (d), "field" and "desert" never co-occur so "field" wins and "desert" is removed.

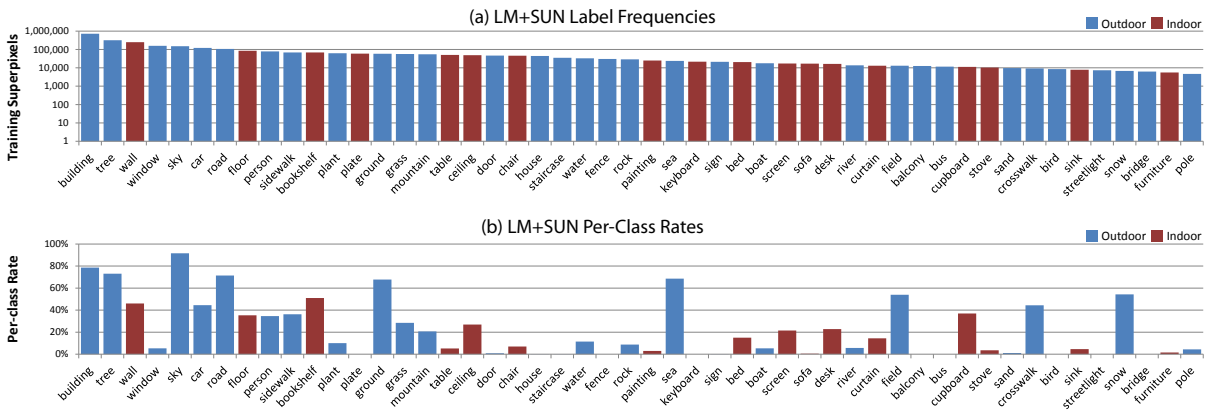

Figure 3.6: Label frequencies for the superpixels in the training set and the classification rate broken down by class for my full system on the LM+SUN dataset. Only the 50 most common classes are shown.

	All		Outdoor		Indoor	
	Semantic	Geometric	Semantic	Geometric	Semantic	Geometric
Local	50.6 (7.1)	79.8 (85.6)	56.7 (7.7)	83.0 (87.5)	27.0 (4.9)	67.8 (74.6)
MRF	54.4 (6.8)	82.6 (86.8)	60.4 (7.6)	85.2 (88.6)	31.2 (4.5)	72.6 (76.1)
Joint	54.9 (7.1)	85.9 (86.8)	60.8 (7.7)	88.3 (89.3)	32.1 (4.8)	76.6 (74.0)

Table 3.3: Performance on the LMSun dataset broken down by outdoor and indoor test images. Per-pixel classification rate is listed first, followed by the average per-class rate in parentheses.

compared to 77% for SIFT Flow); the respective rates for outdoor and indoor images are 60.8% and 32.1%. Figure 3.6(b) gives a breakdown of rates for the 50 most common classes, and Figure 3.7 shows the parsing output on a few example images. It is clear that indoor scenes are currently much more challenging than outdoor ones, due at least in part to their greater diversity and sparser training data. In fact, parsing of indoor scenes has become of great interest; most existing work dealing with indoor scenes uses specialized geometric information and focuses on only a few target classes. For example, Hedau et al. (2009) infer the "box" of a room and then leverage the geometry of that

box to align object detectors to the scene (Hedau et al., 2010). Gupta et al. (2011) infer the possible use of a space rather than directly labeling the objects. There has also been a recent interest in indoor parsing with the help of structured light sensors (Silberman and Fergus, 2011; Janoch et al., 2011; Lai et al., 2011) as a way to combat the ambiguity present in cluttered indoor scenes. It is clear that my system, which relies on generic appearance-based matching, cannot currently achieve very high levels of performance on indoor imagery. However, my reported numbers can serve as a useful baseline for more advanced future approaches. The challenges of indoor classification will be further studied in Section 3.2.3.

3.2.3 Detailed System Evaluation

This section presents a detailed evaluation of various components of my system. Unless otherwise noted, the evaluation is conducted on both the SIFT Flow dataset and LM+SUN, no MRF smoothing is done, and only semantic classification rates are reported.

Retrieval set selection

The initial step of my system is retrieval set selection. Table 3.4 shows the performance of different global features used for this step. Similarly to Hays and Efros (2008), I find that combining global features of complementary descriptive power gives better scene matches. The last line in this table, "Maximum Label Overlap," is meant to be an upper bound on the performance of the retrieval set. Here the retrieval set is found by ranking training

Figure 3.7: Example results from the LM+SUN test set (best viewed in color). For indoor images the geometric label of "sky" corresponds to the semantic label of "ceiling." Examples (a-c) show the best performance the system can achieve on indoor scenes: it gets wall, ceiling, floor and even gets some chairs, bookshelves, beds and desks. Example (d) shows how a retrieval set of mixed indoor and outdoor images can produce incorrect labels. Examples (e-g) show the variety of images the system can work on. In (f), it even corrects the ground truth label "animal" to the more specific class "bison."

Global Descriptor	SIFT Flow	LMSun
Gist (G)	70.8 (29.7)	45.6 (7.0)
Spatial Pyramid (SP)	69.6 (23.1)	47.9 (6.3)
Color Hist. (CH)	66.9 (24.6)	43.5 (5.7)
G + SP	72.3 (27.9)	50.6 (6.9)
G + SP + CH	74.1 (30.2)	50.6 (7.1)
Maximum Label Overlap	80.2 (33.6)	66.0 (13.2)

Table 3.4: Evaluation of global image features for retrieval set generation (retrieval set size 200). "Maximum Label Overlap" is the upper bound that I get by selecting retrieval set images that are the most semantically consistent with the query (see text).

images in terms of the number of pixels their ground truth label maps share with the label map of the query, the logic being that the perfect retrieval set will have images with the correct classes in the correct locations. The big gap in accuracy between this "ideal" retrieval set and the one obtained by global appearance-based matching underscores the shortcomings of global image features in terms of finding scenes *semantically* consistent with the query.

Table 3.5 examines the effect of retrieval set size. Interestingly, using all of the SIFT Flow training set as the retrieval set (last row of Table 3.5) drastically reduces performance. This quantitatively confirms the intuition that the retrieval set is not just a way to limit the computational complexity of superpixel matching, but is also a form of scene-level context. By restricting the superpixel matches to come from a small subset of related scenes, the system can produce a better interpretation of the image. Also note that while on the SIFT Flow dataset the performance degrades once the retrieval set reaches a size greater than 400, the performance on the LM+SUN dataset continues to rise even with a retrieval set size of 1,600: with more than 45,000 images in that

Retrieval Set Size	SIFT Flow	LM+SUN
50	73.0 (32.2)	47.3 (8.1)
100	73.7 (30.1)	48.9 (7.4)
200	74.1 (30.2)	50.6 (7.1)
400	73.0 (28.7)	51.0 (7.6)
800	72.1 (28.1)	51.5 (7.5)
1,600	69.9 (26.2)	51.2 (8.1)
Entire training set	68.4 (23.2)	N/A

Table 3.5: Effect of retrieval set size on local superpixel labeling. Note that the entire LM+SUN training set is too large for my hardware to store in memory.

dataset, there are usually still 1,600 images that are of a sufficiently similar scene type. Thus, it appears that the right retrieval set size depends in a complex way on the size of the dataset and on the distribution of scene types contained in it. Despite this, in all other experiments a retrieval set size of 200 is used for both datasets, primarily for efficiency: the system must read the descriptor data from disk for each query image on the LM+SUN dataset, which becomes prohibitively slow with larger retrieval set sizes. However this experiment does suggest that retrieval set selection is a subtle and crucial step that deserves further study.

While the total number of labels in the LM+SUN datasets is quite high, any single image only contains a small subset of all possible labels. The retrieval set defines not only the set of training images that can be used to interpret the test image, but also the "shortlist" of all possible labels that can appear in the test image. By default, this shortlist is composed of all the classes present in the retrieval set. Table 3.6 examines the effect of restricting these shortlists in various ways. The first row corresponds to the default shortlist (the same one that is used in the experiments of the previous two

Shortlist	SIFTFlow	LM+SUN
Classes in retrieval set	74.1 (30.2)	50.6 (7.1)
10 most common classes	74.9 (21.4)	51.0 (3.1)
Perfect shortlist	81.4 (35.4)	61.7 (11.1)

Table 3.6: Accuracy of local superpixel labeling obtained by restricting the set of possible classes in the test image to different "shortlists" (see text).

sections). To demonstrate the effect of long-tail class frequencies, the second row shows the performance obtained by classifying every superpixel in every test image to the ten most common classes in the dataset. This slightly increases the overall per-pixel rate, but lowers the average per-class rate dramatically. On the other hand, it is worth observing that the average per-class rate can be inflated upwards by good performance on a few very rare classes (e.g., there are only two "suns" in the SIFT Flow test set, and both are correctly classified). Finally, the third row of Table 3.6 shows the results produced by restricting the shortlist to the ground truth labels in the query image, giving an upper bound for the performance of superpixel matching. Just like the "maximum label overlap" retrieval set of Table 3.4, a perfect shortlist "oracle" would give a significant boosts in overall per-pixel rate and average per-class rate on both datasets. This suggests that to further improve system performance, it is important to work on more accurate scene-level label prediction and better scene-level matching for generating the retrieval sets. In fact, I have observed that in many of the unsuccessfully labeled images, incompatible scene classes with strong local support over large regions vie for the interpretation of the image, and neighborhood context, though it may detect the conflict, has no plausible path towards resolving it (Figure 3.7(d) is one example of this).

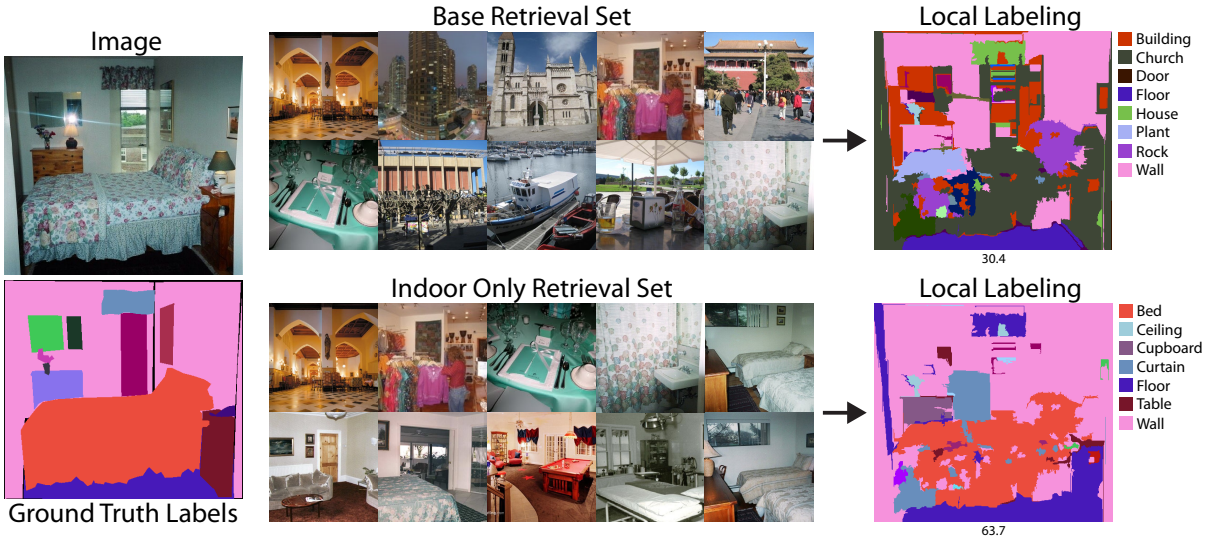

Figure 3.8: Effect of drawing the retrieval set from the entire training set ("base retrieval set") vs. drawing it from the correct scene type ("indoor only retrieval set"). With the base retrieval set, the local labeling result has a mix of indoor (floor, wall) and outdoor (building, rock) classes. On the other hand, if the retrieval set is restricted to consist only of indoor images, the parsing output is much cleaner – in particular, most of the bed is correctly labeled.

The LM+SUN dataset has two obvious sub-classes: indoor and outdoor. However, retrieval set selection based on low-level features does not do a very good job of separating them: for an indoor query image there are often outdoor images in the retrieval set and vice versa (see Figure 3.8 for an illustration). To get an idea of how much this confusion hurts performance, ground truth knowledge is used to force the retrieval set to have only images of the correct scene type – that is, an indoor query image would only be matched against indoor images and likewise for an outdoor one (this is equivalent to splitting the LM+SUN dataset into two separate indoor and outdoor datasets). Table 3.7 (line 2) shows the resulting improvement in overall performance. Most of this gain is with the indoor images as they tend to have retrieval sets with more outdoor images and more

	All	Outdoor	Indoor
Local labeling	50.6 (7.1)	56.7 (7.7)	27.0 (4.9)
Ground truth	54.4 (7.8)	59.1 (8.3)	36.6 (5.6)
Classifier	52.7 (7.3)	57.4 (8.2)	34.4 (5.5)

Table 3.7: Effect of indoor/outdoor separation on the accuracy of local superpixel labeling on LM+SUN. "Local labeling" corresponds to the default system with no separation between outdoor and indoor training images (the numbers are the same as in line 1 of Table 3.3). "Ground truth" uses the ground truth label for the query image to determine if the retrieval set should consist of indoor or outdoor images, while "Classifier" uses a trained indoor/outdoor SVM classifier (see text).

confusion with outdoor classes in general. To get most of this gain without "cheating," an indoor/outdoor classifier is trained using a linear SVM on all the global image features concatenated and normalized by the standard deviation along each dimension. This classifier achieves a rate of 92% on my test set. The last row of Table 3.7 shows the performance of my system when this classifier is used to determine which set of training images to draw the retrieval set from. As expected, the accuracy is somewhere in-between that of "perfect" indoor-outdoor classification and no classification altogether.

Note that performing automatic indoor/outdoor image classification and then using the inferred scene type to constrain the interpretation of the image is conceptually analogous to performing a geometric labeling of the image and using the inferred geometric classes of regions to constrain the semantic classes. In both cases I am taking advantage of the high accuracy that can be achieved on relatively easier two- and three-class problems to improve the accuracy on a harder many-class problem.

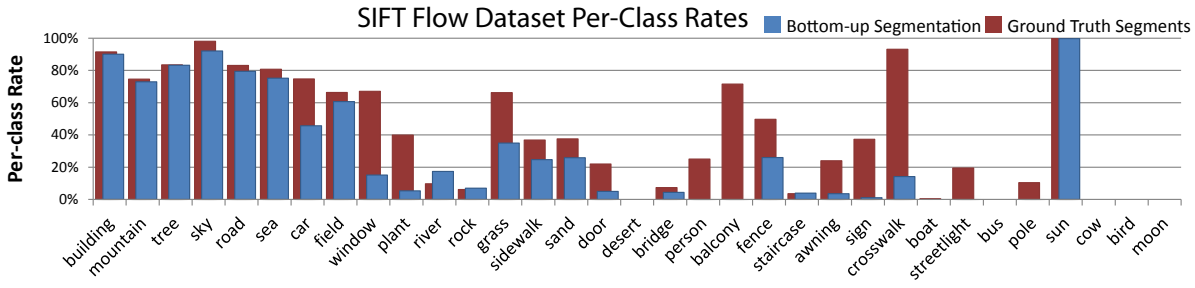

Figure 3.9: Per-class classification rates on the SIFT Flow dataset for superpixels (blue) versus ground truth object polygons (maroon). With the ground truth segmentation, an overall per-pixel rate of 82.3 and average class rate of 46.0 is obtained, showing that even with correct knowledge of object shape, the system still has a hard time classifying many of the classes.

Superpixel classification

After retrieval set selection, the next stage of my system is superpixel classification. One of the most important factors affecting the success of this stage is the quality of the bottom-up segmentation, or how well the spatial support of the superpixels reflects true object boundaries. To see what would happen with perfect segmentation, I use the ground truth object polygons to create segments for the SIFT Flow dataset and then apply my local classification scheme to these segments. Figure 3.9 compares the per-class rates of my system with bottom-up segmentation and to those of ground truth segments. We can see that ground truth segmentation significantly helps with many classes such as car, window, balcony, and crosswalk. However, performance on many of the "thing" classes such as person, boat, and streetlight is still quite poor. Thus, even if one could obtain perfect object segmentation, the task of classifying many of the rarer classes would remain quite challenging.

Next we look at the contributions of the multiple superpixel-level features (refer back

to Table 3.1(b) for a list of these features). Figure 3.10 plots the classification rate of the system on both datasets with superpixel features added consecutively in decreasing order of their contribution to performance. Note that this evaluation is carried out on the test set itself (as opposed to a separate validation set), as my goal is simply to understand the behavior of the chosen representation, not to tune performance or to perform feature selection. The experiment starts with the single superpixel feature that has the highest per-pixel classification rate, and then adds one feature at a time, always choosing the one that gives the largest boost in per-pixel classification rate. At each step the overall and average per-class rates for local, MRF and joint geometric/semantic labeling are shown. One observation is that SIFT histograms constitute three or four of the top ten features selected. The dilated SIFT histogram, which already incorporates some context from the superpixel neighborhood, is the single strongest feature for both datasets, and it effectively makes the non-dilated SIFT histogram redundant. After SIFT Histogram Dilated, the order of the features selected for both datasets is quite different, though Top Height and Mean Color show up in the top five in both cases, confirming that location and color cues add important complementary information to texture. Another observation is that while adding features does sometimes hurt performance, it does so minimally. One could combat this effect by learning feature weights as in Eigen and Fergus (2012) but this would make the system more prone to over-fitting and introduce an off-line learning component that I would like to avoid.

It is also interesting to compare the curves for three versions of my system: local superpixel labeling, separate semantic and geometric MRF, and joint semantic/geometric

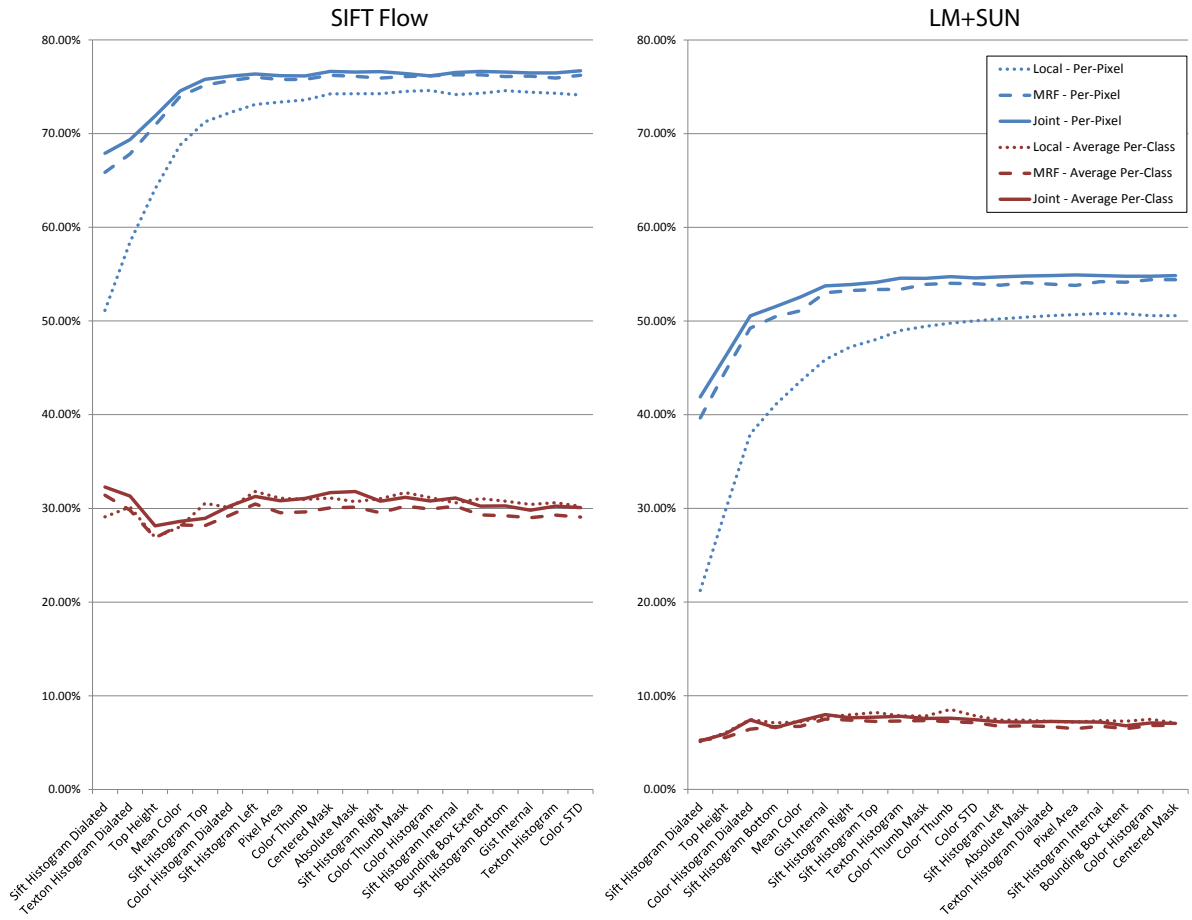

Figure 3.10: The classification rate of my system computed by consecutively adding superpixel features in decreasing order of their contribution. Interestingly, the best features for one dataset are not necessarily the best for the other one. The rates for the MRF and joint solvers are also plotted. Notice that the MRF and joint solver are more effective when the classifier is weaker, and that the joint solver consistently outperforms the MRF for the average per-class rate, correcting for the over-smoothing that occurs due to the MRF.

MRF. Consistent with the results reported in Tables 3.2 and 3.3, the separate MRF tends to lower the average per-class rate due to over-smoothing and then the joint MRF brings it back up. More surprisingly, Figure 3.10 reveals that both of my contextual models have a much greater impact when they are applied on top of a relatively weak local model, i.e., one with fewer features. As more local features are added, the improvements afforded by non-local inference gradually diminish. The important message here is that "local features" and "context" are, to a great degree, interchangeable. For one, many of the features are not truly "local" since they include information from outside the spatial support of the superpixel. But also, it seems that contextual inference can fix relatively few labeling mistakes that cannot just as easily be fixed by a more powerful local model. This is important to keep in mind when critically evaluating contextual models proposed in the literature: a big improvement over a local baseline does not necessarily prove that the proposed form of context is especially powerful – the local features may simply not be strong enough.

Next, Figure 3.11 shows the effect of the smoothing constant ϵ in the likelihood ratio equation (equation 3.2). As noted in Eigen and Fergus (2012), increasing ϵ biases the classifier toward the rarer classes. In turn, this tends to decrease the overall per-pixel rate and increase the average per-class accuracy. I found the value of $\epsilon = 1$ to achieve a good trade-off and use it in all other experiments.

Further, I wish to examine how well my nonparametric scheme is doing compared to offline discriminative learning techniques. To this end, I train boosted decision trees (BDT) for all 33 labels in the SIFT Flow dataset the same way I train them for the

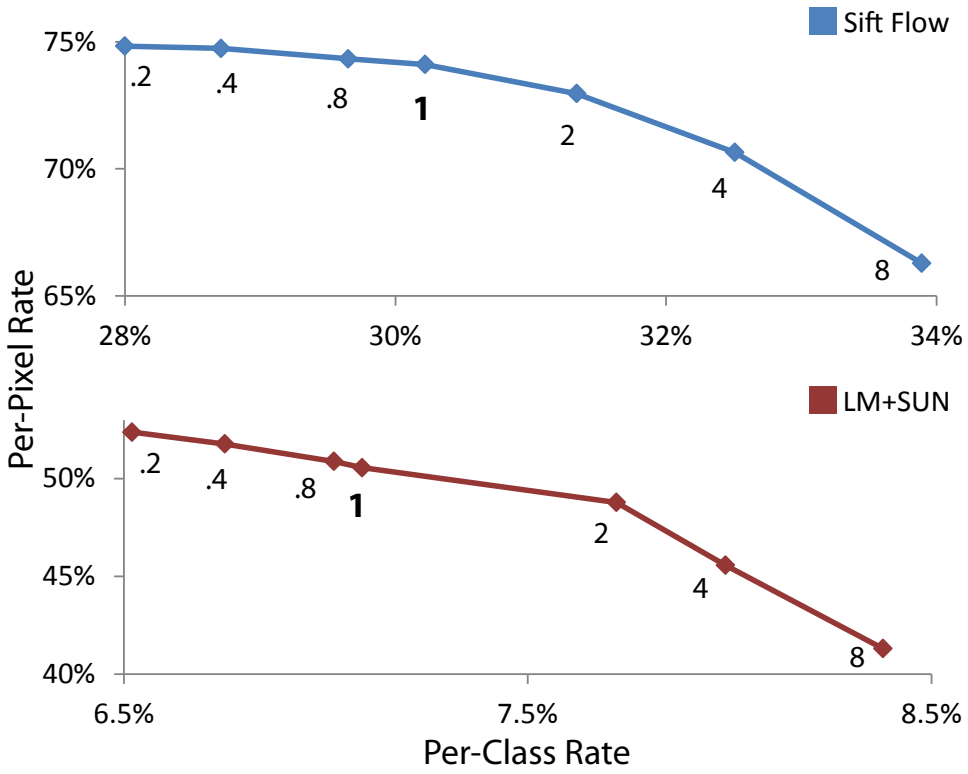

Figure 3.11: Results on SIFT Flow and LMSun dataset for local superpixel labeling with different values for the likelihood smoothing constant ϵ (section 3.1.2, equation 3.2). The constant adjusts the tradeoff between average class rate and per-pixel rate.

geometric classes (Section 3.1.5). I use 100 trees with a depth of 8 for each one-versus-all classifier. Table 3.8 compares my nearest neighbor (NN) scheme and BDT on both semantic and geometric labels (note that a retrieval set is not used with BDT). On the semantic classes which have a very unbalanced class distribution, the per-pixel rate is higher for BDT but the average per-class rate is lower. On the other hand, BDT easily outperforms the NN classifier for the geometric classes, which have a much more even class distribution. This validates the implementation choice discussed in Section 3.2.1, namely, using NN for semantic classes and BDT for geometric ones. Indeed, comparing the joint NN/NN and BDT/BDT results in the last line of Table 3.8 to the hybrid

	Nearest Neighbor		Boosted Decision Trees	
	Semantic	Geometric	Semantic	Geometric
Local labeling	74.1 (30.2)	88.4 (86.1)	75.4 (26.7)	90.2 (88.7)
MRF	76.2 (29.1)	89.0 (86.2)	77.0 (26.4)	90.6 (88.9)
Joint	76.5 (29.3)	89.1 (86.9)	76.9 (26.4)	90.7 (88.9)

Table 3.8: Comparison of my nearest neighbor classifier to boosted decision trees. While the boosted decision trees constantly perform better on the relatively balanced geometric labels, they have worse per-class rates on semantic labels with heavily skewed label counts.

NN/BDT result in Table 3.2, we can see that the latter one offers the best performance.

Similarly to the likelihood ratio smoothing constant ϵ, the MRF smoothing constant λ (equation 3.3) gives a trade-off between per-pixel and per-class accuracy, as shown in Figure 3.12. After $\lambda = 1$ both drop off, so I use that value for both datasets.

Running time

Finally, I analyze the computational requirements of my system. My current implementation is mostly in unoptimized and un-parallelized MATLAB (with some outside C code for feature extraction and MRF optimization), and all my tests are run on a single PC with Xeon 3.33 GHz six-core processors and 48 GB RAM. Table 3.9 shows a breakdown of the main stages of the computation. On the SIFT Flow dataset, feature extraction and image parsing takes less than 10 seconds. In comparison, as reported in (Liu et al., 2011b), to classify a single query image, the SIFT Flow system required 50 alignment operations that took 31 seconds each, or 25 minutes total without parallelization.

As can be inferred from Figure 3.13, my algorithm complexity is approximately quadratic in the average number of superpixels per image in the dataset due to the need to exhaustively match every test superpixel to every retrieval set superpixel. On

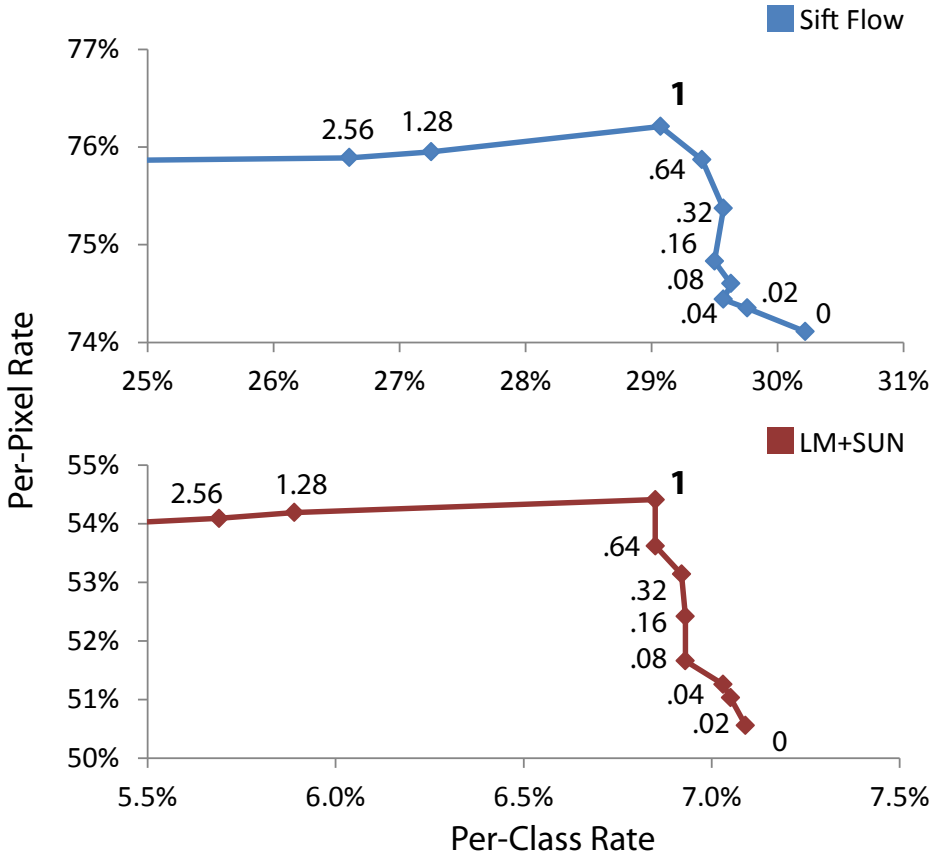

Figure 3.12: The effect of MRF smoothing parameter λ from equation 3.3. After $\lambda = 1$ the performance drops off rapidly so I use $\lambda = 1$.

the other hand, given a fixed retrieval set size, this time is independent of the overall number of training images. The big O complexity of my full system is $O(n^2t^2p^2l)$ (see table 3.9 for variable definitions). For the datasets used, retrieval and inference are fairly fast; these steps are represented by $O(t^2p^2l)$, which is small compared to $O(n^2)$ (the superpixel matching step). For LM+SUN, the main bottleneck of the system is not superpixel search, but file I/O for loading retrieval set superpixel descriptors from disk. However, it should be possible to overcome this bottleneck with appropriate hardware, parallelization, and/or data structures for fast search. As the dataset grows computing the retrieval

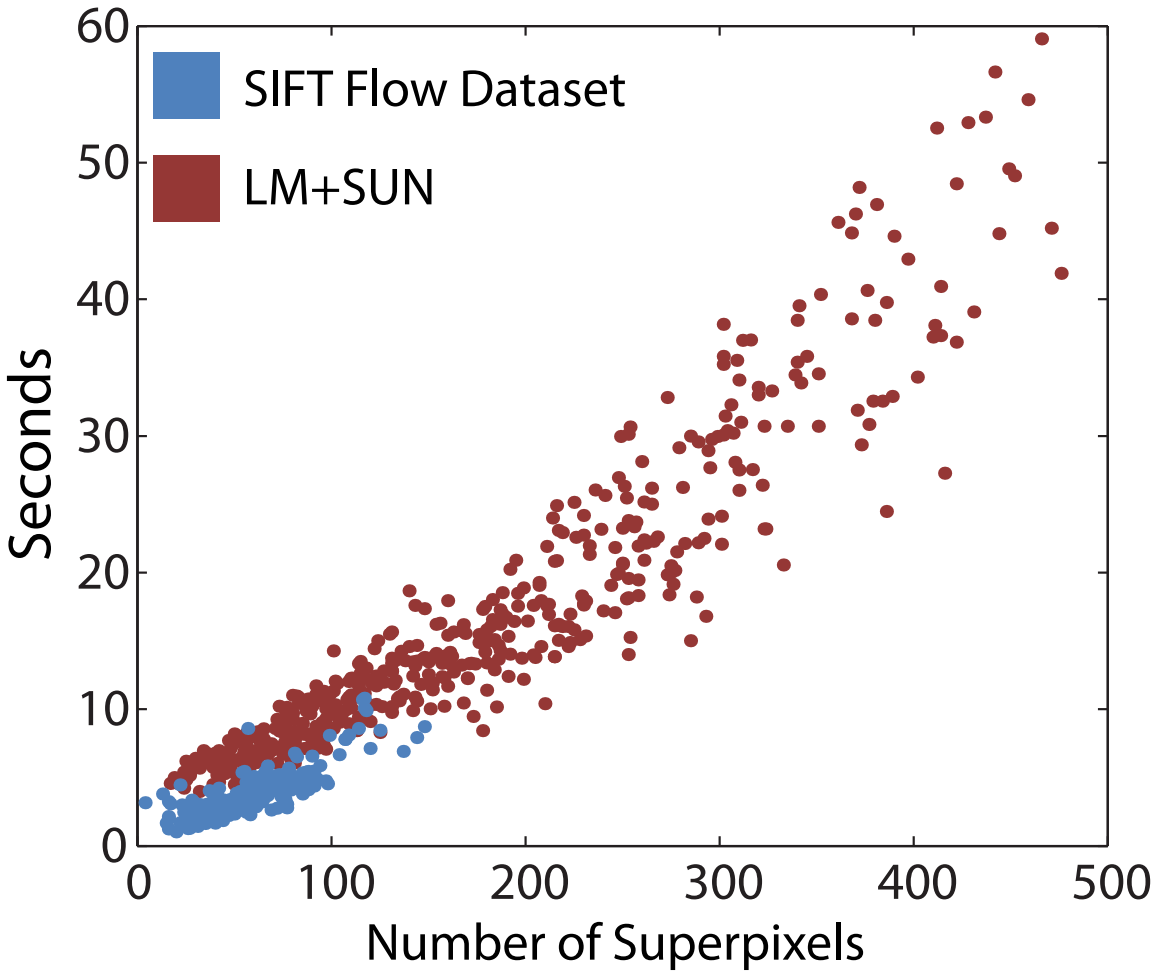

Figure 3.13: Query time vs. number of superpixels in the query image. Notice that for small images in the LM+SUN dataset, the retrieval set query time is the only part of the system that takes longer than on the SIFT Flow dataset and thus the total processing time remains similar.

	SIFT Flow	LM+SUN
Training set size (t)	2,488	45,176
Image size (p)	256×256	800×600
Avg. # superpixels (n)	63.9	178.2
Number of labels (l)	33	232
Feature extraction	1.5 ± 0.5	5.2 ± 1.8
Retrieval set search	0.04 ± 0.0	3.5 ± 0.51
Superpixel search	3.75 ± 1.8	13.1 ± 11.2
MRF solver	0.005 ± 0.003	$.009 \pm .006$
Total (excluding features)	4.4 ± 2.3	16.6 ± 11.7

Table 3.9: The average timing in seconds of the different stages in my system (excluding file I/O). While the runtime is significantly longer for the LM+SUN dataset, this is primarily due to the change in image size and not the number of images.

set will require a more significant amount of computation. Approximate nearest neighbor approaches, such as locality sensitive hashing (Indyk and Motwani, 1999; Gionis et al., 1999), can be used to reduce the computational complexity of finding the retrieval set with very little impact on performance as the retrieval set does not need to be exact. On a more fundamental research level, the dependence of running time on image resolution deserves some attention. The problem of efficiently parsing megapixel images while deriving additional recognition cues from the higher resolution is currently wide open and extremely challenging. Curiously, even as the sizes of datasets used in recognition research have increased dramatically in recent years, the resolution of individual images has not.

3.3 Video Parsing

This section presents the extension of my system to video. Video sequences provide richer information, which should be useful for better understanding scenes. Intuitively, motion

cues can improve object segmentation, and being able to observe the same objects in multiple frames, possibly at different angles or scales, can help build a better model of the objects' shape and appearance. On the other hand, the large volume of video data makes parsing very challenging computationally.

Previous approaches have tried a variety of strategies for exploiting the cues contained in video data. Brostow et al. (2008), Sturgess et al. (2009), and Zhang et al. (2010) extract 3D structure (sparse point clouds or dense depth maps) from the video sequences and then use the 3D information as a source of additional features for parsing individual frames. Xiao and Quan (2009) run a region-based parsing system on each frame and enforce temporal coherence between regions in adjacent frames as a post-processing step.

The video is segmented using a spatiotemporal segmentation method (Grundmann et al., 2010) that gives 3D regions or *supervoxels* that are spatially coherent within each frame (i.e., have roughly uniform color and optical flow) as well as temporally coherent between frames. The hope is that these regions will contain the same object from frame to frame. They system then computes local likelihood scores for possible object labels over each supervoxel, and finally, constructs a single graph for each video sequence where each node is a supervoxel and edges connect adjacent supervoxels. Inference on this graph is performed using the same MRF formulation as in Section 3.1.5. Section 3.3.1 will give details of my video parsing approach, and Section 3.3.2 will show that this approach significantly improves the performance compared to parsing each frame independently.

3.3.1 System Description

I wish to take advantage of the motion cues in video without explicitly adding motion or geometric features to my system. I do this by using the hierarchical video segmentation method of Grundmann et al. (2010), which Xu and Corso (2012) show to be quite effective at capturing the boundaries of objects in video. I run all videos through the segmentation website of Grundmann et al. (2010)[4] with default parameters to obtain a hierarchy of segmentation volumes and use the lowest level of the hierarchy as supervoxels.[5] Figure 3.14 contrasts the outputs of still image and video segmentation, showing that the supervoxel boundaries tend to better adhere to boundaries of objects such as cars.

Once supervoxels are computed for a video sequence, the data term $E_{\text{data}}(v_i, c)$ for each supervoxel v_i and each class label c needs to be computed. In principle, this could be done directly by extracting spatiotemporal features from each v_i, but to simplify the extension of the system from still images, I have chosen to combine scores computed over 2D time slices of v_i. Specifically, given a class c and the slice of supervoxel v_i in the jth frame, denoted s_i^j, a log likelihood ratio score $L(s_i^j, c)$ is computed. This can be done with either my NN scheme (equation 3.1) or with BDTs (Section 3.2.1), though in the experiments of the next section I use only BDTs. For combining the per-frame scores, I have tried a number of approaches and found the following heuristic to give the best

[4] http://videosegmentation.com/

[5] Since the videos were taken from a forward-moving camera, I have found the segmentation results to be better if I run the videos through the system backwards.

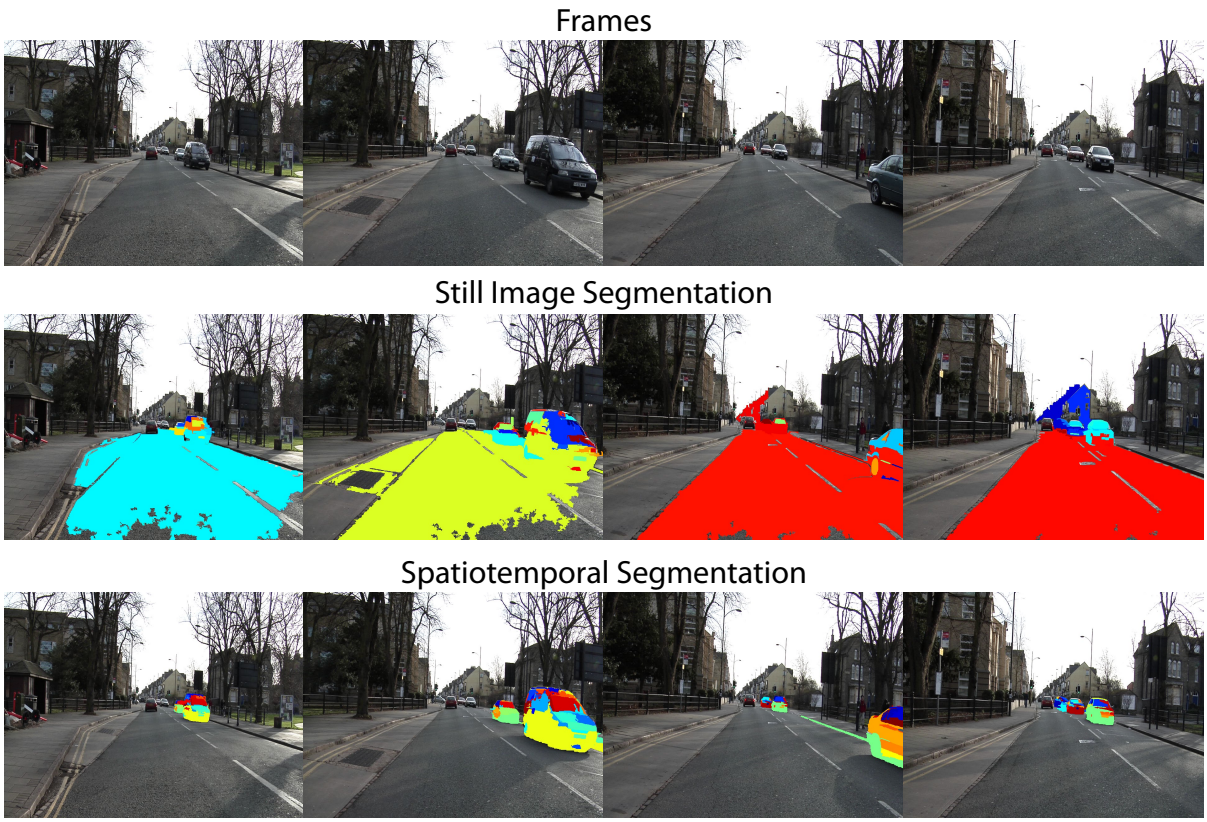

Figure 3.14: A comparison of still image segmentation of Felzenszwalb and Huttenlocher (2004) (second row) to the spatiotemporal segmentation of Grundmann et al. (2010) (third row). Shown are only the segments required to cover the foreground cars in each frame. The still image segmentation is not able to separate the lower parts of the cars from the road, while the spatiotemporal segmentation does not suffer from the same problem.

performance:

$$E_{\text{data}}(v_i, c) = -\max_j [w_i^j \sigma(L(s_i^j, c))], \tag{3.7}$$

where w_i^j is the relative size of s_i^j and $\sigma(\cdot)$ is a normalizing sigmoid function as in equation 3.4. In other words, each per-frame score is weighted by the size of the region in that frame (the idea being that frames in which the supervoxel is larger give better evidence

about its class identity) and take the maximum of the weighted scores over all the frames in which the supervoxel appears (intuitively, the frame in which the weighted score is highest is the one in which we got the best "look" at the object and were the most confident about its identity).

Finally, an MRF is constructed for the entire video sequence where nodes represent supervoxels and edges connect pairs of supervoxels that are spatially adjacent in at least one frame. I define the edge energy term in the same way as in the 2D case, using equation 3.5. I do this for both semantic and geometric classes and solve for them simultaneously using the same joint formulation as in equation 3.6. For the video sequences in my experiments, which range from 1,500 to 4,000 frames, I typically obtain graphs of 10,000 to 30,000 nodes, which are very tractable.

3.3.2 Results

I test the video segmentation on the standard CamVid dataset (Brostow et al., 2008), which consists of daytime and dusk videos taken from a car driving through Cambridge, England. There are a total of five video sequences. I follow the training/test split of Brostow et al. (2008), with two daytime and one dusk sequence used for training, and one daytime and one dusk sequence used for testing. The sequences are densely labeled at one frame per second with 11 class labels: Building, Tree, Sky, Car, Sign-Symbol, Road, Pedestrian, Fence, Column-Pole, Sidewalk, and Bicyclist. There are a total of 701 labeled frames in the dataset with 468 used for training and 233 for testing. Note that while I evaluate the accuracy of the output on only the labeled testing frames, the system

does obtain dense labels for all frames in the test video.

Table 3.10(a) shows baseline performance using the still image parsing approach that segments and labels each frame independently. Table 3.10(b) shows results with spatiotemporal segmentation used in two different ways. The first variant, "temporally incoherent," just uses the segmentation to generate the regions in each frame; each frame is still parsed independently, and regions belonging to the same supervoxel are not required to have the same label from frame to frame. The second variant, "temporally coherent," combines the per-frame likelihood scores as described in Section 3.3.1 to assign a single label to each supervoxel. Both methods give a significant improvement over still image parsing. Note that even though the temporally coherent method has a similar accuracy to the incoherent one, the output video is much more visually pleasing in the former case, since the labeling "flickers" much less over time (see Figure 3.3.2 for examples).

The sixth and seventh rows of Table 3.10(b) show the performance of the temporally coherent setup following contextual MRF smoothing and joint semantic/geometric inference. Somewhat disappointingly, both versions of the MRF give a very minimal improvement. This is likely due to a number of factors. First, MRF energy minimization on the spatiotemporal graph appears to be a harder problem, and the solutions tend to show a much greater tendency to oversmooth. Second, I gain a big improvement in object boundaries by incorporating motion cues into the segmentation, and this is likely diminishing the subsequent power of the MRF. Recall that in Section 3.2.3 we have seen a similar effect: as the local appearance model became more powerful by adding features, the improvement afforded by the MRF diminished (Figure 3.10). Finally, joint seman-

		Semantic	Geometric
(a)	Still Image Parsing		
	Local Labeling	76.9 (44.3)	91.6 (92.0)
	MRF	77.4 (43.5)	91.6 (91.9)
	Joint	77.6 (43.8)	91.7 (92.1)
(b)	Spatiotemporal Parsing		
	Temporally Incoherent	82.6 (51.2)	94.6 (94.8)
	Temporally Coherent	82.6 (51.3)	94.2 (94.8)
	MRF	83.0 (51.0)	94.2 (94.4)
	Joint	83.3 (51.2)	94.2 (94.7)
(c)	Brostow et al. (2008)	69.1 (53.0)	
	Sturgess et al. (2009)	83.8 (59.2)	
	Zhang et al. (2010)	82.1 (55.4)	
	Ladický et al. (2010a)	83.8 (62.5)	

Table 3.10: CamVid dataset results. (a) Still image segmentation baseline. (b) Results with spatiotemporal segmentation (see text). (c) Competing state-of-the-art approaches. As before, per-pixel classification rate is followed by the average per-class rate in parentheses.

tic/geometric inference introduces very few new constraints, since the CamVid dataset has only three non-vertical classes (sky, road, and sidewalk).

For reference, Table 3.10(c) shows the performance of recent state-of-the-art methods on the CamVid dataset. My system beats Brostow et al. (2008) and comes close to Ladický et al. (2010a), Sturgess et al. (2009) and Zhang et al. (2010). Table 3.11 gives a more detailed class-by-class comparison. By comparing the first two lines of the table, we can see that spatiotemporal segmentation gives the biggest improvements on the smaller moving object classes such as car, pedestrian, and bicyclist. In absolute terms, however, the system does not do well on these classes, just as it did not do well on them in my still image datasets. Interestingly, spatiotemporal segmentation also gives a significant boost on "sidewalk," which happens to be similar to the effect I got by using

	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian
Still image parsing (joint sem./geom.)	84.8	65.1	94.7	47.5	24.6	96.2	8.3
Spatiotemporal parsing (joint sem./geom.)	**87.0**	67.1	96.9	62.7	30.1	95.9	14.7
Brostow et al. (2008)	46.2	61.9	89.7	68.6	42.9	89.5	**53.6**
Sturgess et al. (2009)	84.5	72.6	**97.5**	72.7	34.1	95.3	34.2
Zhang et al. (2010)	85.3	57.3	95.4	69.2	**46.5**	**98.5**	23.8
Ladický et al. (2010a)	81.5	**76.6**	96.2	**78.7**	40.2	93.9	43.0

	Fence	Column-Pole	Sidewalk	Bicyclist		Per-class	Per-pixel
Still image parsing (joint sem./geom.)	9.1	3.4	43.7	3.9		43.8	78.6
Spatiotemporal parsing (joint sem./geom.)	17.9	1.7	70.0	19.4		51.2	83.3
Brostow et al. (2008)	46.6	0.7	60.5	22.5		53.0	69.1
Sturgess et al. (2009)	45.7	8.1	77.6	28.5		59.2	**83.8**
Zhang et al. (2010)	44.3	**22.0**	38.1	28.7		55.4	82.1
Ladický et al. (2010a)	**47.6**	14.3	**81.5**	**33.9**		**62.5**	**83.8**

Table 3.11: Per-class performance on the CamVid (Brostow et al., 2008) dataset.

ground truth segmentation on the SIFT Flow dataset (Figure 3.9). Thus, it is plausible that the video segmentation gets closer to the true object boundaries.

Note that I use the motion information in video only to improve the segmentation, not to change the features. By contrast, Brostow et al. (2008), Sturgess et al. (2009) and Zhang et al. (2010) use features derived from 3D point clouds or depth maps, while Ladický et al. (2010a) incorporate sliding window object detectors. Overall, my experiments on video confirm the flexibility and broad applicability of my image parsing frame-

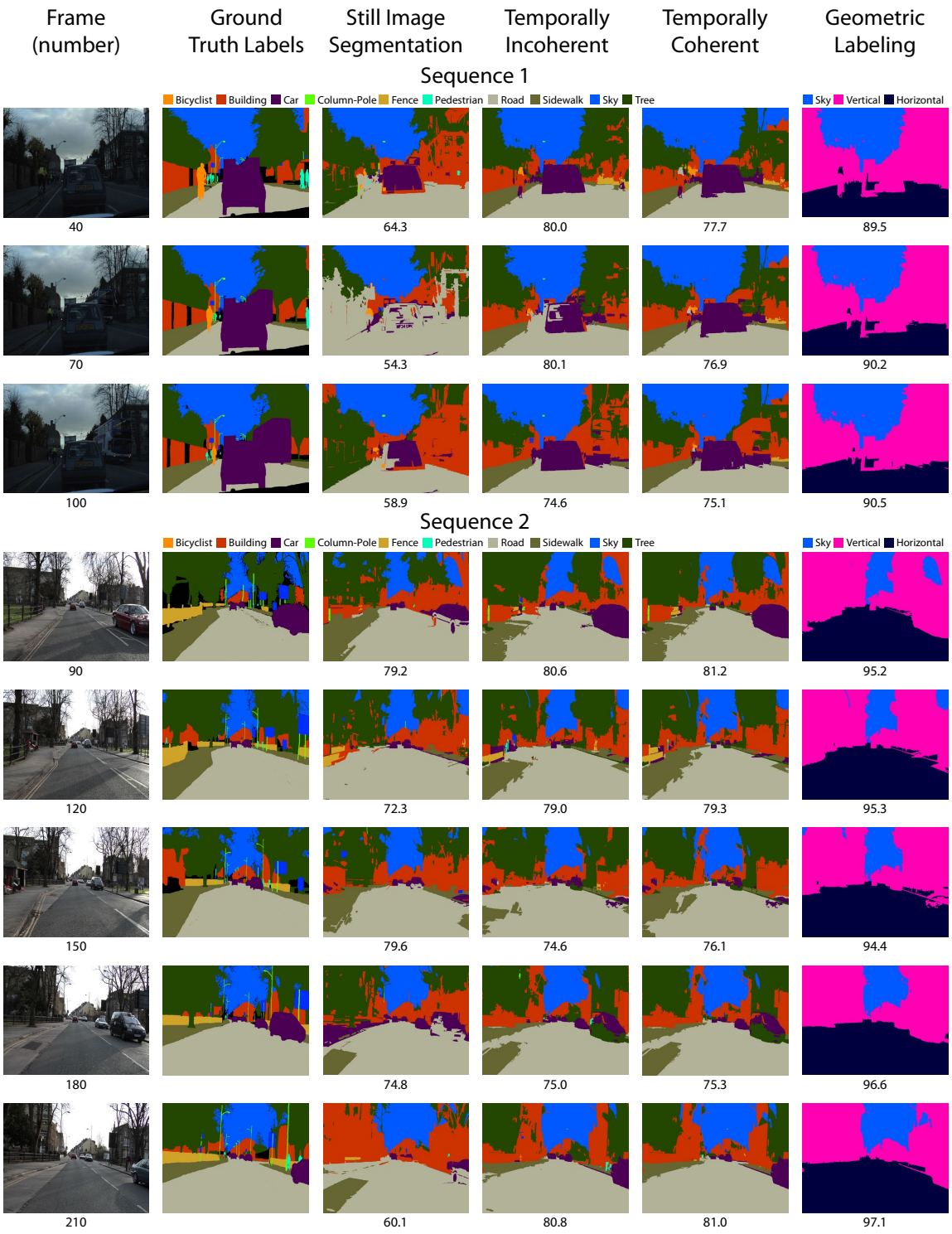

Figure 3.15: Example results from the CamVid test set (best viewed in color). Still image parsing (third column) is very unstable, with both the superpixels and their inferred labels changing incoherently from one frame to the next. Temporally incoherent parsing (fourth column) uses spatiotemporal segmentation, which corrects most of these issues but can still be inconsistent in time as shown in frames 70 and 100. The final system (fifth column) has temporally consistent segments and labels.

work, and give me additional insights into its strengths and weaknesses that complement my findings on still image datasets. While I did not outperform current state-of-the-art methods on the CamVid dataset here, I show how to boost this performance to the state-of-the-art in Chapter 5 by incorporating object detectors into my parsing framework.

3.4 Discussion

In this chapter I have presented a superpixel-based approach to image parsing that can take advantage of datasets consisting of tens of thousands of images annotated with hundreds of labels. My underlying feature representation, based on multiple appearance descriptors computed over segmentation regions, is simple and allows new features to be easily incorporated. I also use efficient MRF optimization to capture label co-occurrence context, and to jointly label regions with semantic and geometric classes.

My system has demonstrated state-of-the-art results on the SIFT Flow and LM+SUN datasets with a nonparametric version of my system based on a two-stage approach (global retrieval set matching followed by superpixel matching). This framework does not need any training, except for computation of basic statistics such as label co-occurrence probabilities, and it relies on just a few constants that are kept fixed for all datasets. In principle, it is applicable to "open universe" datasets where the set of training examples and target classes may evolve over time. In particular, my results on the LM+SUN dataset, which has 45,676 images and 232 labels, constitute an important baseline for future approaches. To my knowledge, it is currently the largest dense per-pixel image parsing dataset and, unlike most other general-purpose image parsing benchmarks, it

includes both outdoor and indoor images. As I have shown, the latter pose severe recognition challenges, and deserve more study in the future.

Besides the nonparametric "open universe" regime, my system has the flexibility to operate with offline pre-trained classifiers, such as boosted decision trees. The use of these may be preferable for static datasets with smaller numbers of classes and a more balanced class distribution.

Finally, I have demonstrated an extension of my system to video. This extension segments the video into spatiotemporal "supervoxels" and uses a simple heuristic to combine local appearance cues across frames. The resulting approach does not exploit all the motion information that is potentially available in video (in particular, it does not attempt to extract 3D geometry), but it still affords a big improvement over incoherent frame-by-frame parsing.

Through the extensive analysis of Section 3.2.3, I have identified two major limitations of my system. First, the scene matching step for obtaining the retrieval set suffers from an inability of low-level global features such as GIST to retrieve semantically similar scenes, resulting in incoherent interpretations (e.g., indoor and outdoor class labels mixed together). Second, my reliance on bottom-up segmentation really hurts my performance on "thing" classes. Traditionally, such classes are handled using sliding window detectors, and there exists work (e.g., Ladický et al. (2010a)) attempting to incorporate such detectors into region-based parsing. I explore the idea of per-exemplar detectors (Malisiewicz et al., 2011) to complement my superpixel-based approach in Chapter 5.

CHAPTER 4: UNDERSTANDING SCENES ON MANY LEVELS

This chapter examines the question of what labeling to use to best represent the structure and semantics of a scene. We can label image regions with basic-level category names such as grass, sheep, cat, and person, as has been done thus far. Of course, coarser superordinate-level labels can be assigned, such as animal, vehicle, manmade object, natural object, etc. We can continue to assign geometric labels such as horizontal, vertical and sky. We can also assign material labels such as skin, metal, wood, glass, etc. Further, some regions belonging to structured, composite objects may be given labels according to their part identity: if a region belongs to a car, it may be a windshield, a wheel, a side door, and so on.

The goal of this chapter is to understand scenes on multiple levels: rather than assigning a single label to each region, multiple labels will be assigned simultaneously, such as a basic-level category name, a superordinate category name, material, and part identity. First I obtain datasets with different overlapping sets of labels assigned to pixels. Then by inferring all the labelings jointly, the system can take into account constraints of the form "roads are horizontal," "cars are made of metal," "cars have wheels" but "horses have legs," leading to an improved interpretation of the image.

In the previous chapter I jointly inferred labels from two label types that formed a strict hierarchy. In this chapter I generalize this idea to an arbitrary number of label types with arbitrary relationships. Relationships between two different label types may

be hierarchical (e.g., a car is a vehicle) or many-to-many (e.g., a wheel may belong to multiple types of vehicles, while a vehicle may have many other parts besides a wheel).

I show how to formulate the inference problem so that agreement between different types of labels is enforced and apply this formulation to two large-scale datasets with very different characteristics (Figure 4.1). My results show that simultaneous multi-level inference gives a higher performance than treating each label set in isolation. Figure 4.2 illustrates the source of this improvement. In this image, the basic-level object labeling is not sure whether the object is an airplane or a bird. However, the superordinate animal/vehicle labeling is confident that it is a vehicle, and the materials labeling is confident that the object is made (mostly) of painted metal. By performing joint inference over all these label sets, the correct hypothesis, airplane, is allowed to prevail. This work was originally published in ICCV (Tighe and Lazebnik, 2011).

4.1 Multi-Level Inference

I begin with my core, single-level image parsing system presented in Section 3.1.4 and extend it to an arbitrary number of label types. I extend the single-level MRF objective function (3.3) to perform simultaneous inference over multiple label sets. If I have n label sets, then I want to infer n labelings $\mathbf{c}^1, \ldots \mathbf{c}^n$, where $\mathbf{c}^l = \{c_i^l\}$ is the vector of labels from the lth set for every region $r_i \in R$. I can visualize the n labelings as being "stacked" together vertically as shown in Figure 4.3. "Intra-set" edges connect labels of neighboring regions in the same level just as in the single-level setup of Section 3.1.4, and "inter-set" edges connect labels of the same region from two different label sets. The

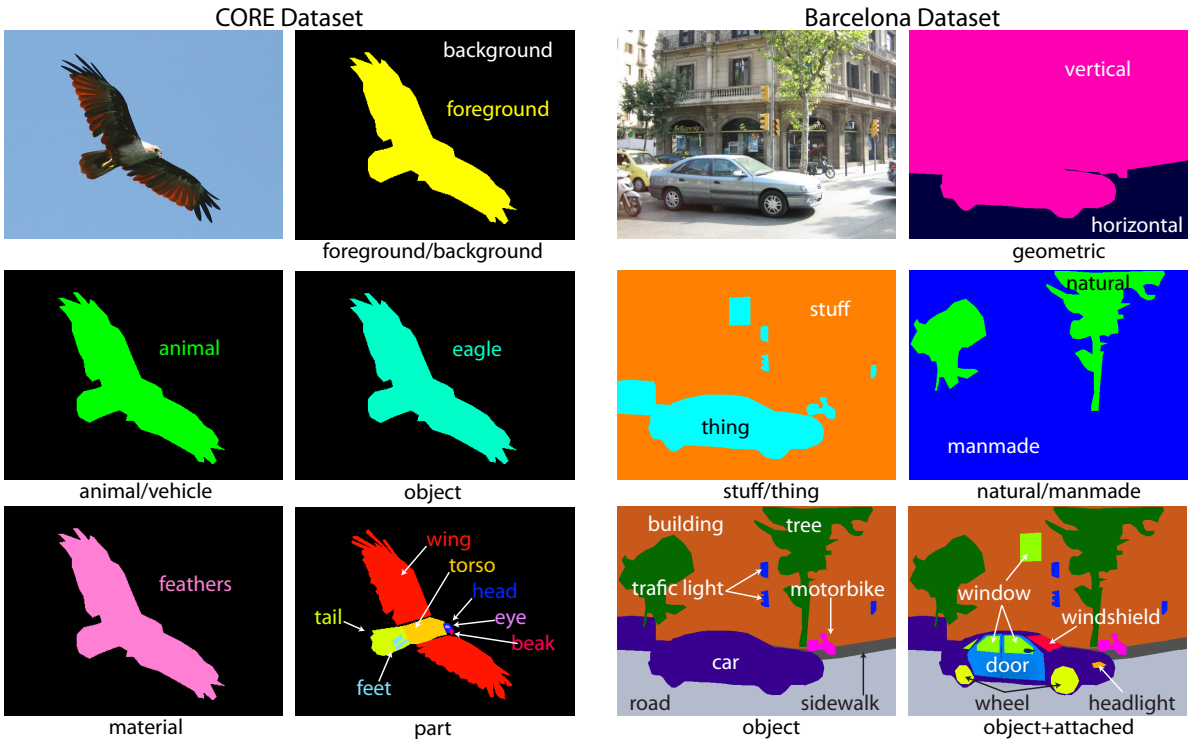

Figure 4.1: Sample ground truth labelings from my two datasets (see Section 4.2 for details).

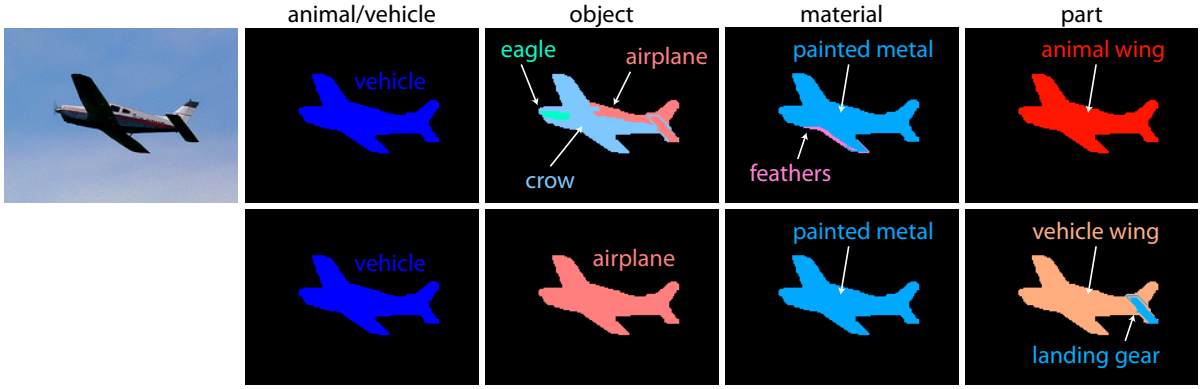

Figure 4.2: It's a bird, it's a plane! First row: single-level MRF inference for each label set in isolation. Second row: joint multi-level MRF inference. Animal/vehicle and material labelings are strong enough to correct the object and part labelings.

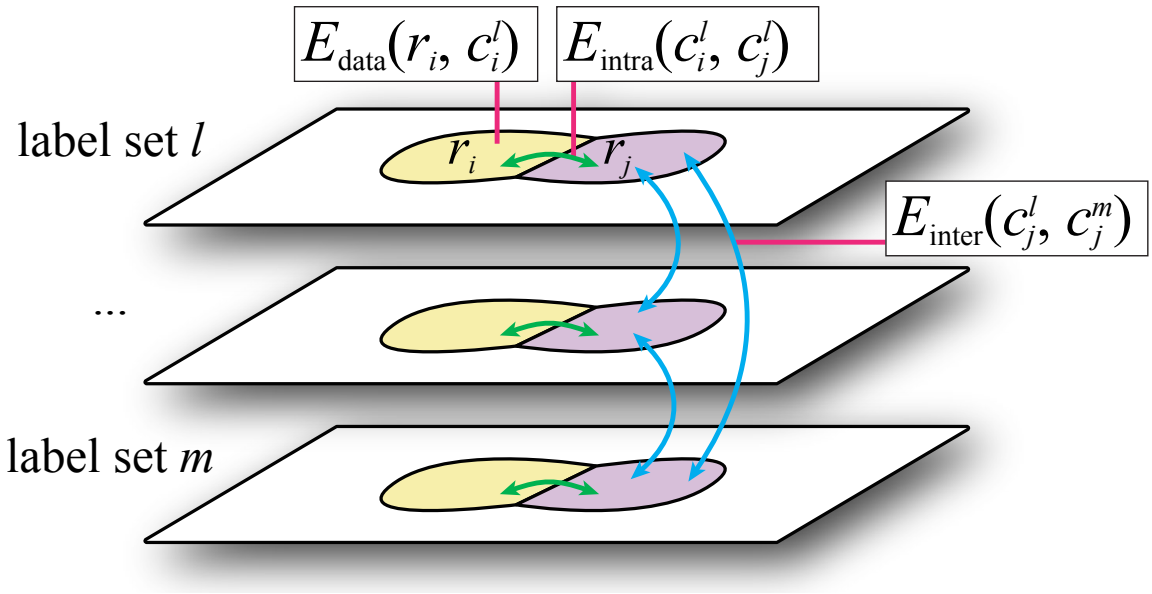

Figure 4.3: Illustration of my multi-level MRF (equation 4.1).

MRF energy function on the resulting field is

$$\begin{aligned} E(\mathbf{c}^1, \ldots, \mathbf{c}^n) = & \sum_l \sum_{r_i \in R} E_{\text{data}}(r_i, c_i^l) \\ & + \lambda \sum_l \sum_{(r_i, r_j) \in A} E_{\text{intra}}(c_i^l, c_j^l) \\ & + \mu \sum_{l \neq m} \sum_{r_i \in R} E_{\text{inter}}(c_i^l, c_i^m), \end{aligned} \tag{4.1}$$

where $E_{\text{data}}(r_i, c_i^l)$ is the data term for region r_i and label c_i^l on the lth level, $E_{\text{intra}}(c_i^l, c_j^l)$ is the single-level smoothing term, $E_{\text{inter}}(c_i^l, c_i^m)$ is the term that enforces consistency between the labels of r_i drawn from the lth and mth label sets. Finally, the constants λ and μ control the amount of intra-set and inter-set smoothing.

E_{data} and E_{intra} are defined in the same way as in Section 3.1.4. As for the cross-level penalty E_{inter}, it is defined very similarly to to the within-level penalty (3.5), based on

cross-level co-occurrence statistics from the training set:

$$E_{\text{inter}}(c_i^l, c_i^m) = -\log[(P(c_i^l|c_i^m) + P(c_i^m|c_i^l))/2]\,. \tag{4.2}$$

Note that there is no need for a $\delta[c^l \neq c^m]$ factor here as in (3.5) because the labels in the different label sets are defined to be distinct by construction. Intuitively, E_{inter} can be thought of as a "soft" penalty that incorporates information not only about label compatibility, but also about the frequencies of different classes. If c^l and c^m co-occur often as two labels of the same region in the training dataset, then the value of $E_{\text{inter}}(c^l, c^m)$ will be low, and if they rarely or never co-occur, then the value will be high. For example, buildings are very common manmade objects, so the value of E_{inter} for "building" and "manmade" would be low. Traffic lights are also manmade objects, but there are fewer "traffic light" regions in the training set, so the penalty between "traffic light" and "manmade" would be higher. I have found (4.2) to work well for both many-to-many and hierarchical relationships between label sets. Figure 4.4 shows a few of the E_{inter} values for objects and parts on the CORE dataset (Section 4.2.2).

Finally, before performing graph cut inference, I linearly rescale all the values of E_{intra} and E_{inter} to the range $[0, 1]$ (note that E_{data} is already in the range $[0, 1]$ due to the application of the sigmoid).

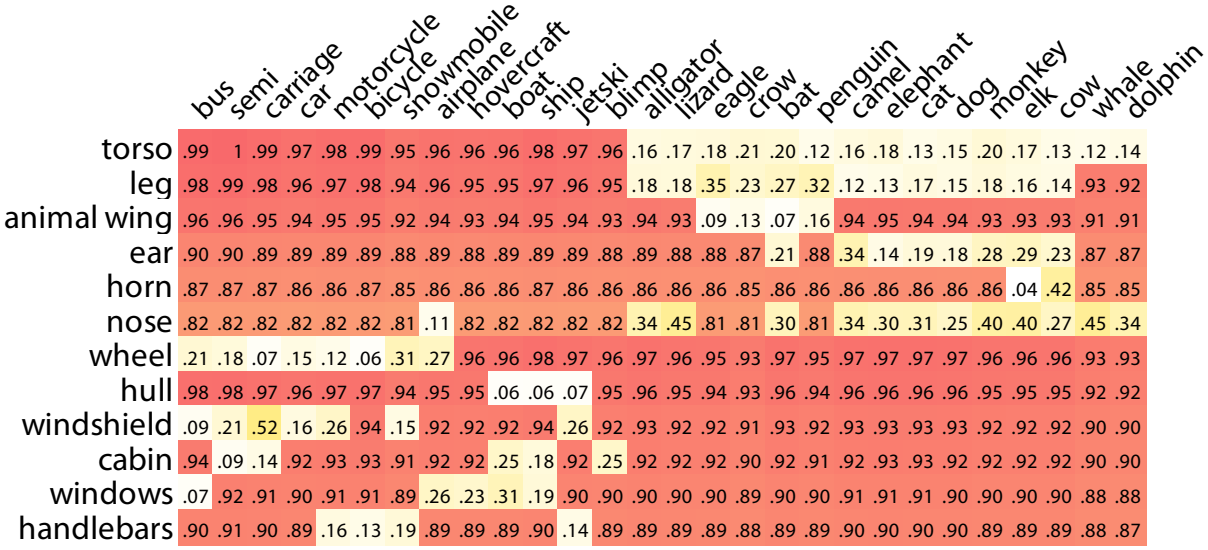

Figure 4.4: A sample of E_{inter} penalties (equation 4.2) for the "object" and "part" label sets on the CORE dataset (only 12 of the 66 parts are shown). The values are rescaled to $[0, 1]$. From the table, I can see, for example, that elk almost always have horns (a very small penalty of .04), cows sometimes have horns (a moderate penalty of .42), and all other classes have no horns. As another example, legs have a slightly higher penalty for birds than for the other animals, because they are visible less often.

4.2 Experiments

4.2.1 Barcelona Dataset

My first dataset is the "Barcelona" dataset from Tighe and Lazebnik (2010) (originally from LabelMe (Russell et al., 2008)). It consists of 14,592 training images of diverse scene types, and 279 test images depicting street scenes from Barcelona.

For this dataset, I create five label sets (Table 4.1) based on the synonym-corrected annotations from Tighe and Lazebnik (2010). The largest label set is given by the 170 "semantic" labels from Tighe and Lazebnik (2010). Here, I call this set "objects+attached" because it includes not only object names such as car, building, and person, but also

Label set	Labels	Sample labels in set
Geometric	3	sky, vertical, horizontal
Stuff/thing	2	stuff, thing
Natural/manmade	2	natural, manmade
Objects	135	road, car, building, ...
Objects+attached	170	window, door, wheel, ...

Table 4.1: Barcelona dataset label sets.

names of attached objects or parts such as wheel, door, and crosswalk. Note that ground-truth polygons in LabelMe can overlap; it is common, for example, for an the image region covered by a "wheel" polygon to also be covered by a larger "car" polygon. In fact, polygon overlap in LabelMe is a very rich cue that can even be used to infer 3D structure (Russell and Torralba, 2009). I want my automatically computed labeling to be able to reflect overlap and part relationships, e.g., that the same region can be labeled as "car" and also "wheel", or "building" and also "door." To accomplish this, I create another label set consisting of a subset of the "objects+attached" labels corresponding to stand-alone objects. The 135 labels in this set have a many-to-many relationship with the ones in the "object+attached" set: a wheel can belong to a car, a bus or a motorbike, while a car can have other parts besides a wheel attached to it.

The remaining three label sets represent different groupings of the "object+attached" labels. One of these is given by the "geometric" labels (sky, ground, vertical) from Tighe and Lazebnik (2010). The other two are "natural/manmade" and "stuff/thing." "Stuff" (Adelson, 2001) includes classes like road, sky, and mountain; "things" include car, person, sign, and so on. Just as the "geometric" assignments from Section 3.1.5, these assignments are done manually, and are in some cases somewhat arbitrary: I designate

buildings as "stuff" because they tend to take up large portions of the images, and their complete boundaries are usually not visible. Once the multi-level ground truth labelings are specified, the cross-level co-occurrence statistics are automatically computed and used to define the E_{inter} terms as in equation 4.2.

To obtain the E_{data} terms for the different label sets, I train boosted decision tree classifiers on the smaller "geometric," "natural/manmade," and "stuff/thing" sets, and use the nonparametric nearest-neighbor classifiers from Chapter 3 on the larger "objects" and "objects+attached" sets. This is consistent with the system of in Chapter 3, which used the decision trees for "geometric" classes and the nonparametric classifiers for "semantic" classes.

The scores output by the decision trees and the nonparametric classifiers have different ranges, roughly $[-5, 5]$ and $[-50, 50]$. I set the respective scale parameters for the sigmoid in (3.4) to $\gamma = 0.5$ and $\gamma = 0.05$. As for the smoothing constants from (4.1), I use $\lambda = 8$ and $\mu = 16$ for all the experiments. Even though these values were set manually, they were not extensively tuned, and I found them to work well on all label sets and on both of my datasets, despite their very different characteristics.

Table 4.2 shows the quantitative results for multi-level inference on the Barcelona dataset. I report both the overall classification rate (the percentage of all ground-truth pixels that are correctly labeled) and the average of the per-class rates. The former performance metric reflects primarily how well I do on the most common classes, while the latter one is more biased towards the rare classes. What I would really like to see is an improvement in *both* numbers; in my experience, even a small improvement of this

	Base (E_{data} only)	Single-level MRF (eq. 3.3)	Multi-level MRF (eq. 4.1)	Two-level MRF (eq. 3.6)
Geometric	91.7 (87.6)	91.4 (86.5)	**91.8 (87.6)**	91.5 (86.8)
Stuff/thing	86.9 (66.7)	89.2 (64.2)	**90.3 (66.8)**	
Natural/manmade	87.6 (81.8)	88.4 (81.0)	**88.9 (81.8)**	
Objects	66.1 (9.7)	68.2 (8.6)	**69.3 (9.9)**	
Objects+attached	62.3(7.4)	64.4 (6.5)	**65.2 (7.4)**	65.0 (7.3)

Table 4.2: Barcelona dataset results. The first number in each cell is the overall per-pixel classification rate, and the second number in parentheses is the average of the per-class rates. The base system classifies each region according to the label with the lowest E_{data} term, single-level MRF performs separate inference in each label set using equation 3.3, and multi-level MRF performs inference using equation 4.1. For reference, the far right column shows the performance of the system from section 3.1.5, based on joint inference of only the "geometric" and "objects+attached" label sets.

kind is not easy to achieve and tends to be an indication that "You're doing someting right." Compared to a baseline that minimizes the sum of E_{data} terms (first column of Table 4.2), separately minimizing the MRF costs of each level (second column) tends to raise the overall classification rate but lower the average per-class rate. This is due to the MRF smoothing out many of the less common classes. By contrast, the joint multi-level setup (third column) raises *both* the overall and the average per-class rates for *every* label set. In addition, performing inference over my five label sets simultaneously gives slightly higher results than the two-level geometric/semantic setup from section 3.1.5 (last column of table). Thus, adding extra levels contributes some useful information to the overall image interpretation problem.

Figure 4.5 shows the output of my system on a few images. In these examples, the system is able to partially identify attached objects such as doors, crosswalks, and wheels. On the whole, though, my system does not currently do a great job on attached objects: it correctly labels only 13% of the door pixels, 11% of the crosswalks, and 4% of the wheels.

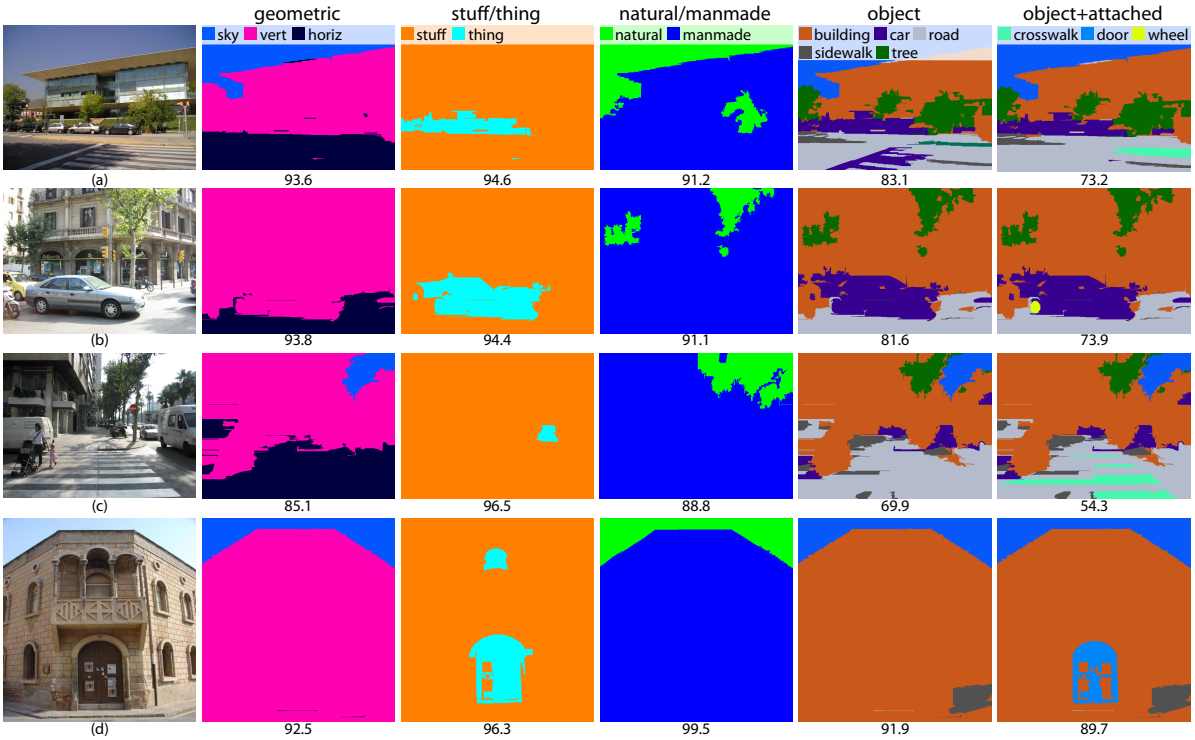

Figure 4.5: Output of the five-level MRF on sample images from the Barcelona dataset. The number under each image shows the percentage of pixels labeled correctly. Notice that I am able to correctly classify a wheel (b), parts of the crosswalk (a,c) and a door (d).

While my multi-level setup ensures that only plausible "attached" labels get assigned to the corresponding objects, I don't yet have any specific mechanism for ensuring that "attached" labels get assigned at all—for example, a wheel on the "objects+attached" level can still be labeled as "car" without making much difference in the objective function. Compounding the difficulty is the fact that my system performs relatively poorly on small objects: they are hard to segment correctly and MRF inference tends to smooth them away. Nevertheless, the preliminary results confirm that my framework is at least expressive enough to support "layered" labelings.

4.2.2 CORE Dataset

My second set of experiments is on the Cross-Category Object Recognition (CORE) dataset from Farhadi et al. (2010). This dataset consists of 2,780 images and comes with ground-truth annotation for four label sets.[1] Figure 4.1 shows a sample annotated image and Table 4.3 lists the four label sets. The "objects" set has 28 different labels, of which 15 are animals and 13 are vehicles. The "animal/vehicle" label set designates each object accordingly. The "material" set consists of nine different materials and the "part" set consists of 66 different parts such as foot, wheel, wing, etc. Each CORE image contains exactly one foreground object, and only that object's pixels are labeled in the ground truth. The "material" and "part" sets have a many-to-many relationship with the object labels; both tend to be even more sparsely labeled than the objects (i.e., not all of an object's pixels have part or material labels). Finally, because this is an object-centric dataset, only the object pixels are labeled. To account for this, I introduce a special "foreground/background" label set. This set is not included in the multi-level inference, but is used separately to produce a foreground mask that can be superimposed on the results of the joint inference of the other four layers.

I create validation and test sets of 224 images each by randomly taking eight images each from the 28 different object classes. To obtain the data terms, a boosted decision tree classifier is trained for each label in each set. For example, a "leg" classifier is

[1]There are also annotations for attribute information, such as "can fly" or "has four legs," which I do not use. Note that I use the CORE dataset differently than Farhadi et al. (Farhadi et al., 2010), so I cannot compare with their results. The representation of (Farhadi et al., 2010) is based on sliding window detectors, and does not produce dense image labelings. Moreover, while (Farhadi et al., 2010) focuses on cross-category generalization, I focus on exploiting the context encoded in the relationships between object, material, and part labels.

Label Set	Labels	Sample labels in set
Foreground/background*	2	foreground, background
Animal/vehicle	2	animal, vehicle
Objects	28	airplane, alligator, bat, bus, ...
Material	9	fur/hair, glass, rubber, skin, ...
Parts	66	ear, flipper, foot, horn, ...

Table 4.3: The CORE dataset label sets. *The foreground/background set is excluded from multi-level inference. It is used separately to generate a mask that is applied to the results of the other four levels as a post-process (see text).

trained using as positive examples all regions labeled as "leg" regardless of their object class (elk, camel, dog, etc.); all the regions from the "parts" level that have any label except "leg" are used as negative examples. Note that unlabeled regions are excluded from the training. The sigmoid parameter in the data term is $\gamma = 0.5$, the same as for the decision trees on the Barcelona dataset.

Figure 4.6 shows the output of four-level inference on an example image. One problem becomes immediately apparent: my approach, of course, is aimed at dense scene parsing, but the CORE dataset is object-centric. My inference formulation knows nothing about object boundaries and all the data terms are trained on foreground regions only, so nothing prevents the object labelings from getting extended across the entire image. At the end of this section, I will present a method for computing a foreground/background mask, but in the meantime, I quantitatively evaluate the multi-level inference by looking at the percentage of pixels with ground-truth labels (i.e., foreground pixels) that I label correctly. The first three rows of Table 4.4 show a comparison of the overall and per-class rates on each label set for the baseline (data terms only), separate single-level MRF inference, and joint multi-level inference. Even though CORE is very different from the

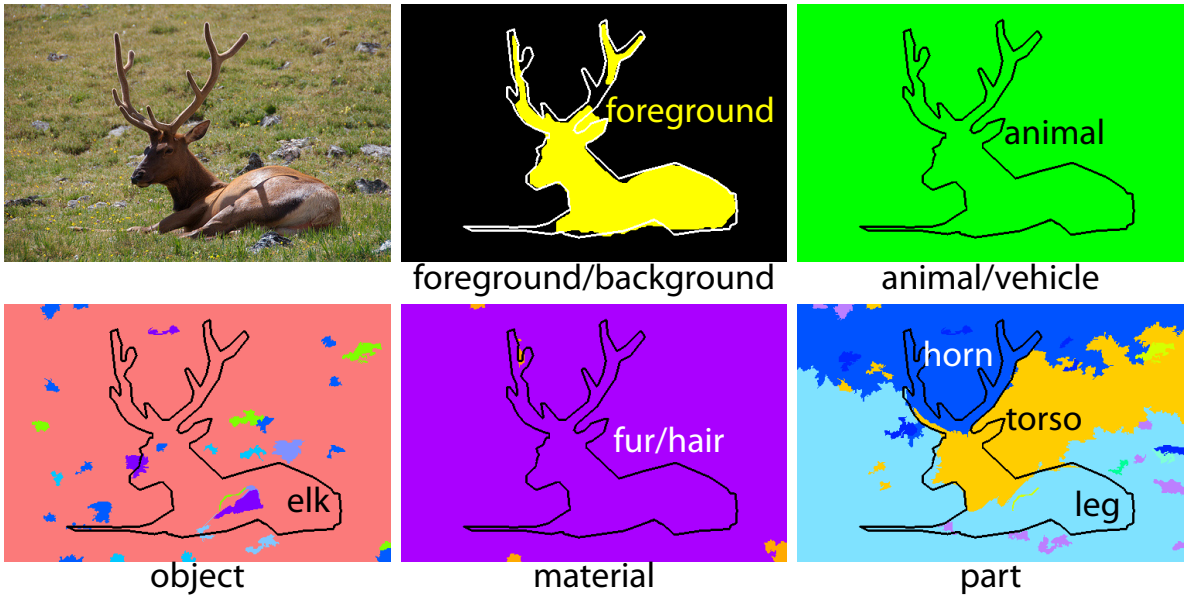

Figure 4.6: The output of my multi-level MRF on the CORE dataset. The ground truth foreground outline is superimposed on every label set. Note the "objects" level contains numerous small misclassified regions. They could be eliminated with stronger smoothing (higher value of λ in (4.1)), but that would also eliminate many "small" classes, lowering the overall performance on the dataset.

Barcelona dataset, the trend is exactly the same: multi-level inference improves *both* the overall and the average per-class rate for *every* label set.

Next, one can observe that each CORE image has only a single foreground object, so it can contain only one label each from the "animal/vehicle" and "object" sets. To introduce this global information into the inference, a one-vs-all SVM classifiers is trained for the 28 object classes and a binary SVM for animal/vehicle. The classifiers are based on three global image features: gist (Oliva and Torralba, 2006), a three-level spatial pyramid (Lazebnik et al., 2006) with a dictionary size of 200, and an RGB color histogram, the same features used to generate the retrieval set. A Gaussian kernel is used for GIST and histogram intersection kernels (Barla et al., 2003) are used for the other

	Animal/vehicle	Object	Material	Part
Base	86.6 (86.6)	34.4 (33.4)	51.8 (36.0)	37.1 (11.2)
Single-level MRF	91.1 (91.0)	43.2 (41.7)	54.1 (34.3)	42.6 (11.7)
Multi-level MRF	91.9 (92.0)	44.5 (43.1)	54.9 (35.9)	42.7 (11.9)
Base + SVM	92.8 (92.9)	43.5 (41.8)	51.8 (36.0)	37.1 (11.2)
Single-level MRF + SVM	92.8 (92.9)	53.2 (50.5)	54.1 (34.3)	42.6 (11.7)
Multi-level MRF + SVM	92.8 (92.9)	**53.9 (51.0)**	**56.4 (36.7)**	**43.9 (12.3)**

Table 4.4: CORE dataset results. The first number in each cell is the overall per-pixel classification rate, and the number in parentheses is the average of the per-class rates. All rates are computed as the proportion of ground-truth foreground pixels correctly labeled. The bottom three rows show results for using global SVM classifiers to reduce the possible labels in the animal/vehicle and object label sets (see text).

two features, then the kernel outputs are averaged. The classifiers are rebalanced on the validation set by setting their threshold to the point yielding an equal error rate on the ROC curve. Then, given a query image, a "shortlist" of labels is created of positive SVM responses. The multi-level inference is then restricted to using only those labels. For the animal/vehicle set, only one label is selected (since the classifier is binary), but for the object set, multiple labels can be selected. I have found that selecting multiple labels produces better final results than winner-take-all, and it still allows my superpixel classifier to have some influence in determining which object is present in the image. The last three rows of Table 4.4 show that the SVMs add 3.3% to the accuracy of the "animal/vehicle" level and 6.0% to "object." More significantly, the joint inference is able to capitalize on these improvements and boosts performance on materials and parts, even though the SVMs do not work on them directly. Figure 4.7 shows final classification rates for a selection of classes.

There still remains the issue of distinguishing foreground from background regions. To do this, I first train a boosted decision tree classifier using all the labeled (respectively

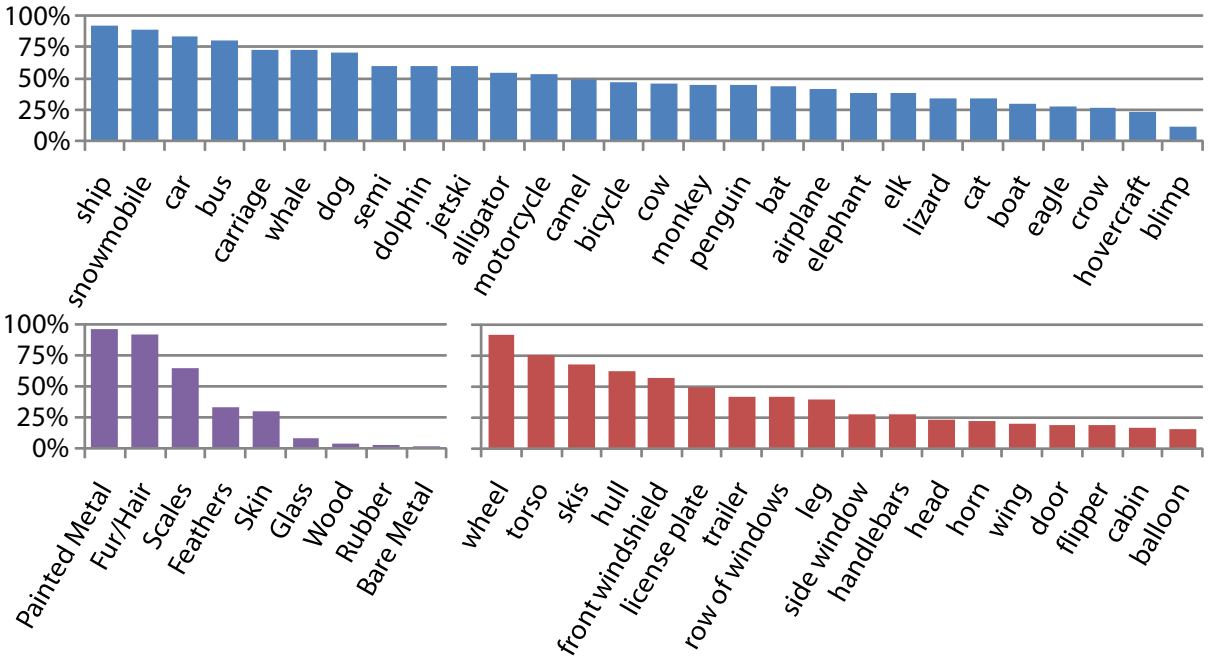

Figure 4.7: The per-class rates of the 28 objects, 9 materials, and top 18 parts in the CORE dataset. The labels are ordered by classification rate. As on the Barcelona dataset, the system does worse on the parts that tend to be small: nose, ear, beak (not shown), and on materials that have few examples for training: wood, rubber, bare metal.

unlabeled) training image regions as positive (respectively negative) data. This classifier correctly labels 64.7% of foreground pixels and 94.3% of background pixels. The resulting foreground masks are not very visually appealing, as shown in Figure 4.8. To improve segmentation quality I augment the foreground/background data term obtained from this classifier with an image-specific color model based on GrabCut (Rother et al., 2004). The color model (a Gaussian mixture with 40 centers in LAB space) is initialized in each image from the foreground mask output by the classifier and combined with the

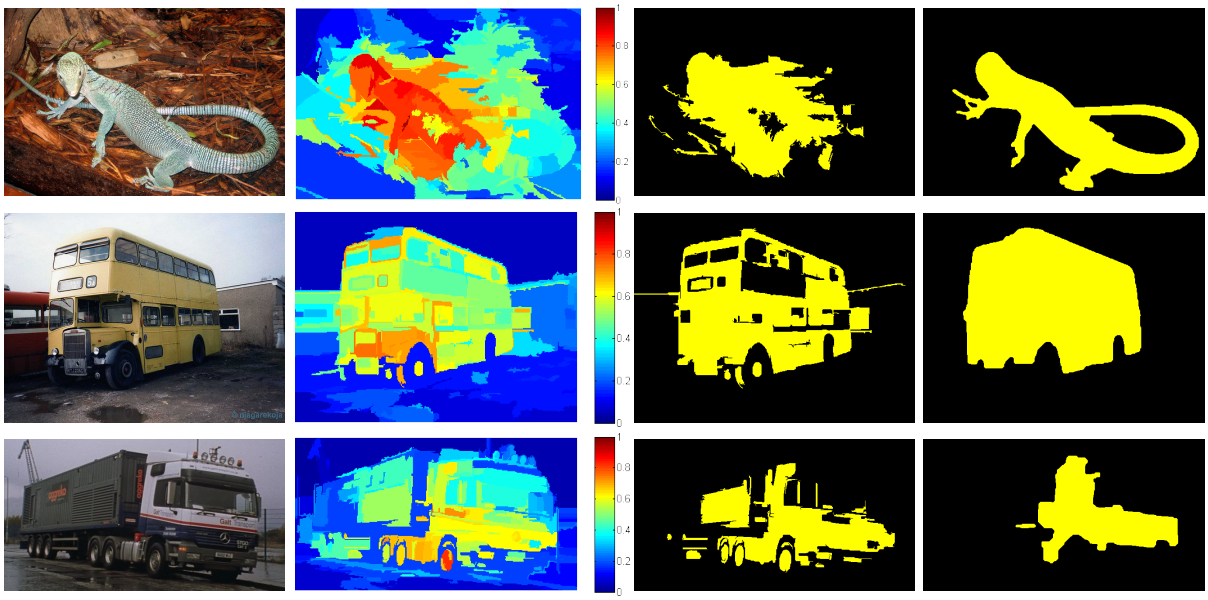

Figure 4.8: Example of foreground masks my color model generates on the CORE dataset. The second column shows the likelihood map from my foreground region classifier; the third column shows the foreground mask resulting from that classifier; and the fourth column shows the foreground map obtained with a GrabCut-like color model. The color model generally improves reasonable initial foreground estimates (first two rows), but makes bad initial estimates worse (last row).

foreground/background data term in a *pixelwise* graph cut energy:

$$E(\mathbf{c}) = \sum_{p_i} [\alpha E_{\text{color}}(p_i, c_i) + E_{\text{data}}(p_i, c_i)] \\ + \lambda \sum_{(p_i, p_j) \in A} E_{\text{smooth}}(c_i, c_j)\,, \tag{4.3}$$

where p_i and p_j are now pixels, A is the set of all adjacent pixels (I use eight-connected neigborhoods), E_{color} is the GrabCut color model term (see Rother et al. (2004) for details), E_{data} is the foreground/background classifier term for the region containing the given pixel (*not* weighted by region area), and E_{smooth} is obtained by scaling my previous smoothing penalty (3.5) by a per-pixel contrast-sensitive constant (see, e.g., Shotton et al.

(2009)). Finally, α is the weight for the GrabCut color model and λ is the smoothing weight. I use $\alpha = 8$ and $\lambda = 8$ in the implementation.

After performing graph cut inference on (4.3) I update the color model based on the new foreground estimate and iterate as in Rother et al. (2004). After four iterations, the foreground rate improves from 64.7% to 76.9%, and the background rate improves from 94.3% to 94.4%. In addition, as shown in Figure 4.8, the subjective quality of the foreground masks generated with this approach becomes much better. Figure 4.9 shows the output of the four-level solver on a few test images with the estimated foreground masks superimposed.

Unfortunately, the role played by foreground masks in my present system is largely cosmetic. Due to the fact that (4.1) and (4.3) are defined on different domains (regions vs. pixels) and use different types of inference, I have not found an easy way to integrate the multi-level labeling with the iterative GrabCut-like foreground/background segmentation. Unifying the two respective objectives into a single optimization is subject for my future work.

4.3 Discussion

This chapter has presented a multi-level inference formulation to simultaneously assign multiple labels to each image region while enforcing agreement between the different label sets. My system is flexible enough to capture a variety of relationships between label sets, including hierarchies and many-to-many. The interaction penalties for different label sets are computed automatically based on label co-occurrence statistics. I have applied the

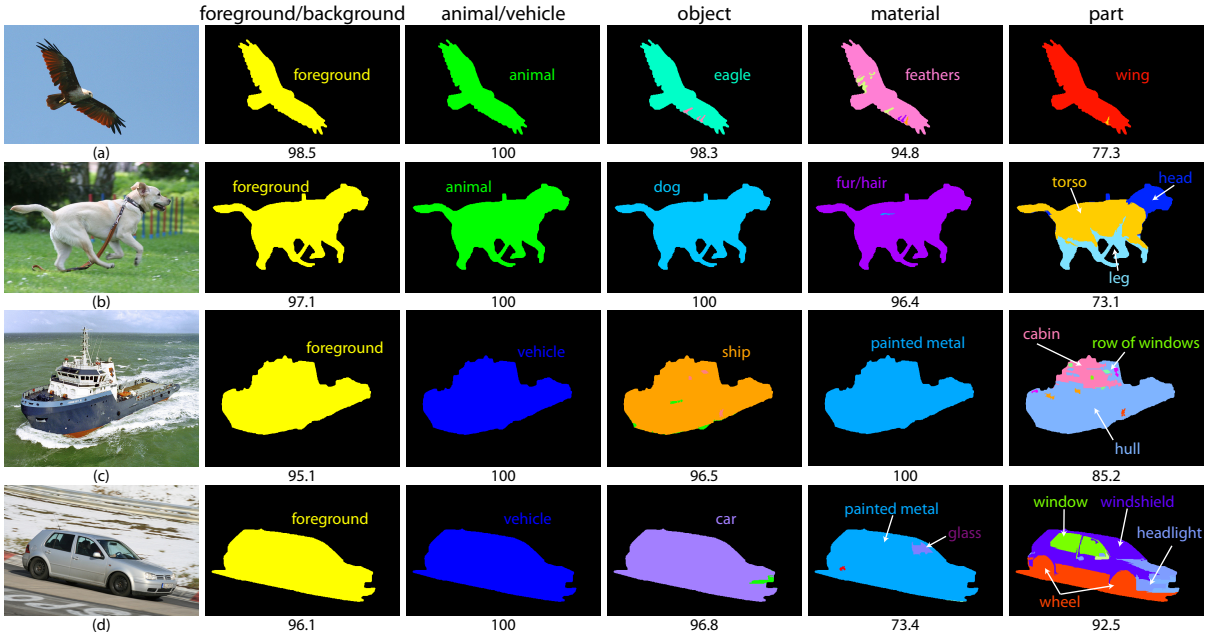

Figure 4.9: The output of my multi-level MRF on the CORE dataset after masking the label sets with my estimated foreground masks. The number under each image is the percentage of ground-truth pixels I label correctly. Note that I am able to correctly classify a number of parts (b,c,d) though those parts do have a tendency to spill into nearby regions of the image.

proposed framework to two challenging and very different datasets and have obtained consistent improvements across every label set, thus demonstrating the promise of a new form of semantic context.

Thus far my parsing system has been purely superpixel based, which works well for more common "stuff" classes as it can reliably capture the texture components that differentiate such classes. "Thing" classes are usually better described by their overall shape, and thus the superpixel representation can not take advantage of key cues for these types of objects. The following chapter presents a method to increase parsing accuracy for "things" by modeling the shape of such classes.

CHAPTER 5: IMAGE PARSING WITH REGIONS AND PER-EXEMPLAR DETECTORS

This chapter returns to the problem of assigning a single semantic label to each pixel but focuses on achieving broader coverage of class labels—the ability to recognize hundreds or thousands of object classes that commonly occur in everyday street scenes and indoor environments. Unfortunately the frequency with which classes appear in realistic scenes is highly non-uniform, making it very hard to correctly classify any but the most common classes. A small number of classes—mainly ones associated with large regions or "stuff," such as road, sky, trees, and buildings (Adelson, 2001)—constitute most image pixels and object instances in the dataset. But a much larger number of "thing" classes—people, cars, dogs, mailboxes, vases, and stop signs–occupy a small percentage of image pixels and have relatively few instances each.

"Stuff" categories have no consistent shape but fairly consistent texture, so they can be adequately handled by image parsing systems based on pixel- or region-level features. However, these systems have difficulty with "thing" categories, which are better characterized by overall shape than local appearance. In order to improve performance on "things," I propose an image parsing system that integrates region-based cues with the promising novel framework of *per-exemplar detectors* or *exemplar-SVMs* (Malisiewicz et al., 2011).

My proposed method is outlined in Figure 5.1. It combines the region-based parser

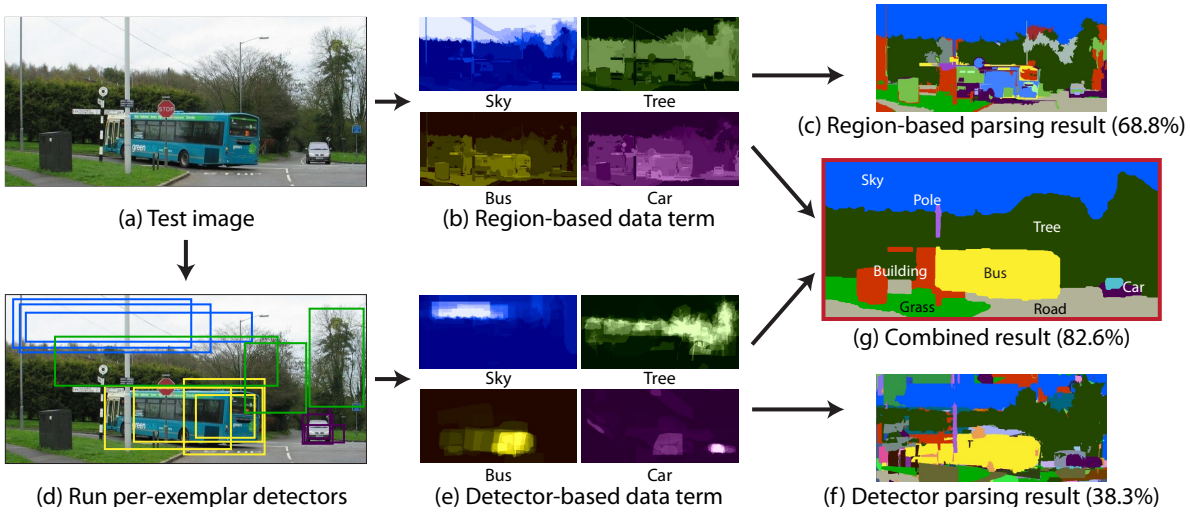

Figure 5.1: Overview and sample result of the combined region- and detector-based approach of this chapter. The test image (a) contains a bus – a relatively rare "thing" class. The region-based parsing system from Chapter 3 computes class likelihoods (b) based on superpixel features, and it correctly identifies "stuff" regions like sky, road, and trees, but is not able to get the bus (c). To find "things" like bus and car, per-exemplar detectors (Malisiewicz et al., 2011) are run on the test image (d) and transfer masks corresponding to detected training exemplars (e). Since the detectors are not well suited for "stuff," the result of detector-based parsing (f) is poor. However, combining region-based and detection-based data terms (g) gives the highest accuracy of all and correctly labels most of the bus and part of the car.

from Chapter 3 with a novel parser based on per-exemplar detectors. Each parser produces a score or *data term* for each possible label at each pixel location, and the data terms are combined using a support vector machine (SVM) to generate the final labeling. This scheme produces state-of-the-art results on the three challenging datasets. This work has first appeared in CVPR 2013 (Tighe and Lazebnik, 2013a).

5.1 Method

This section presents my hybrid image parsing method as illustrated in Figure 5.1. Section 5.1.1 describes the region- and detector-based data terms, and Section 5.1.2 details

the proposed combination method.

5.1.1 Local Data Terms

For each pixel p and class c, the system produces a region- and a detector-based data term, denoted as $E_{\mathrm{R}}(p, c)$ and $E_{\mathrm{D}}(p, c)$, respectively. The region-based data term is defined as

$$E_{\mathrm{R}}(p, c) = L(s_p, c)\,, \tag{5.1}$$

where $L(s_p, c)$ comes from equation 3.1 and s_p is the region containing p.

The novel part of the approach of this section is the computation of the detector-based data term $E_{\mathrm{D}}(p, c)$, which is derived from the per-exemplar framework of Malisiewicz et al. (2011). Following this framework, I train a per-exemplar detector for each labeled object instance in my dataset. While it may seem intuitive to only train detectors for "thing" categories, I train them for *all* categories, including ones seemingly inappropriate for a sliding window approach, such as "sky." As my experiments will demonstrate, this actually yields the best results for the combined region- and detector-based system. I follow the detector training procedure of Malisiewicz et al. (2011), with negative mining done on all training images that do not contain an object of the same class. For the largest LM+SUN dataset I only do negative mining on 1,000 training images most similar to the positive exemplar's image (I have found that using more does not increase the detection accuracy). An alternative training method that relies on linear discriminant analysis (LDA) (Hariharan et al., 2012) could be used in place of the exemplar SVM approach of

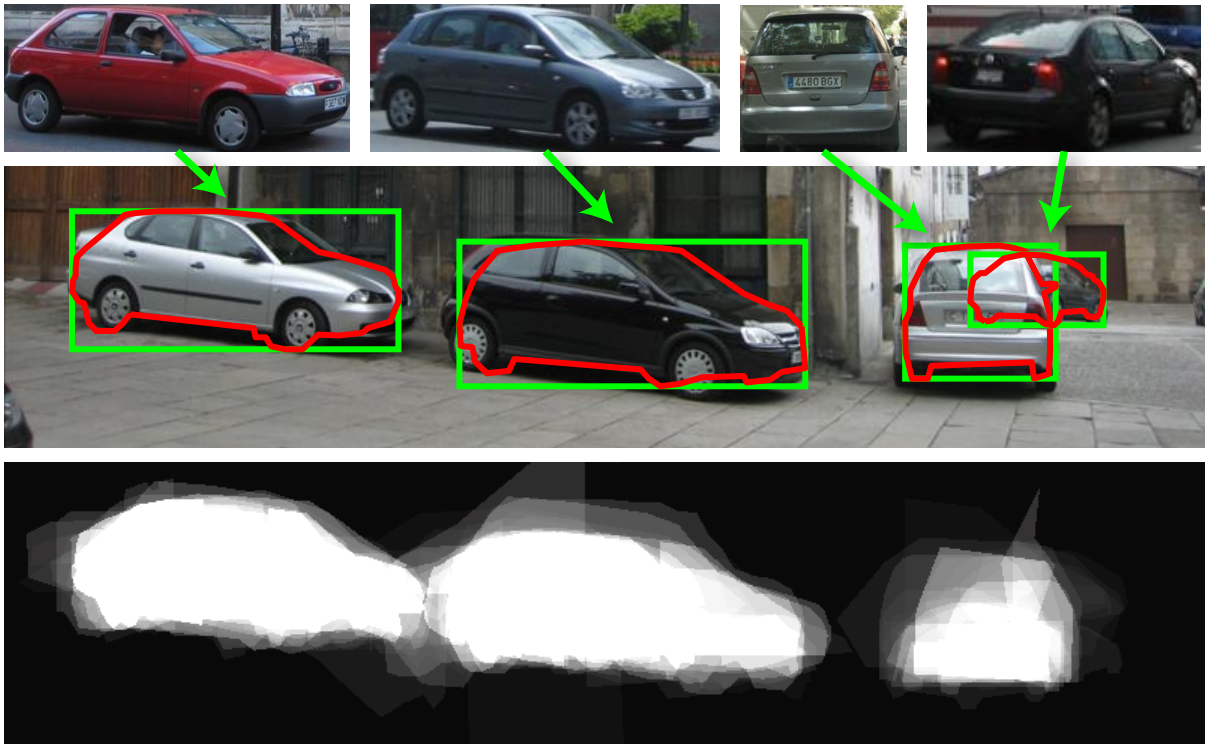

Figure 5.2: Computation of the detector-based data term. For each positive detection (green bounding box) in the test image (middle row) I transfer the mask (red polygon) from the associated exemplar (top) into the test image. The data term for "car" (bottom) is obtained by summing all the masks weighted by their detector responses.

Malisiewicz et al. (2011) to greatly reduce training time and is an area for future work.

At test time, given an image that needs to be parsed, first a retrieval set of globally similar training images is obtained as in Section 3.1.1. Then the detectors associated with the first k instances of each class in that retrieval set are run (the instances are ranked in decreasing order of the similarity of their image to the test image, and different instances in the same image are ranked arbitrarily). Restricting the number of detectors per class to k is done purely to reduce computation; all experiments use $k = 100$. Next, all detections that are above a given threshold t_d are considered positive(I use the negative margin or $t_d = -1$ as suggested in Malisiewicz et al. (2011)). For each detection the

associated object mask is projected into the detected bounding box (Figure 5.2). The *detector-based data term* E_{D} for a class c and pixel p is computed as the sum of all detection masks from that class weighted by their detection scores:

$$E_{\mathrm{D}}(p, c) = \sum_{d \in D_{p,c}} (w_d - t_d)\,, \tag{5.2}$$

where $D_{p,c}$ is the set of all detections for class c whose transferred mask overlaps pixel p and w_d is the detection score of d. Figure 5.1(e) shows some detector-based data terms for the test image of Figure 5.1(a).

Note that the full training framework of Malisiewicz et al. (2011) includes computationally intensive calibration and contextual pooling procedures that are meant to make scores of different per-exemplar detectors more consistent. In my implementation, I have found these steps to be unnecessary, as they are at least partially superseded by the combined region- and detector-based inference scheme described next.

5.1.2 SVM Combination and MRF Smoothing

For each pixel p and each class c in a test image, the parsing systems of Sections 3.1.3 and 5.1.1 produce two data terms, $E_{\mathrm{R}}(p, c)$ and $E_{\mathrm{D}}(p, c)$, as defined by eqs. (5.1) and (5.2). For a dataset with C classes, concatenating these values gives us a $2C$-dimensional feature vector at each pixel. Next, C one-vs-all SVMs are trained, each of which takes as input the $2C$-dimensional feature vectors and returns final per-pixel scores for a given class c.

Training data for each SVM is generated by running region- and detector-based parsing on the entire training set using a leave-one-out method: for each training image a retrieval set of similar training images is obtained, regions are matched to generate $E_{\mathrm{R}}(p, c)$, and the per-exemplar detectors from the retrieval set are run to generate $E_{\mathrm{D}}(p, c)$. Unfortunately, the resulting amount of data is huge: my largest dataset has over nine billion pixels, which would require 30 terabytes of storage. To make SVM training feasible, the data must be subsampled—a tricky task given the unbalanced class frequencies in my many-category datasets.

The data could be subsampled uniformly (i.e., reduce the number of points by a fixed factor without regard to class labels). This preserves the relative class frequencies, but in practice it heavily biases the classifier towards the more common classes. Conversely, subsampling the data so that each class has a roughly equal number of points produces a bias towards the rare classes. I have found that combining these two schemes in a 2:1 ratio achieves a good trade-off on all my datasets. That is, 67% of the training data is obtained by uniform sampling and 33% by even per-class sampling. All SVMs are trained on 250,000 points—using more did not significantly improve performance for any of my setups.

For training one-vs-all SVMs, each feature dimension is normalized by its standard deviation and fivefold cross-validation is used to find the regularization constant. Another important aspect of the implementation is the choice of the SVM kernel. As will be shown in Section 5.2, the linear kernel already does quite well, but I can obtain further improvements with the radial basis function (RBF) kernel. Since it is infeasible to train a

nonlinear SVM with the RBF kernel on my largest dataset, I approximate it by training a linear SVM on top of the random Fourier feature embedding (Rahimi and Recht, 2007). The dimensionality of the embedding is set to 4,000 and the kernel bandwidth is found using fivefold cross-validation. Experiments on my two smaller datasets confirm the quality of the approximation (Table 5.2).

The resulting SVMs produce C responses at each pixel. Let $E_{\text{SVM}}(p_i, c_i)$ denote the response of the SVM for class c_i at pixel p_i. To obtain the final labeling, the highest-scoring label at each pixel can be assigned to that pixel, but this produces noisy results. Instead, the labeling is smoothed with an MRF energy function similar to Liu et al. (2011b) and Shotton et al. (2008) defined over the field of pixel labels $\mathbf{c}$:

$$\begin{aligned} J(\mathbf{c}) = &\sum_{p_i \in I} \max[0, M - E_{\text{SVM}}(p_i, c_i)] \\ &+ \lambda \sum_{(p_i, p_j) \in \epsilon} E_{\text{pix\_smooth}}(c_i, c_j)\,, \end{aligned} \tag{5.3}$$

where I is the set of all pixels in the image, ϵ is the set of adjacent pixels, M is the highest expected value of the SVM response (about 10 on my data), λ is a smoothing constant (I set $\lambda = 16$), and $E_{\text{pix\_smooth}}(c_i, c_j)$ imposes a penalty when two adjacent pixels (p_i, p_j) are similar but are assigned different labels (c_i, c_j) (see equation 8 in Liu et al. (2011b)). MRF inference is performed using α-expansion (Boykov and Kolmogorov, 2004; Kolmogorov and Zabih, 2004).

5.2 Evaluation

5.2.1 Datasets

I perform experiments on the three datasets detailed in Sections 3.2.1, 3.2.2 and 3.3.2: SIFT Flow (Liu et al., 2011b), LM+SUN (Tighe and Lazebnik, 2013b), and CamVid (Brostow et al., 2008).

While CamVid is not my target dataset type, I use it for comparison with a number of recent approaches (Brostow et al., 2008; Floros et al., 2011; Ladický et al., 2010b; Sturgess et al., 2009; Zhang et al., 2010). Unlike SIFT Flow and LM+SUN, CamVid does not have object polygons, only per-pixel labels. For training detectors, I fit a bounding box and a segmentation mask to each connected component of the same label type. Thus, if multiple object instances (e.g., cars) overlap, they are treated as one exemplar. Following my video parsing system from Section 3.3, I segment the video using the method of Grundmann et al. (2010), and use boosted decision tree classifiers instead of nonparametric likelihood estimates. To obtain training data for the SVM, I compute the responses of the boosted decision tree classifiers on the same images on which they were trained (I have found this to work better than cross-validation on this dataset). I do not enforce any spatio-temporal consistency in the final labeling as described in 3.3.1.

5.2.2 Experiments

First, I analyze the contributions of individual components of my system. In particular, one may wonder whether the power of the approach of this chapter truly lies in combining

	SIFT Flow	LM+SUN	CamVid
Detector ML	65.1 (25.8)	33.0 (14.1)	61.2 (45.5)
Detector SVM	62.5 (25.4)	46.1 (12.0)	61.4 (47.0)
Detector SVM MRF	71.1 (26.7)	52.5 (11.3)	63.8 (47.3)
Region ML	74.1 (30.2)	51.5 (7.5)	82.7 (51.2)
Region SVM	75.0 (35.9)	56.3 (6.7)	81.4 (55.7)
Region SVM MRF	77.7 (32.8)	58.3 (5.9)	83.5 (55.7)
Region + Thing SVM	74.4 (36.9)	58.5 (14.1)	82.4 (60.0)
Region + Thing SVM MRF	77.5 (35.7)	60.0 (12.9	84.2 (59.5)
Combined SVM	75.6 (**41.1**)	59.6 (**15.5**)	82.3 (62.1)
Combined SVM MRF	**78.6** (39.2)	**61.4** (15.2)	**84.0** (**62.2**)

Table 5.1: Comparison of different data terms. Per-pixel classification rate is listed first, followed by the average per-class rate in parentheses. All SVMs use the approximate RBF embedding. Detector ML and Region ML directly assign the highest-scoring label at each pixel based on the respective data terms, while Detector SVM and Region SVM use SVM outputs trained on the individual data terms. Region + Thing uses the SVM trained on the full region data term and the subset of the detector data term corresponding to "thing" classes.

detector- and region-based cues, or whether most of the performance gain comes from the extra layers of SVM and MRF inference. Table 5.1 shows the performance of various combinations of region- and detector-based data terms with and without SVM training, with and without MRF smoothing. The region-based data term obtains higher per-pixel accuracy than the detector-based one on all three datasets, and higher per-class accuracy on SIFT Flow and CamVid. On the LM+SUN dataset, which has the largest number of rare "thing" classes, the detector-based data term actually obtains higher per-class accuracy than the region-based one. While the SVM can sometimes improve performance when applied to the individual data terms, applying it to their combination gives by far the biggest and most consistent improvements. As observed in Chapter 3, MRF inference further raises the per-pixel rate, but often lowers the per-class rate by

	SIFT Flow	LM+SUN	CamVid
Linear	75.4 (40.0)	57.2 (16.6)	82.4 (60.7)
Linear MRF	77.5 (40.2)	59.5 (15.9)	83.8 (60.7)
Approx. RBF	75.6 (41.1)	59.6 (15.5)	82.3 (62.1)
Approx. RBF MRF	78.6 (39.2)	61.4 (15.2)	83.9 (62.5)
Exact RBF	75.4 (41.6)	N/A	82.3 (61.9)
Exact RBF MRF	77.6 (42.0)	N/A	84.0 (62.2)

Table 5.2: Comparison of different SVM kernels. Per-pixel classification rate is listed first, followed by the average per-class rate in parentheses. The RBF kernel has a slight edge over the linear kernel, and the approximate RBF embedding of (Rahimi and Recht, 2007) has comparable performance to the exact nonlinear RBF. Note that training the exact RBF on the largest LM+SUN dataset was computationally infeasible.

smoothing away some of the smaller objects.

Because part of my motivation for incorporating detectors is to improve performance on the "thing" classes, I want to know what will happen if I train detectors only on "things"—if detectors are completely inappropriate for "stuff," then not using them on "stuff" may improve accuracy, not to mention speed up the system considerably. The "Region + Thing" section of Table 5.1 shows the performance of the SVM trained on the full region-based data term and the subset of the detector-based data term corresponding only to "thing" classes (specified manually). Interestingly, the results for this setup are weaker than those of the full combined system using both "thing" and "stuff" detectors.

Next, Table 5.2 compares SVMs with the linear kernel, approximate RBF embedding (Rahimi and Recht, 2007), and exact nonlinear RBF. The linear kernel may be a better choice if speed is a concern, but the approximate and exact RBF are able to boost performance by 1-2%. All subsequent figures and tables will report only the SVM results with the approximate RBF.

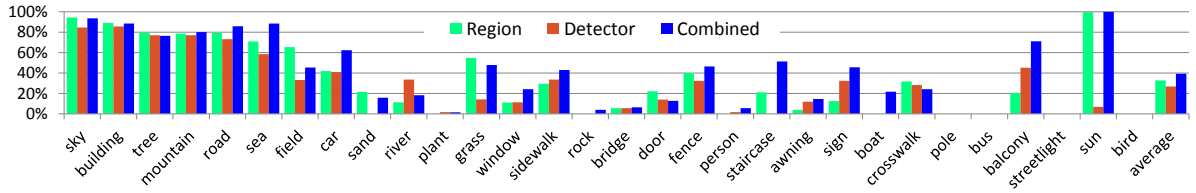

Figure 5.3: Classification rates of individual classes (ordered from most to least frequent) on the SIFT Flow dataset for region-based, detector-based, and combined parsing. All results include SVM and MRF smoothing.

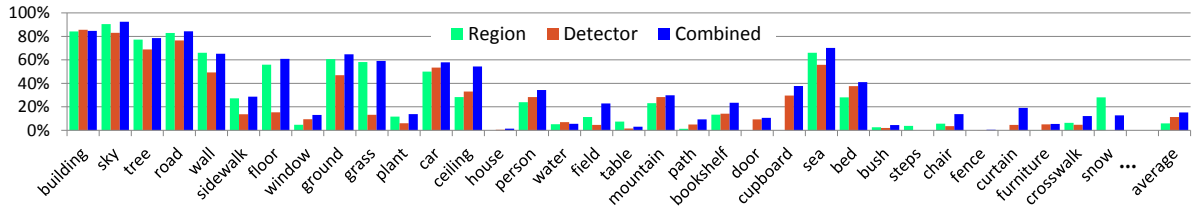

Figure 5.4: Classification rates of the most common individual classes (ordered from most to least frequent) on the LM+SUN dataset for region-based, detector-based, and combined parsing. All results include SVM and MRF smoothing.

Figures 5.3 and 5.4 show the per-class rates of my system on the most common classes in the SIFT Flow and LM+SUN datasets, respectively. As expected, adding detectors significantly improves many "thing" classes (including car, sign, and balcony) but also some "stuff" classes (road, sea, sidewalk, fence). Figure 5.5 gives a close-up look at my performance on many small object categories, and Figure 5.6 shows several parsing examples on the LM+SUN dataset.

Table 5.3 compares my combined system to a number of state-of-the-art approaches on the SIFT Flow dataset. The system outperforms them, in many cases beating the average per-class rate by up to 10% while maintaining or exceeding the per-pixel rates. The one exception is the system of Farabet et al. (2012) when tuned for balanced per-class rates, but their per-pixel rate is much lower than my result in this case. When their

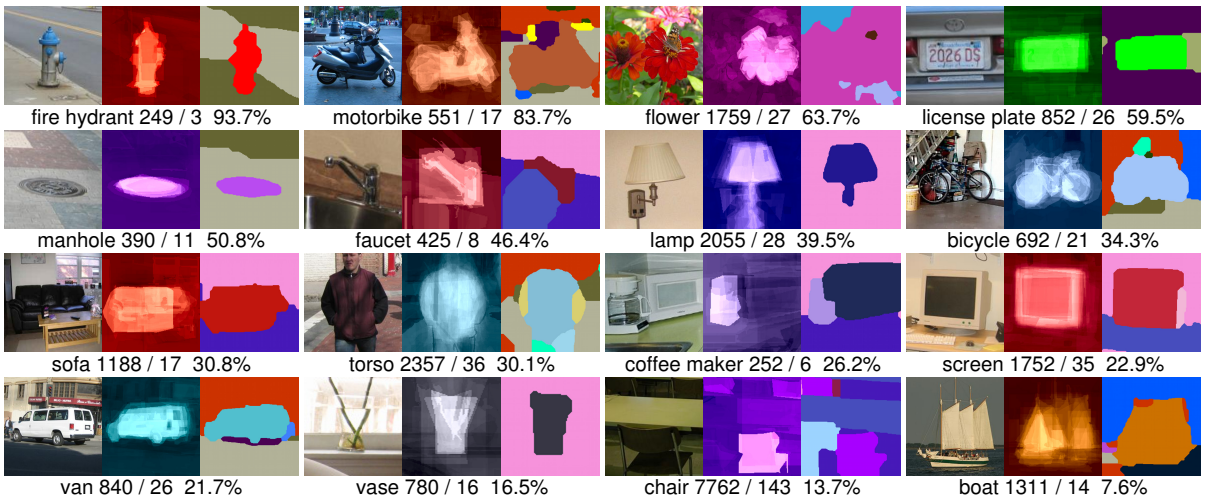

Figure 5.5: Examples of "thing" classes on LM+SUN. For each class I show a crop of an image, the SVM combined output, and the smoothed final result. The caption for each class shows: (# of training instances of that class) / (# of test instances) (per-pixel rate on the test set)%. Best viewed in color.

system is tuned to a per-pixel rate similar to mine, their average per-class rate drops significantly below mine.

On LM+SUN, which has an order of magnitude more images and labels than SIFT Flow, the only previously reported results are from my base region-based system (Chapter 3). As Table 5.4 shows, by augmenting the region-based term with a novel detector-based data term and SVM inference, the system is able to raise the per-pixel rate from 54.9% to 61.4% and the per-class rate from 7.1% to 15.2%.

Table 5.5 compares my per-class rates on the CamVid dataset to a number of recent methods. When compared to my region-based system (Chapter 3), performance is improved for every class except for building and sky, towards which the region-based parser seems to be overly biased. Furthermore, this system is able to match the state-of-the-art method of Ladický et al. (2010b), which incorporates DPM detectors, bounding box

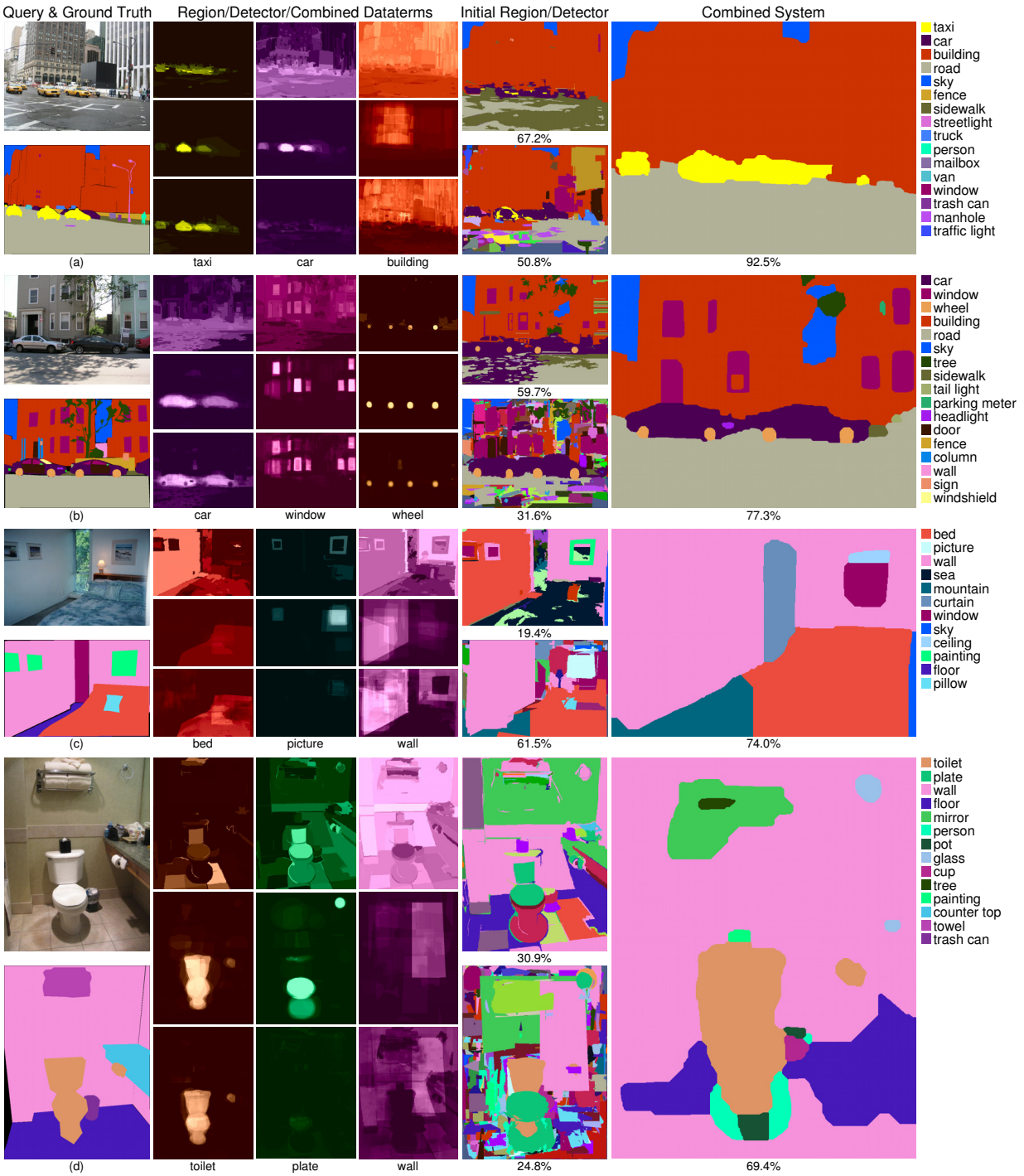

Figure 5.6: Example results on the LM+SUN dataset (best viewed in color). First column: query image (top) and ground truth (bottom). Second through fourth columns: region-based data term (top), detector-based data term (middle), and SVM combination (bottom) for three selected class labels. Fifth column: region-based parsing results (top) and detector-based parsing results (bottom) without SVM or MRF smoothing. Rightmost column: smoothed combined output.

SIFT Flow	Per-Pixel	Per-Class
Combined MRF	78.6	39.2
Tighe and Lazebnik (Tighe and Lazebnik, 2013b)	77.0	30.1
Liu et al. (Liu et al., 2011b)	76.7	N/A
Farabet et al. (Farabet et al., 2012)	78.5	29.6
Farabet et al. (Farabet et al., 2012) balanced	74.2	46.0
Eigen and Fergus (Eigen and Fergus, 2012)	77.1	32.5
Myeong et al. (Myeong et al., 2012)	77.1	32.3

Table 5.3: Comparison to state-of-the-art on the SIFT Flow dataset.

LM+SUN	Per-Pixel	Per-Class
Combined MRF	61.4	15.2
Outdoor Images	65.5	15.3
Indoor Images	46.3	12.2
Base system	54.9	7.1
Outdoor Images	60.8	7.7
Indoor Images	32.1	4.8

Table 5.4: Comparison to my base system on the LM+SUN dataset with results broken down by outdoor and indoor test images.

segmentation, and a complex CRF model. Figure 5.7 shows the output of my system on example CamVid images.

5.2.3 Running Time

Finally, I examine the computational requirements of my system on my largest dataset, LM+SUN, by timing my MATLAB implementation (feature extraction and file I/O excluded) on a six-core 3.4 GHz Intel Xeon processor with 48 GB of RAM. There are a total of 354,592 objects in the training set, and a per-exemplar detector is trained for each of them. The average training time per detector is 472.3 seconds; training all of them would require 1,938 days on a single CPU, but instead training is performed on a

	Building	Tree	Sky	Car	Sign-Symbol	Road	Pedestrian
Combined MRF	83.1	73.5	94.6	78.1	**48**	96	**58.6**
Chapter 3	**87.0**	67.1	96.9	62.7	30.1	95.9	14.7
Brostow et al. (2008)	46.2	61.9	89.7	68.6	42.9	89.5	53.6
Sturgess et al. (2009)	84.5	72.6	**97.5**	72.7	34.1	95.3	34.2
Zhang et al. (2010)	85.3	57.3	95.4	69.2	46.5	**98.5**	23.8
Floros et al. (2011)	80.4	76.1	96.1	86.7	20.4	95.1	47.1
Ladický et al. (2010a)	81.5	**76.6**	96.2	**78.7**	40.2	93.9	43.0

	Fence	Column-Pole	Sidewalk	Bicyclist		Per-class	Per-pixel
Combined MRF	32.8	5.3	71.2	**45.9**		**62.5**	**83.9**
Chapter 3	17.9	1.7	70.0	19.4		51.2	83.3
Brostow et al. (2008)	46.6	0.7	60.5	22.5		53.0	69.1
Sturgess et al. (2009)	45.7	8.1	77.6	28.5		59.2	**83.8**
Zhang et al. (2010)	44.3	**22.0**	38.1	28.7		55.4	82.1
Floros et al. (2011)	47.3	8.3	79.1	19.5		59.6	83.2
Ladický et al. (2010a)	**47.6**	14.3	**81.5**	33.9		**62.5**	**83.8**

Table 5.5: Per-class performance on the CamVid (Brostow et al., 2008) dataset.

512-node cluster in approximately four days. Leave-one-out parsing of the training set (see below for average region- and detector-based parsing times per image) takes 939 hours on a single CPU, or about two hours on the cluster. Next, training a set of 232 one-vs-all SVMs takes a total of one hour on a single machine for the linear SVM and ten hours for the approximate RBF. Note that the respective feature dimensionalities are 464 and 4,000; this nearly tenfold dimensionality increase accounts for the tenfold increase in running time. Tuning the SVM parameters by fivefold cross-validation on the cluster only increases the training time by a factor of two.

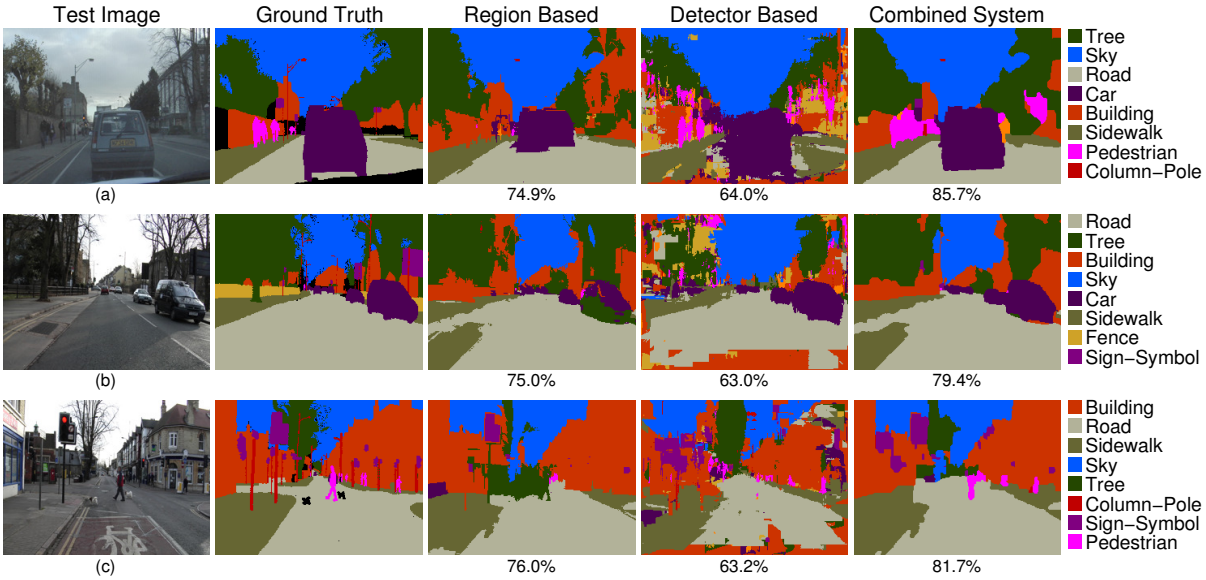

Figure 5.7: Example results on the CamVid dataset. The region- and detector-based results are shown without SVM or MRF smoothing; the final output has both. Notice how the detectors are able to complete the car in (a) and (b). In (c) I are able to correctly parse a number of pedestrians and signs.

At test time, the region-based parsing takes an average of 27.5 seconds per image. The detector-based parser runs an average of 4,842 detectors per image in 47.4 seconds total. The final combination (SVM testing) step takes an average of 8.9 seconds for the linear kernel and 124 seconds for the approximate RBF (once again, the tenfold increase in feature dimensionality and the overhead of computing the embedding account for the increase in running time). MRF inference takes only 6.8 seconds per image.

5.3 Discussion

The combined system presented in this chapter achieves very promising results, but at a considerable computational cost. Reducing this cost is an important future research direction. To speed up training of per-exemplar detectors, I plan to try the whitened

HOG method of Hariharan et al. (2012). At test time, I would like to to reduce the number of detectors that need to be run per image. As shown in Table 5.1, doing this naively, e.g., by running only "thing" detectors, lowers the accuracy. Instead, I want to develop methods for dynamically selecting detectors for each test image based on context. Also, SVM testing with the approximate RBF embedding imposes a heavy overhead in my current implementation. However, this is a nuisance that can be resolved with better choice of embedding dimensionality and more optimized code.

Ultimately, I want my system to function on *open universe* datasets, such as LabelMe (Russell et al., 2008), that are constantly evolving and do not have a pre-defined list of classes of interest. The region-based component of my system from Chapter 3 already has this property. In principle, per-exemplar detectors are also compatible with the open-universe setting, since they can be trained independently as new exemplars come in. The SVM combination step (Section 5.1.2) is the only one that relies on batch offline training (including leave-one-out parsing of the entire training set). In the future, I plan to investigate online methods for this step.

CHAPTER 6: DISCUSSION

I have presented a retrieval-based image parsing methodology as a more scalable and accurate alternative to image parsing methods based on offline training. My approach provides a rich form of scene understanding that infers multiple types of labels with broad coverage of classes. My novel retrieval-based system was outlined in Section 3.1, with the retrieval method described in Section 3.1.1 and the superpixel-level matching described in Section 3.1.3. I described my novel method to simultaneously label both semantic and geometric label types in Section 3.1.5. I then presented a novel technique to learn the relationships between an arbitrary number of label types with many-to-many relationships and presented a efficient inference framework once such relationships are learned (Section 4.1). Finally in Section 5.1, I greatly broadened the coverage of classes my system successfully identifies by presenting a new, elegant, per-exemplar based approach to incorporate detectors in an image parsing system, which, unlike previous methods, can easily scale to hundreds of classes.

6.1 Future directions

While I have been successful in improving the accuracy and scalability of image parsing systems, I have not "solved" the image parsing problem by any means. In this section I will discuss directions for future work in image parsing.

All of my work relies on a retrieval-based methodology, but I have only explored some

of the simplest forms of retrieval sets: those based on nearest neighbors according to a fixed set of global image features. My system analysis in Section 3.2.3 showed that a more accurate retrieval set would also greatly improve the parsing performance. To that end, exploring more advanced methods for obtaining a retrieval set could be quite beneficial. One could learn per-exemplar distance functions for each image in the training set, or one could learn a better feature representation from a convolution neural network. In general, it seems feasible that from the high level of supervision available in the training set more powerful matching criteria could be learned.

One of the limitations of my multi-level inference in Chapter 4 was the poor classification accuracy for attached objects. However, my work in Chapter 5 boosted the performance of "thing" classes, including those corresponding to attached objects (e.g., wheels, windows). It might be interesting then to combine the systems from Chapter 4 and Chapter 5 as it would seem they would be complementary. Pushing the idea of multiple label types further, it would be interesting examining image parsing using even more complex label structures, like the noun hierarchy of WordNet (Fellbaum, 1998). In this case, instead of discrete sets of labels, where each region must be assigned a single label from each set, a region would be assigned a label somewhere down the hierarchy depending on the evidence available similar to Deng et al. (2012).

Scene understanding does not stop at inferring labels in the 2D images plane. The work I have presented may be able to correctly label the car pixels in an image, but if there are multiple overlapping cars there is no mechanism for my parsing systems to indicate which is in front of the other, or even where the boundary between the two cars

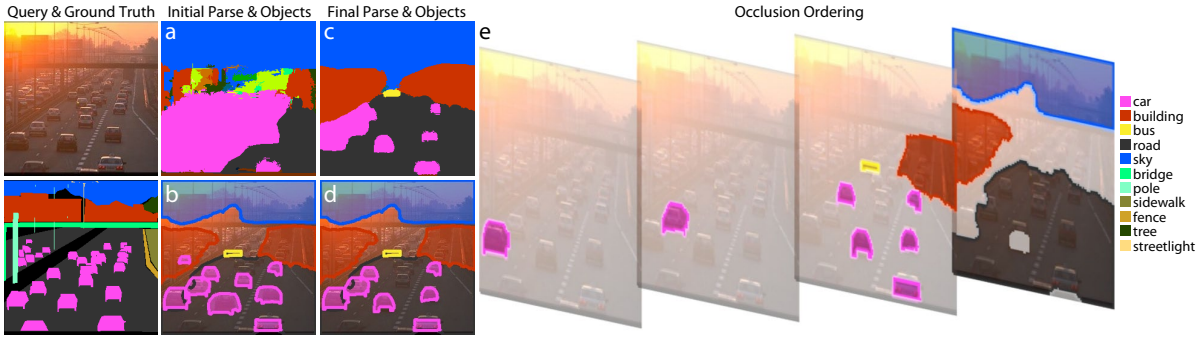

Figure 6.1: Example of separating object instances. Starting with the output from the system of Chapter 5 (a) we could separate the car blob into individual instances (b). These predicted object instance could be used to refine the pixel labels (c), and those labels could be used to refine the instance predictions (d). Finally, it would be beneficial to infer the occlusion ordering of the car objects (e).

lies. In the future I plan to investigate parsing a scene into its coherent objects and assigning a depth ordering to the inferred objects. I hope I can use the fairly accurate object silhouettes found by the per-exemplar detectors to split object instances apart. A illustration of how this could work and why this would be useful can be seen in Figure 6.1. By inferring objects and their depth ordering, we would be one step closer to true scene understanding.

The ultimate goal for computer vision is to determine the full 3D structure of the scene, all the objects and parts it decomposes into, and the interactions between these parts. Many researchers believe that to create a system with these capabilities we must solve the more general problem of artificial intelligence. It is yet to be seen if by solving the computer vision problem we will create artificial intelligence, but I am very excited to continue exploring these possibilities with my fellow researchers.

BIBLOGRAPHY

Adelson, E. (2001). On seeing stuff: The perception of materials by humans and machines. In *Proceedings of the SPIE*, number 4299, pages 1–12.

Arbelaez, P., Hariharan, B., Gu, C., Gupta, S., and Bourdev, L. (2012). Semantic segmentation using regions and parts. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Barla, A., Odone, F., and Verri, A. (2003). Histogram intersection kernel for image classification. In *Proceedings International Conference on Image Processing*, volume 3. IEEE.

Bourdev, L. and Malik, J. (2009). Poselets: Body part detectors trained using 3d human pose annotations. In *Proceedings IEEE International Conference Computer Vision*.

Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–37.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Efficient approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1222–1239.

Brostow, G. J., Shotton, J., Fauqueur, J., and Cipolla, R. (2008). Segmentation and recognition using structure from motion point clouds. In *Proceedings European Conference Computer Vision*.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Deng, J., Berg, A., Li, K., and Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In *Proceedings European Conference Computer Vision*.

Deng, J., Krause, J., Berg, A. C., and Li, F.-F. (2012). Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

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

Eigen, D. and Fergus, R. (2012). Nonparametric image parsing using adaptive neighbor sets. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2012). Scene parsing with multiscale feature learning, purity trees, and optimal covers. In *Proceedings of the International Conference on Machine Learning*.

Farhadi, A., Endres, I., and Hoiem, D. (2010). Attribute-centric recognition for cross-category generalization. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Fellbaum, C. (1998). Wordnet: An electronic lexical database. In *Cambridge, MA: MIT Press.*

Felzenszwalb, P., Mcallester, D., and Ramanan, D. (2008). A discriminatively trained , multiscale , deformable part model. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition.*

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International Journal of Computer Vision*, 2(2):1–26.

Floros, G., Rematas, K., and Leibe, B. (2011). Multi-class image labeling with top-down segmentation and generalized robust pn potentials. In *Proceedings of the British Machine Vision Conference.*

Fu, K. S. and Albus, J. E. (1982). *Syntactic pattern recognition and applications*, volume 4. Prentice-Hall Englewood Cliffs, NJ.

Galleguillos, C. and Belongie, S. (2010). Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722.

Galleguillos, C., Mcfee, B., Belongie, S., and Lanckriet, G. (2010). Multi-class object localization by combining local contextual interactions. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition.*

Galleguillos, C., Rabinovich, A., and Belongie, S. (2008). Object categorization using co-occurrence, location and appearance. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition.*

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Gionis, A., Indyk, P., and Motwani, R. (1999). Similarity search in high dimensions via hashing. In *Proceedings of the 25th Very Large Database.*

Gould, S., Fulton, R., and Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. In *Proceedings IEEE International Conference Computer Vision.*

Griffin, G. and Perona, P. (2008). Learning and using taxonomies for fast visual categorization. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition.*

Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010). Efficient hierarchical graph-based video segmentation. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition.*

Guo, R. and Hoiem, D. (2012). Beyond the line of sight: labeling the underlying surfaces. In *Proceedings European Conference Computer Vision.*

Gupta, A. and Davis, L. S. (2008). Beyond nouns: Exploiting prepositions and comparative adjectives for learning visual classifiers. In *Proceedings European Conference Computer Vision.*

Gupta, A., Efros, A. A., and Hebert, M. (2010). Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *Proceedings European Conference Computer Vision*.

Gupta, A., Satkin, S., Efros, A. A., and Hebert, M. (2011). From 3d scene geometry to human workspace. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Hanson, A. R. and Riseman, E. M. (1978). Visions: A computer system for interpreting scenes. *Computer vision systems*, 78.

Hariharan, B., Malik, J., and Ramanan, D. (2012). Discriminative decorrelation for clustering and classification. In *Proceedings European Conference Computer Vision*.

Hays, J. and Efros, A. A. (2007). Scene completion using millions of photographs. *ACM Transactions on Graphics (SIGGRAPH 2007)*, 26(3).

Hays, J. and Efros, A. A. (2008). Im 2 gps : estimating geographic information from a single image. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

He, X., Zemel, R. S., and Carreira-Perpinan, M. A. (2004). Multiscale conditional random fields for image labeling. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Hedau, V., Hoiem, D., and Forsyth, D. (2009). Recovering the spatial layout of cluttered rooms. In *Proceedings IEEE International Conference Computer Vision*.

Hedau, V., Hoiem, D., and Forsyth, D. (2010). Thinking inside the box: Using appearance models and context based on room geometry. In *Proceedings European Conference Computer Vision*.

Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. In *Proceedings European Conference Computer Vision*.

Hoiem, D., Efros, A. A., and Hebert, M. (2005). Automatic photo pop-up. In *ACM SIGGRAPH*.

Hoiem, D., Efros, A. A., and Hebert, M. (2006). Putting objects in perspective. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Hoiem, D., Efros, A. A., and Hebert, M. (2007). Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1).

Indyk, P. and Motwani, R. (1999). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of 30th Symposium on Theory of Computing*.

Janoch, A., Karayev, S., Jia, Y., Barron, J. T., Fritz, M., Saenko, K., and Darrell, T. (2011). A category-level 3-d object dataset : Putting the kinect to work. In *Proceedings IEEE International Conference Computer Vision Workshop*.

Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–59.

Kulkarni, G., Premraj, V., Ordonez, V., Dhar, S., Li, S., Choi, Y., Berg, A. C., and Berg, T. L. (2013). Babytalk: Understanding and generating simple image descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Kumar, N., Berg, A., Belhumeur, P. N., and Nayar, S. (2011). Describable visual attributes for face verification and image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):1962–1977.

Kumar, S. and Hebert, M. (2006). Discriminative random fields. *International Journal of Computer Vision*, 68(2):179–201.

Ladický, L., Russell, C., Kohli, P., and Torr, P. H. (2009). Associative hierarchical crfs for object class image segmentation. In *International Conference on Computer Vision*.

Ladický, L., Russell, C., Kohli, P., and Torr, P. H. (2010a). Graph cut based inference with co-occurrence statistics. In *European Conference on Computer Vision*.

Ladický, L., Sturgess, P., Alahari, K., Russell, C., and Torr, P. H. (2010b). What, where & how many? combining object detectors and crfs. In *European Conference on Computer Vision*.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011). A scalable tree-based approach for joint object and pose recognition. In *Twenty-Fifth Conference on Artificial Intelligence*.

Lampert, C. H., Nickisch, H., and Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *Proceedings IEEE International Conference Computer Vision*.

Lazebnik, S. and Raginsky, M. (2009). An empirical bayes approach to contextual region classification. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Lazebnik, S., Schmid, C., and Ponce, J. (2006). Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Lee, D. C., Gupta, A., Hebert, M., and Kanade, T. (2010). Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *Advances in Neural Information Processing Systems*.

Leibe, B., Leonardis, A., and Schiele, B. (2008). Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(13):259289.

Liu, C., Yuen, J., and Torralba, A. (2011a). Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2368–2382.

Liu, C., Yuen, J., and Torralba, A. (2011b). Sift flow: dense correspondence across scenes and its applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):978–94.

Malisiewicz, T. and Efros, A. A. (2008). Recognition by association via learning per-exemplar distances. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Malisiewicz, T., Gupta, A., and Efros, A. A. (2011). Ensemble of exemplar-SVMs for object detection and beyond. In *ICCV*.

Marszalek, M. and Schmid, C. (2007). Semantic hierarchies for visual object recognition. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Myeong, H. J., Chang, Y., and Lee, K. M. (2012). Learning object relationships via graph-based context model. *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Nowozin, S., Carsten Rother, Bagon, S., Sharp, T., Yao, B., and Kohli, P. (2011). Decision tree fields. In *Proceedings IEEE International Conference Computer Vision*.

Ohta, Y. (1985). *Knowledge-based interpretation of outdoor natural color scenes*, volume 4. Morgan Kaufmann.

Ohta, Y., Kanade, T., and Sakai, T. (1978). An analysis system for scenes containing objects with substructures. In *Proceedings of the Fourth International Joint Conference on Pattern Recognitions*, pages 752–754.

Oliva, A. and Torralba, A. (2006). Building the gist of a scene: the role of global image features in recognition. *Visual Perception, Progress in Brain Research*, 155:23–36.

Ordonez, V., Kulkarni, G., and Berg, T. L. (2011). Im2text: Describing images using 1 million captioned photographs. In *Proceedings Neural Information Processing Systems*.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Proceedings IEEE International Conference Computer Vision*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Proceedings Neural Information Processing Systems*.

Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *Proceedings IEEE International Conference Computer Vision*.

Roberts, L. (1965). Machine perception of 3-d solids.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM SIGGRAPH*.

Russell, B. C. and Torralba, A. (2009). Building a database of 3d scenes from user annotations. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Russell, B. C., Torralba, A., Liu, C., Fergus, R., and Freeman, W. T. (2007). Object recognition by scene alignment. In *Proceedings Neural Information Processing Systems*.

Russell, B. C., Torralba, A., Murphy, K. P., and Freeman, W. T. (2008). Labelme : a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173.

Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Shotton, J., Winn, J. M., Rother, C., and Criminisi, A. (2009). Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23.

Silberman, N. and Fergus, R. (2011). Indoor scene segmentation using a structured light sensor. In *Proceedings IEEE International Conference Computer Vision Workshop*.

Silberman, N., Hoiem, D., Kohli, P., and Fergus, R. (2012). Indoor segmentation and support inference from rgbd images. In *Proceedings European Conference Computer Vision*.

Singh, G. and Košecká, J. (2013). Nonparametric scene parsing with adaptive feature relevance and semantic context. Proceedings IEEE Conference Computer Vision and Pattern Recognition.

Socher, R., Lin, C. C.-Y., Ng, A. Y., and Manning, C. D. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the International Conference on Machine Learning*.

Sturgess, P., Alahari, K., Ladicky, L., and Torr, P. H. S. (2009). Combining appearance and structure from motion features for road scene understanding. *British Machine Vision Conference*.

Tighe, J. and Lazebnik, S. (2010). SuperParsing: Scalable nonparametric image parsing with superpixels. In *Proceedings European Conference Computer Vision*.

Tighe, J. and Lazebnik, S. (2011). Understanding scenes on many levels. In *Proceedings IEEE International Conference Computer Vision*.

Tighe, J. and Lazebnik, S. (2013a). Finding things: Image parsing with regions and per-exemplar detectors. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Tighe, J. and Lazebnik, S. (2013b). SuperParsing: Scalable nonparametric image parsing with superpixels. *International Journal of Computer Vision*, 101(2):329–349.

Torralba, A., Fergus, R., and Freeman, W. T. (2008). 80 million tiny images: a large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(11):1958–1970.

Tu, Z. (2008). Auto-context and its application to high-level vision tasks. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Tu, Z., Chen, X., Yuille, A. L., and Zhu, S.-C. (2005). Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Xiao, J. and Quan, L. (2009). Multiple View Semantic Segmentation for Street View Images. In *Proceedings IEEE International Conference Computer Vision*.

Xu, C. and Corso, J. J. (2012). Evaluation of super-voxel methods for early video processing. *Proceedings IEEE Conference Computer Vision and Pattern Recognition*.

Zhang, C., Wang, L., and Yang, R. (2010). Semantic segmentation of urban scenes using dense depth maps. In *Proceedings European Conference Computer Vision*.

Zhao, Y. and Zhu, S.-C. (2011). Image parsing via stochastic scene grammar. In *Advances in Neural Information Processing Systems*.

Zhu, S. and Mumford, D. (2006). A stochastic grammar of images. *Foundations and Trends in Computer Graphics and Vision*, 2(4):259–362.