SEEING THE WORLD BEHIND THE IMAGE

Spatial Layout for 3D Scene Understanding

Derek Hoiem

CMU-RI-TR-07-28

Submitted in partial fulfilment of
the requirements for the degree of
Doctor of Philosophy

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

August 2007

Thesis Committee:
Alexei A. Efros, Co-Chair
Martial Hebert, Co-Chair
Takeo Kanade
Rahul Sukthankar, Intel Research Pittsburgh
William T. Freeman, Massachusetts Institute of Technology

ABSTRACT

When humans look at an image, they see not just a pattern of color and texture, but the world behind the image. In the same way, computer vision algorithms must go beyond the pixels and reason about the underlying scene. In this dissertation, we propose methods to recover the basic spatial layout from a single image and begin to investigate its use as a foundation for scene understanding.

Our spatial layout is a description of the 3D scene in terms of surfaces, occlusions, camera viewpoint, and objects. We propose a *geometric class* representation, a coarse categorization of surfaces according to their 3D orientations, and learn appearance-based models of geometry to identify surfaces in an image. These surface estimates serve as a basis for recovering the boundaries and occlusion relationships of prominent objects. We further show that simple reasoning about camera viewpoint and object size in the image allows accurate inference of the viewpoint and greatly improves object detection. Finally, we demonstrate the potential usefulness of our methods in applications to 3D reconstruction, scene synthesis, and robot navigation.

Scene understanding from a single image requires strong assumptions about the world. We show that the necessary assumptions can be modeled statistically and learned from training data. Our work demonstrates the importance of robustness through a wide variety of image cues, multiple segmentations, and a general strategy of soft decisions and gradual inference of image structure. Above all, our work manifests the tremendous amount of 3D information that can be gleaned from a single image. Our hope is that this dissertation will inspire others to further explore how computer vision can go beyond pattern recognition and produce an understanding of the environment.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

Although thesis advisors always play a large role in the inspiration, development, and conclusion of the thesis, I believe that my own primary advisors, Alyosha Efros and Martial Hebert, deserve special mention. They have provided a perfect blend of big thinking and practical guidance. I am thankful for having the chance to work closely with them and proud of our accomplishments over the past three years.

I also thank Henry Schneiderman, my pre-thesis advisor, for encouraging me to develop a solid understanding of the fundamentals and background material that has served me well. I thank Rahul Sukthankar for his advisement during Intel internships and as a thesis committee member. I have enjoyed working with Rahul on several projects, including my first papers, and he has provided sound advice in many important decisions. I am also grateful for the involvement of Takeo Kanade and Bill Freeman who have each provided key insights into my work and offered suggestions that led to improvements.

I also owe much to innumerable friends and students at CMU and elsewhere, for whom I have great respect and admiration. I have enjoyed collaborating with Yan Ke, Jean-François Lalonde, Andrew Stein, Bart Nabbe, Larry Huston, Carsten Rother, John Winn, Pushmeet Kohli, and Martin Szummer. I thank the CMU Shotokan Karate group for years of practice, especially Bruce, Mark, and John who have given so much of their time for instruction.

Finally, I thank my parents, Brian and Dietlind, my sister, Kirsten, and my wife, Liz, for their many years of love and encouragement.

CHAPTER 1

Introduction

The eye sees only what the mind is prepared to comprehend.
Henri Bergson (1859 - 1941)

In recent years, computer vision researchers have made tremendous progress in many of the individual components of vision, such as face recognition, structure from motion, matching, and tracking. However, when it comes to a general understanding of the visual scene, or "seeing" as humans think of seeing, capabilities are virtually non-existent. For instance, in Figure 1.1, a computer may be able to detect the road surface and some of the cars, but it will have no idea that the cars are actually driving on the road. The computer would therefore also be quite content to decide that a car is sitting just outside of the second story window or nestled among the treetops. As a consequence, computer vision algorithms are completely inept at answering simple questions such as "Am I about to be hit by a bicycle?" or "How do I get inside that red building?"

Our ultimate goal is to develop computer vision algorithms that can answer these questions and more – to allow computers to truly *understand the scene*. The key, we believe, is to formulate the vision problem in terms of the underlying 3D scene, applying real-world knowledge to gain a spatial understanding of the scene layout and its contents. Our primary contribution in this dissertation is to provide a coherent interpretation of the 3D surfaces, occlusion relationships, and camera viewpoint of the scene, all from a *single* image. This spatial understanding can then be used as a foundation for other visual tasks, enabling a wide variety of applications such as image search, 3D scene reconstruction and synthesis, and mobile robot navigation and environment awareness.

Figure 1.1: Computer vision algorithms may be able to detect a few cars and pedestrians in this image but have no understanding of the scene as a whole. As a result, computers cannot answer even simple questions, such as "Am I about to be hit by a bicycle?"

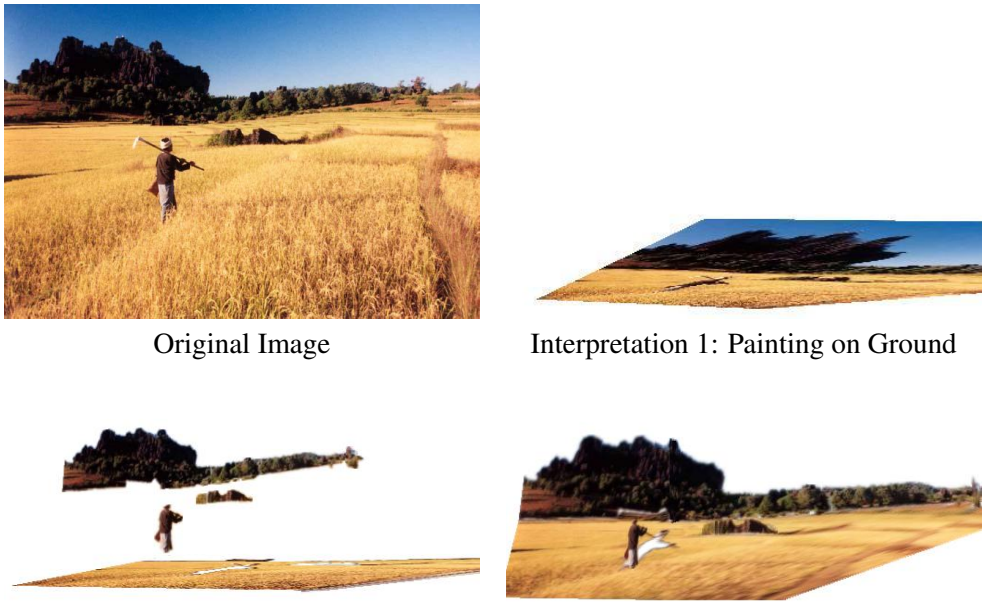
The key contributions of this dissertation are as follows:[1]

- Estimation of spatial layout
  - A coarse description of the major scene surfaces
  - Boundaries and occlusion relationships of prominent objects
  - Camera viewpoint, sufficient to measure true object heights and relative depth
- Creation of a Geometric Context Dataset suitable to evaluate our methods
- Application to object detection, automatic single-view reconstruction, and long-range robot path planning

1. Challenges

One big challenge is dealing with the inherent ambiguity of our task. Mathematically, there is no way to recover the 3D spatial layout from a single 2D image because the image could be generated by an infinite number of geometrical configurations (see Figure 1.2). One image, by itself, simply does not contain enough information to recover 3D spatial layout. And yet, a person that views a photograph sees, not just a plane full of color and texture, but the world behind the image. How can we give computers this same ability? The key is that while an infinitude of interpretations are possible, very few are plausible.

[1] These contributions were originally reported in [**47, 48, 49, 94, 50, 51**].



Original Image | Interpretation 1: Painting on Ground

Interpretation 2: Floating Objects | Interpretation 3: Man in Field

Figure 1.2: Original image and novel views under three different 3D interpretations. Each of these projects back into the input image from the original viewpoint. Although any image could be produced by any of an infinite number of 3D geometrical configurations, very few are plausible. Our main challenge is to learn the structure of the world to determine the most likely solution.

Thus, to succeed, we must take a statistical approach, learning the regularities of the natural world and finding the most likely 3D interpretation of a given image. This still leaves several questions unanswered: How many examples are needed to learn these regularities? Which monocular cues are useful? How can we combine these cues to give a reliable scene interpretation? We will show that a surprisingly small number of examples are required to get a good 3D sense of a wide variety of scenes. Many cues have been proposed by vision researchers, such as linear perspective, texture gradients, shape, height in the image, occlusion, and relative size. We will explore all of these cues and also show that attributes that may seem irrelevant to 3D geometry, such as color, can aid 3D interpretation.

A second major challenge is that of representation. How can we represent the 3D space of the scene in a way that facilitates tasks such as navigation and recognition? A wide variety of 3D representations have been proposed as components of the human visual system, including metric depth maps, surface orientations, and ordinal relationships. We propose a very coarse representation of 3D surfaces which, together with ordinal relationships from

recovered occlusion boundaries and the camera viewpoint, can provide an estimate of depth, up to a scale. Although our 3D estimates are coarse, we will show that they are sufficient to reconstruct convincing models of the scene and are useful for navigation.

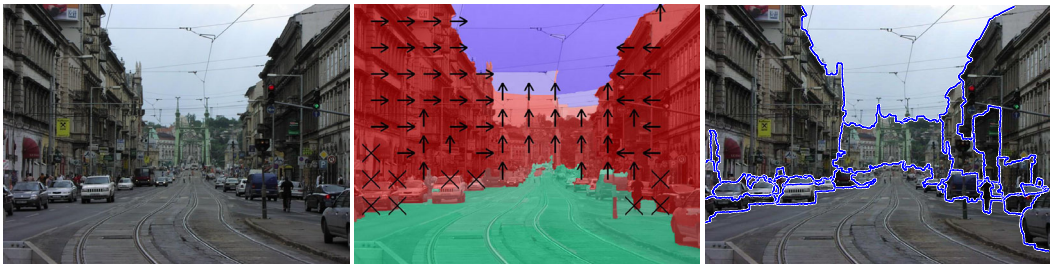
A third challenge is how to use 3D scene information to recognize objects within the scene. Although it seems obvious that knowledge about the spatial layout aids recognition, the extent and manner in which it is helpful is widely under debate in the human vision community. In the recent computer vision literature, the use of context is typically limited to 2D positional relationships in the image and the likelihood of appearance of an object within a scene. We will show how simple methods of reasoning about the 3D size and support of objects can improve recognition.

2. Spatial Layout

Our aim is to recover the rough *spatial layout* of a scene, a representation of the major surfaces and their relationships to each other. Having such a representation would then allow each object to be physically "placed" within the frame and permit reasoning between the different objects and their 3D environment. Our representation consists of three key elements: surface orientations, major occlusion relationships, and camera viewpoint. An example result is shown in Figure 1.3.

Surface Layout. We propose a technique to estimate surface layout, coarsely describing the orientations of major scene surfaces. Rather than attempting to recover exact 3D orientations at every point in the scene, our goal is to label the image into coarse geometric classes. Each image pixel is classified as either being parallel to the ground plane, belonging to a surface that sticks up from the ground, or being part of the sky. Surfaces sticking up from the ground are then further divided into planar surfaces facing left, right or toward the camera and non-planar surfaces, either porous (e.g., leafy vegetation) or solid (e.g., a tree trunk).

Occlusion Relationships. To provide a more complete notion of spatial layout, we propose a method to recover the major occlusion relationships. Our goal is to delineate the occlusion boundaries of major free-standing physical structures and determine the ordinal depths. This is very much related to the aims of image segmentation – to partition the visual field into meaningful, coherent regions. The major difference is how we define what makes



Input Image Surfaces Boundaries

Objects/Viewpoint Novel View

Figure 1.3: Our goal is to recover 3D surface, occlusion, camera viewpoint, and object information from a single input image. Here, we show a result for the input image on the upper-left. For the surface estimates, main geometric classes are shown by color (green="support", red="vertical", blue="sky"), and the subclasses are shown by markings (←="planar left", ↑="planar center", →="planar right", O="non-planar porous", X="non-planar solid"). Occlusion boundaries are indicated with the blue/white lines. Detected cars and pedestrians objects are indicated by soft object masks at detected positions, with one color per object and brightness indicating confidence. The estimated horizon position is shown with a blue line. Most of the pedestrians are partially occluded. Finally, we show a novel view of the automatically reconstructed 3D scene, taken after moving into the scene, to the left, and down.

a coherent region. Most segmentation algorithms rely on 2D perceptual grouping cues, such as brightness, color or texture similarity for region-based methods or edge strength, continuity, and closure for contour-based methods. The boundaries of such segmentations often correspond to reflectance, illumination, or material discontinuities, rather than true object boundaries. In this work, we aim to produce a segmentation due solely to geometric boundaries, with each region containing its own physical 3D object. For each boundary, we then attempt to determine which side occludes the other.

Camera Viewpoint. To project observed objects and events from the image coordinates into the 3D scene, we additionally need to know the viewpoint of the camera. In particular, we show that recovering just two parameters, the horizon position and the camera height, is often sufficient to estimate the 3D heights of objects. We show that by leveraging a large

training set and by incorporating object and surface information into our inference, we are able to recover these parameters accurately for a wide variety of images while providing large improvement in object detection.

3. Our Approach

If we are to recover 3D scene geometry from a single image, we need to take full advantage of the natural regularities of our world and the common ways that people view it. We take a soft, statistical approach: *learn* the structure of the visual world from a set of training images and incorporate as much information as possible to gradually infer the *most likely* 3D scene. Our approach, which we apply to typical photographs of general scenes, stands in stark contrast to past work that recovers geometric structure from line drawings of simple scenes (e.g., [**6, 64, 38**]) or multiple images or video sequences (e.g., [**96, 109**]), or that estimates viewpoint under restrictive assumptions [**69, 17**]. That we are able to get any 3D information from one image is surprising. That we can often go as far as to reconstruct convincing 3D models of the scene is a testament to strong regularities of the visual world and the power of a data-driven approach.

One key idea in our work is to use all of the available cues. For instance, in estimating surface orientations, we compute several dozen cues to represent color, texture, position, shape, and perspective. To estimate occlusion boundaries, we incorporate edge strength, continuity, and length, region color and texture, and estimates of surface orientation and relative depth. While the long lists of cues used in our algorithms may give the impression of a highly engineered system, the opposite is in fact true. Rather than spending weeks figuring out which texture filters or color space to use or whether edge information helps, we represent all information that we think *might* be helpful and employ our machine learning algorithms (usually boosted decision trees [**16, 30**]) to determine which cues to use and how to use them. This approach, made possible by advances in machine learning over the last ten years, helps to provide robust solutions that require little or no tweaking, as demonstrated in our experiments.

Many of the relevant cues, such as texture histograms, perspective, or relative depth, require good spatial support (i.e., a segmentation) to be useful. A second key aspect in our approach is to gradually infer the required spatial support. To estimate surface orientations,

we slowly build from pixels to superpixels to multiple segmentations. We use the spatial support provided by the segmentations to better evaluate whether each segment is good and to determine the label of the segment, giving a surface estimate for each segmentation. These are then combined into a final estimate for the image. In estimating occlusion boundaries, we first group pixels into an oversegmentation containing thousands of regions. Then, incorporating additional information as it becomes useful, we reduce the number of regions from thousands to a few hundred to a few dozen to the final segmentation. Our experiments show that the provided spatial support improves accuracy considerably.

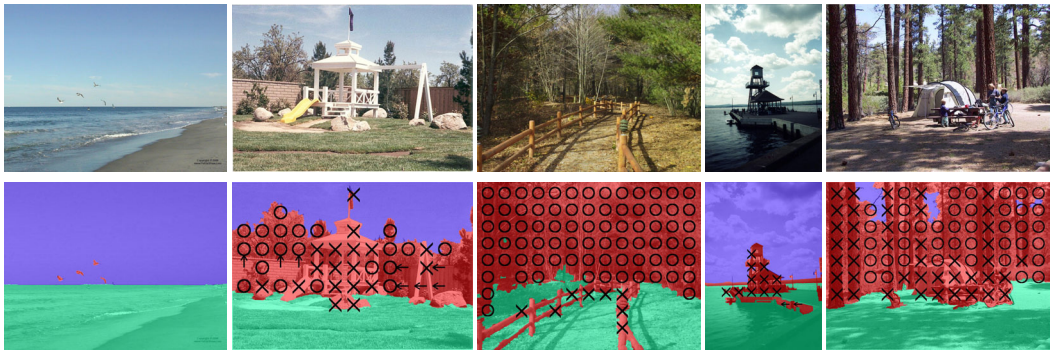
A third important idea is to obtain a coherent interpretation of the image. For instance, we simultaneously recover the camera viewpoint and objects that are consistent with each other and the estimated surface geometry. Likewise, when estimating occlusion boundaries, we enforce consistency among our surface orientations, figure/ground labels, and relative depth estimates. Thus, we are able to obtain a geometrically coherent understanding of the scene that is both more intuitive and more accurate than solutions we could obtain by applying various computer vision algorithms independently.

4. Datasets

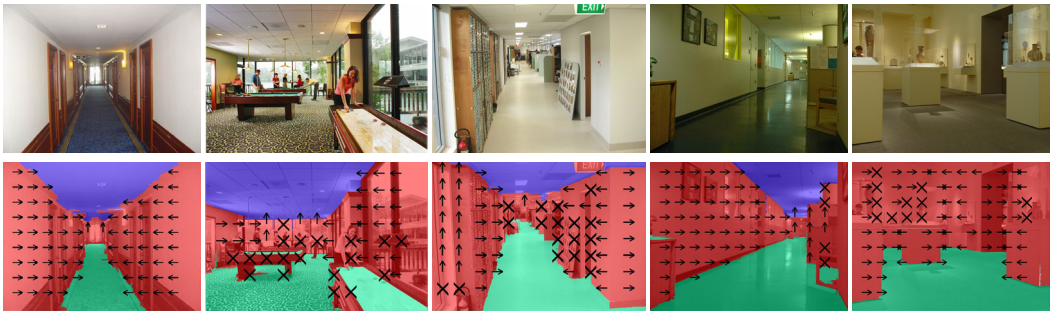
Since no previously existing datasets are appropriate to train or evaluate our algorithms, we create a new Geometric Context dataset of outdoor images. We also provide additional annotation for the Stanford indoor image dataset [**23**] and the LabelMe dataset [**116**]. We show examples of typical images from our datasets in Figure 1.4.

4.1. Geometric Context

Our Geometric Context dataset is intended to be suitable for evaluating surface layout and 3D reconstruction. We restrict this dataset to outdoor images taken from common viewpoints (the horizon is within the image and aerial photos are removed). In all, we gathered 300 outdoor images using Google image search, with keywords such as “alley”, “beach”, “buildings”, “city”, “cliff”, and “college”. The resulting images offer a wide variety of environments (forests, cities, roads, beaches, lakes, etc.) and conditions (snowy, sunny, cloudy, twilight). The image sizes range from about 300x240 to 1024x768, with varying aspects. The complexity of the scenes ranges from highly cluttered (e.g., a crowd of people) to an

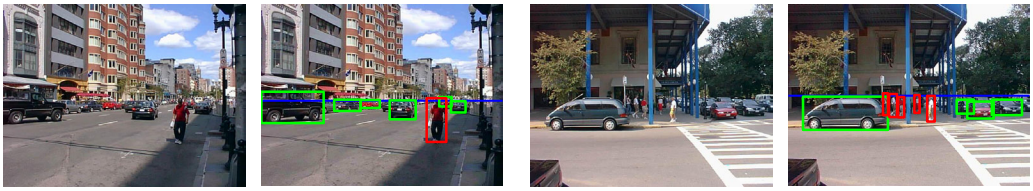


Geometric Context Dataset with Geometric Classes



Stanford Indoor Dataset with Geometric Classes

Geometric Context Dataset with Occlusion Boundaries



LabelMe Dataset with Horizon and Objects

Figure 1.4: We show examples of images and ground truth annotations for the Geometric Context, Stanford indoor, and Labelme datasets. Main geometric classes are shown by color (green="support", red="vertical", blue="sky"), and the subclasses are shown by markings (←="planar left", ↑="planar center", →="planar right", O="non-planar porous", X="non-planar solid"). Occlusion boundaries are denoted with arrows to indicate figure/ground (side on the left of the arrow is in front). On the LabelMe dataset, horizon is shown with blue line, and ground truth cars and pedestrians with green and red bounding boxes.

open field or ocean. With this dataset, we provide ground truth for the geometric classes on all 300 images and for the occlusion boundaries on 100 images.

To facilitate ground truth labeling of the geometric classes, we first oversegment the images into an average of about 500 regions (or "superpixels") per image. We then assign each superpixel a geometric label (main class $\in$\{support,vertical,sky\}, vertical subclass $\in$\{left,center,right,porous,solid\}) by a combination of drawing and labeling polygons and clicking on individual superpixels. In all, roughly 150,000 superpixels were labeled in a time-consuming process. Though rare, a superpixel will sometimes include more than one geometric class (e.g., when a shadow completely darkens the bottom of a vertical surface and the ground). In those cases, we assign the most appropriate label. In our experiments, we use 50 images to train a segmentation and the remaining 250 images to train and test the geometric class appearance models in five-fold cross-validation.

When labeling occlusion boundaries, we need to respect the boundaries of individual objects. For example, an entire region consisting of two people standing close together would be given a geometric label of "solid", but each person would have a separate boundary. We segment each image into thousands of regions (see Chapter 4) and manually group the regions into objects, which could be discontinuous due to occlusion. We then label the occlusion relationships of adjacent objects. A medium-complexity image will typically contain 10–15 objects, according to our ground truth segmentations. Because precise ground truth information is required to train an occlusion boundary classifier, providing these labels is extremely time consuming. For this reason, we label a subset of the dataset, providing 50 images for training and 50 images for quantitative evaluation. The remaining 200 images are used for further qualitative evaluation.

4.2. Stanford Indoor

To evaluate our geometric class labeling on indoor images, we annotated ground truth geometry labels for the Stanford dataset of 92 indoor images originally used to test the indoor 3D reconstruction method of Delage, Lee, and Ng [**23**]. These images contain a mixture of 48 images taken with a calibrated camera and 44 indoor images collected from the Internet. We provided ground truth geometric class labels for these images using the same method

as for our Geometric Context dataset. For simplicity, we used the same geometric classes as for outdoors, except that the "sky" class is redefined as the ceiling.

4.3. LabelMe

To evaluate camera viewpoint estimation and object detection, we provide additional annotation for the LabelMe dataset [**116**]. Our test set consists of 422 outdoor images from LabelMe. The busy city streets, sidewalks, parking lots, and roads provide realistic environments for evaluating car and pedestrian detectors, and the wide variety of object pose and size and the frequency of occlusions make detection extremely challenging. In the dataset, 60 images have no cars or pedestrians, 44 have only pedestrians, 94 have only cars, and 224 have both cars and pedestrians. In total, the images contain 923 cars and 720 pedestrians. We annotated bounding boxes for cars with heights as small as 14 pixels and pedestrians as small as 36 pixels tall. Bounding boxes are set to include the entire extent of the object, even if the object is partially occluded. If an object is more than 50% occluded, we did not annotate it. It is very difficult to provide consistent ground truth for object detection. For this reason, some objects may be missed in the annotation, or a partially occluded object may be detected that is not annotated. This tends to give a slightly pessimistic estimate of the object detection performance in absolute terms, but it does not obscure the substantial improvement that we obtain using our algorithm. We also created a validation set of 60 images that can be used for training.

To evaluate our camera viewpoint estimation, we manually labeled the horizon position in 100 of the 422 test images. To improve our horizon estimation, we also automatically recovered the camera viewpoint and 3D object sizes for a large portion of the LabelMe dataset. Our method assumes that objects rest on the ground plane and infers the size distributions of manually labeled objects in the dataset and the camera viewpoint in the images that contain those objects (see Chapter 5 for details). After initially providing only a guess of the mean and standard deviation height of people, we inferred the camera viewpoint of over 5,000 images and heights of 13,000 object instances in roughly fifty object classes.

CHAPTER 2

Background

The beginning of knowledge is the discovery of something we do not understand.
Frank Herbert (1920 - 1986)

The mechanism by which humans can perceive depth from light is a mystery that has inspired centuries of vision research, creating a vast wealth of ideas and techniques that make our own work possible. In this chapter, we provide a brief historical context for 3D scene understanding, ranging from early philosophical speculations to more recent work in computer vision. Later chapters provide additional discussion of work concerning particular aspects of our spatial layout algorithms.

1. Theories of Vision

The elementary impressions of a visual world are those of surface and edge.
James Gibson, *Perception of a Visual World* (1950)

For centuries, scholars have pondered the mental metamorphosis from the visual field (2D retinal image) to the visual world (our perception of 3D environment). In 1838, Charles Wheatstone [**150**] offered the explanation of stereopsis, that "the mind perceives an object of three dimensions by means of the two dissimilar pictures projected by it on the two retina." Wheatstone convincingly demonstrated the idea with the stereoscope but noted that the idea is applies only to nearby objects.

Hermann von Helmholtz, the most notable of the 19th century empiricists, believed in an "unconscious inference", that our perception of the scene is based, not only on the immediate sensory evidence, but on our long history of visual experiences and interactions with the world [**148**]. This inference is based on an accumulation of evidence from a variety of cues, such as the horizon, shadows, atmospheric effects, and familiar objects.

Gibson laid out [**31**] a theory of visual space perception which includes the following tenets: 1) the fundamental condition for seeing the visible world is an array of physical surfaces, 2) these surfaces are of two extreme types: frontal and longitudinal, and 3) perception of depth and distance is reducible to the problem of the perception of longitudinal surfaces. Gibson further theorized that gradients are the mechanism by which we perceive surfaces.

By the 1970s, several researchers had become interested in computational models for human vision. Barrow and Tenenbaum proposed the notion of *intrinsic images* [**5**], capturing characteristics, such as reflectance, illumination, surface orientation, and distance, that humans are able to recover from an image under a wide range of viewing conditions. Meanwhile, David Marr proposed a three-stage theory of human visual processing [**87**]: from primal sketch (encoding edges and regions boundaries), to $2\frac{1}{2}$D sketch (encoding local surface orientations and discontinuities), to the full 3D model representation.

More recently, Michael Hucka presented his dissertation work [**56**] that attempts to recover an approximate spatial layout, described as a general sense of the locations and shapes of surfaces in the scene, from an image. Hucka motivated his work from a psychophysical perspective and proposed methods for texture-based segmentation and shape recovery using biologically-motivated cues to interpret artificial scenes and simple natural scenes.

1.1. Depth and Surface Perception

The extent and manner in which humans are able to estimate depth and surface orientations has long been studied. Though the debate continues on nearly every aspect of recovery and use of spatial layout, some understanding is emerging. Three ideas in particular are highly relevant to this dissertation: 1) that monocular cues provide much of our depth and layout sensing ability; 2) that an important part of our layout representation and reasoning primarily is based on surfaces, rather than metric depth; and 3) that many scene understanding processes depend on the viewpoint.

Cutting and Vishton [**20**], using a compilation of experimental data where available and logical analysis elsewhere, rank visual cues according to their effectiveness in determining ordinal depth relationships (which object is closer). They study a variety of monocular, binocular, and motion cues. They define three spatial ranges: personal space (within a one or two meters), action space (within thirty meters), and vista space (beyond thirty meters) and find that the effectiveness of a particular cue depends largely on the spatial range. Within the personal space, for instance, the five most important cues primarily correspond to motion and stereopsis: interposition,[1] stereopsis, parallax, relative size, and accommodation. In the vista range, however, all of the top five cues are monocular: interposition, relative size, height in visual field, relative density, and atmospheric perspective. Likewise, in the action range, three of the top five most important cues are monocular. Cutting and Vishton suggest that stereo and motion cues have been overemphasized due to their importance in the personal space and that much more study is needed in the monocular cues that are so critical for spatial understanding of the broader environment.

Koenderink and colleagues [**67, 66**] experimentally measure the human's ability to infer depth and local orientation from an image. Their subjects were not able to accurately (or even consistently) estimate depths of a 3D form (e.g., a statue), but could indicate local surface orientations. They further provide evidence that people cannot determine the relative depth of two points unless there is some visible and monotonic surface that connects them. Zimmerman et al. [**156**] provide additional evidence that people can make accurate slant judgements but are unable to accurately estimate depth of disconnected surfaces. These experimental results confirm the intuitions of Gibson and others – that humans perceive the 3D scene, not in terms of absolute depth maps, but in terms of surfaces.

There has been an enormous amount of research into the extent to which scene representations and analysis are dependent on viewpoint. If human visual processing is viewpoint-independent, the implication is that we store 3D models of scenes and objects in our memory and manipulate them to match the current scene. Viewpoint-independence, however, implies a 2D centric representation that requires viewpoint for registration. While it is likely that both viewpoint-dependent and viewpoint-independent storage and processing exist, the evidence indicates that a view-dependent representation is more dominant. For instance,

[1] Note that interposition provides near-perfect ordinal depth information, since the occluding object is always in front, but is uninformative about metric depth.

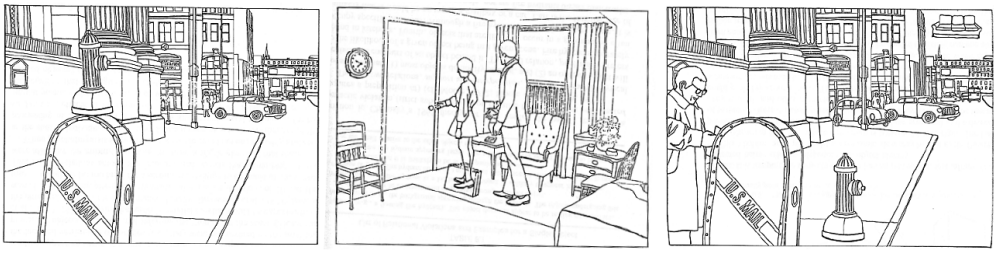
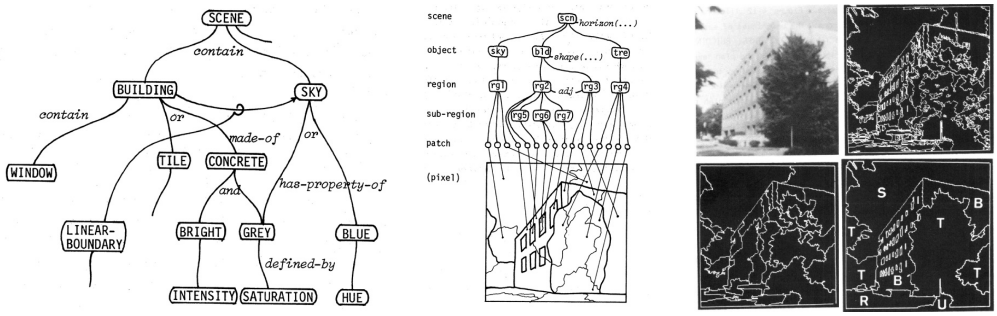


Figure 2.1: Examples violations of Biederman's five relational constraints. On the left is a violation of position; in the center, a violation of interposition; and on the right, there are violations of support, size, and probability of appearance. Illustration from [**7**], used with permission.

Chua and Chun [**13**] find that implicit scene encodings are viewpoint dependent. Subjects were able to find targets faster when presented with previously viewed synthetic 3D scenes, but the gain in speed deteriorates as the viewpoint in a repeated viewing differs from the original. Gottesman [**34**] provides further evidence with experiments involving both photographs and computer-generated images. The subjects are first primed with an image of the scene and then presented with the target, which is the same scene rotated between 0 and 90 degrees. When asked to judge the distance between two objects in the target scene, response times increased with larger differences in viewpoint between the prime and the target.

1.2. A Well-Organized Scene

Irving Biederman [**7**] characterizes a well-organized scene as one that satisfies five relational constraints: support, interposition, probability of appearance, position, and size. See Figure 2.1 for illustrations. In order, these properties are described as follows: objects tend to be supported by a solid surface; the occluding object tends to obscure the occluded object; some objects are more likely to appear in certain scenes than others; objects often belong in a particular place in the scene; and objects usually have a small range of possible sizes. Though he does not explain how these relationships are perceived, Biederman supplies evidence that they are extremely valuable for scene understanding. For instance, when a subject is asked to determine whether a particular object is present in a scene, the response time increases when the object violates one of the relational constraints listed above. Hollingworth and Henderson [**53**] have questioned these results (particularly for probability of appearance), showing that experimental biases could account for some of the



(a) Bottom-up process (b) Top-down process (c) Result

Figure 2.2: A system developed in 1978 by Ohta, Kanade and Sakai [**100, 99**] for knowledge-based interpretation of outdoor natural scenes. The system is able to label an image (c) into semantic classes: S-sky, T-tree, R-road, B-building, U-unknown. Figure used with permission.

reported differences. Instead, they suggest that people may be more likely to remember unusual or unexpected objects, possibly due to memory or attentional processes responding to surprise. This consistency effect, that humans tend to remember objects that are somehow inconsistent with their environment, has long been studied (e.g., [**46, 107**]) and is further evidence that such relational constraints play an important role in recognition and scene understanding.

2. Early Computer Vision and AI

> *Given a single picture which is a projection of a three-dimensional scene onto the two-dimensional picture plane, we usually have definite ideas about the 3-D shapes of objects. To do this we need to use assumptions about the world and the image formation process, since there exist a large number of shapes which can produce the same picture.*
>
> Takeo Kanade, "Recovery of the Three-Dimensional Shape of an Object from a Single View" (1981)

In its early days, computer vision had but a single grand goal: to provide a complete semantic interpretation of an input image by reasoning about the 3D scene that generated it. Some of this effort focused on achieving 3D reconstructions of objects or simple scenes, usually from line drawings. For example, Kanade [**64**] demonstrates how assumptions about

geometric properties such as planarity, parallelism and skewed symmetry can be used to reconstruct the 3D shape of a chair. Barrow and Tenenbaum [**6**] suggest that contour information should be combined with other cues, such as texture gradient, stereopsis, and shading and speculate on the primacy of geometric cues in early vision.

Initial attempts at a broader scene understanding focused largely on toy "blocks worlds" [**114, 37**], but, by the 1970s, several extremely sophisticated approaches were proposed for handling real indoor and outdoor images. For instance, Yakimovsky and Feldman [**152**] developed a Bayesian framework for analyzing road scenes that combined segmentation with semantic domain information at the region and inter-region level. Tenenbaum and Barrow proposed *Interpretation-Guided Segmentation* [**137**] which labeled image regions, using constraint propagation to arrive at a globally consistent scene interpretation. Ohta, Kanade and Sakai [**100, 99**] combined bottom-up processing with top-down control for semantic segmentation of general outdoor images. Starting with an oversegmentation, the system generated "plan images" by merging low-level segments. Domain knowledge was represented as a semantic network in the bottom-up process (Figure 2.2a) and as a set of production rules in the top-down process (Figure 2.2b). Results of applying this semantic interpretation to an outdoor image are shown on Figure 2.2c. By the late 1970s, several complete image understanding systems were being developed including such pioneering work as Brooks' *ACRONYM* [**10**] and Hanson and Riseman's *VISIONS* [**40**]. For example, *VISIONS* was an ambitious system that analyzed a scene on many interrelated levels including segments, 3D surfaces and volumes, objects, and scene categories.

It is interesting to note that a lot of what are considered modern ideas in computer vision – region and boundary descriptors, superpixels, combining bottom-up and top-down processing, Bayesian formulation, feature selection, etc. – were well-known three decades ago. But, though much was learned in the development of these early systems, it was eventually seen that the hand-tuned algorithms could not generalize well to new scenes. This, in turn, lead people to doubt the very goal of complete image understanding. However, it seems that the early pioneers were simply ahead of their time. They had no choice but to rely on heuristics because they lacked the large amounts of data and the computational resources to *learn* the relationships governing the structure of our visual world.

3. Modern Computer Vision

The failures of early researchers to provide robust solutions to many real-world tasks led to a new paradigm in computer vision: rather than treat the image as a projection from 3D, why not simply analyze it as a 2D pattern? Statistical methods and pattern recognition became increasingly popular, leading to breakthroughs in face recognition, object detection, image processing, and other areas. Success came from leveraging modern machine learning tools, large data sets, and increasingly powerful computers to develop data-driven, statistical algorithms for image analysis.

It has become increasingly clear, however, that a purely 2D approach to vision cannot adequately address the larger problem of scene understanding because it fails to exploit the relationships that exist in the 3D scene. Several researchers have responded, and much progress in recent years has been made in spatial perception and representation and more global methods for reasoning about the scene.

Song-Chun Zhu and colleagues have contributed much research in computational algorithms for spatial perception and model-driven segmentation. Guo, Zhu, and Wu [**36**] propose an implementation of Marr's primal sketch, and Han and Zhu [**39**] describe a grammar for parsing image primitives. Tu and Zhu [**144**] describe a segmentation method based on a generative image representation that could be used to sample multiple segmentations of an image. Barbu and Zhu [**4**] propose a model-driven segmentation with potential applications to image parsing [**143**].

Nitzberg and Mumford [**97**] describe a 2.1D sketch representation, a segmentation and layering of the scene, and propose an algorithm for recovering the 2.1-D sketch from an image. The algorithms consists of three phases: finding edges and T-junctions, hypothesizing edge continuations, and combinatorially finding the interpretation that minimizes a specified energy function. They show that the algorithm can account for human interpretations of some optical illusions. Huang, Lee, and Mumford [**55**] investigate the statistics of range images, which could help provide a prior on the 3D shape of the scene.

Oliva and Torralba [**101, 102**] characterize the "spatial envelope" of scenes with a set of continuous attributes: naturalness, openness, roughness, expansion, and ruggedness. They further provide an algorithm for estimating these properties from the spectral signature of

an image and, in [**141**], used similar algorithms to estimate mean depth of the scene. These concepts have since been applied to object recognition [**92, 132**] and recovery of depth information based on objects in an image [**133**].

Some recent work has also attempted recovery of 3D information from a single image. Han and Zhu [**38**] reconstruct polyhedral or wire-like objects in simple scenes. Delage, Lee, and Ng [**23**] present a method for reconstructing indoor scenes from a single image, and Saxena, Chung, and Ng [**118**] present a method to learn depth from single outdoor images based on low-level features in an MRF model. More recently, Saxena et al. [**120**] have extended their model to reason over contours, producing more accurate 3D reconstructions.

4. Relation to Our Work

Our own work has been heavily influenced by all of these ideas. Our approach shares Helmholtz' intuition and empiricist philosophy by learning models of surfaces and occlusions from the "experience" of a training set and by drawing from a large and diverse set of visual cues. Our geometric classes correspond strongly to Gibson's notions of basic surface type, though we differ from his belief in the primacy of gradient-based methods. Our spatial layout is also philosophically similar to Marr's $2\frac{1}{2}$D sketch. However, we differ from it in several important ways: 1) we use statistical learning instead of relying solely on a geometric or photometric methodology (e.g., Shape-from-X methods), 2) we are interested in a rough sense of the scene surfaces, not exact orientations, and 3) our spatial layout is to be used *with* the original image data, not as a substitute for it. Our representation of spatial layout as a series of confidence maps for orientations, depth, boundaries, and objects is highly similar to the Barrow and Tenenbaum's idea of intrinsic images. Our work shares similar goals with Hucka's dissertation work but differs in our use of a wide variety of cues (not just texture) and in its application to a diverse set of natural scenes.

Cutting and Vishton claimed that monocular spatial cues have too often been ignored. Our work investigates these cues and is the first to provide a computational process for estimating 3D scene surfaces and occlusion relationships from images of general scenes. Our own viewpoint-centric approach corresponds with findings that humans often condition visual sensing on the viewpoint. We work with images taken from a limited range of viewpoints

(e.g., no overhead views) and find that precise viewpoint estimates improve object detection, surface estimation, and depth estimation.

Our work in object detection can be seen as an operationalization of Biederman's proposed scene characteristics of support, interposition, and size. In particular, we find that exploiting the support and size relationships of objects under the simple assumption that objects rest on a ground plane gives a large improvement in detection accuracy. We present a method for recovering occlusion relationships from an image and begin to investigate how interposition can aid object recognition.

We draw inspiration from the global scene representation of Oliva and Torralba and the model-driven stochastic segmentation processes of Barbu, Tu, and Zhu. The 2.1-D sketch of Nitzberg and Mumford is similar to our representation of an image in terms of the object boundaries and the depth ordering. Our algorithm, however, does not rely on a preprocess of finding edges and junctions but instead forms the boundaries and estimates the depth layering at the same time. The depth estimation work of Saxena et al. could be used to complement our spatial layout by providing additional evidence for occlusion boundaries.

CHAPTER 3

Recovering a Coarse Surface Layout

For I dipped into the future, as far as human eye could see, saw a vision of the world, and all the wonder that would be.
Alfred, Lord Tennyson (1809-1892), *Ulysses*

In this chapter, our goal is to recover the *surface layout*, a coarse representation of the orientations of major surfaces in the scene. The surface layout allows limited reasoning between objects and their 3D environment, such as determining relative depth in uncluttered scenes or physical support relationships.

We pose the problem of 3D surface estimation in terms of statistical learning. Rather than trying to explicitly compute all of the required geometric parameters from the image, we rely on other images (a training set) to furnish this information in an implicit way, through recognition. But in contrast to most recognition approaches that model semantic classes, such as cars, vegetation, roads, or buildings [**99, 27, 68, 126**], our goal is to model *geometric classes* that depend on the orientation of a physical object in the scene. For instance, a piece of plywood lying on the ground and the same piece of plywood propped up by a board belong to same semantic class but different geometric classes. Unlike other reconstruction techniques that require multiple images (e.g., [**110**]), manual labeling [**19, 81**], or very specific scenes [**38**], we want to automatically estimate the 3D surface properties of general outdoor scenes from a single image.

Our main insight is that 3D geometric information can be obtained from a single image by learning appearance-based models of surfaces at various orientations. We present a framework that progressively builds structural knowledge of the scene by alternately using

Figure 3.1: Surface layout. In these images and elsewhere, main class labels are indicated by colors (green=support, red=vertical, blue=sky) and subclass labels are indicated by markings (left/up/right arrows for planar left/center/right, 'O' for porous, 'X' for solid).

estimated scene structure to compute more complex image cues and using these more complex image cues to gain more structural knowledge. We demonstrate the effectiveness of our approach and provide a thorough analysis of the impact of various design choices of our algorithm.

1. Geometric Classes

We pose surface layout recovery as a recognition problem. Our goal is to label an image of an outdoor scene into coarse geometric classes that will be useful for tasks such as navigation, object recognition, and general scene understanding. The feasibility of our goal arises from two tendencies that we observed in 300 outdoor images collected using Google image search. The first is that nearly all pixels (over 97%) belong to horizontal (support) surfaces, nearly vertical surfaces, or the sky. The second is that, in most images, the camera axis is roughly aligned with the ground plane, allowing us to reconcile world-centric cues (such as material) with view-centric cues (perspective). Figure 3.1 shows images labeled with these geometric classes.

1.1. Main Classes

Every region in the image is categorized into one of three main classes: "support", "vertical", and "sky". Support surfaces are roughly parallel to the ground and could potentially support a solid object. Examples include road surfaces, lawns, dirt paths, lakes, and table tops. Vertical surfaces, which are defined as too steep to support an object, include walls, cliffs, curb sides, people, trees, or cows. The sky is simply the image region corresponding to the open air and clouds.

1.2. Subclasses

Because the vertical class contains such a wide variety of surfaces, we divide vertical surfaces further into the following subclasses: planar surfaces facing to the "left", "center", or "right" of the viewer, and non-planar surfaces that are either "porous" or "solid". Planar surfaces include building walls, cliff faces, and other vertical surfaces that are roughly planar. Porous surfaces are those which do not have a solid continuous surface. Tree leaves, shrubs, telephone wires, and chain link fences are all examples of porous surfaces. Solid surfaces are non-planar vertical surfaces that do have a solid continuous surface, including automobiles, people, beach balls, and tree trunks.

The blurred boundaries between the definitions of these subclasses often makes (ground truth) labeling somewhat subjective. For instance, how directly must a wall face the viewer for it to be considered "center" instead of "left" or "right"? Is the side of a small car planar? The side of an 18-wheel truck? Is a large branch solid? What about a jumble of twigs? These ambiguities are the inevitable consequence of imposing a compact labeling on the entire visual world. Most of the time, when a region could be conceivably labeled as either of two classes, the assigned label has little practical importance, but the ambiguity does cause trouble when quantitatively evaluating the performance of the automatic system-assigned labels. In providing the ground truth, we attempted to label each region into the most appropriate class, while being as consistent as possible. The reader should note, however, that some quantitative error (perhaps up to 15% for the subclasses) is due to these ambiguities.

2. Cues for Labeling Surfaces

A patch in the image could theoretically be generated by a surface of any orientation in the world. To determine which orientation is most *likely*, we need to use all of the available cues: material, location, texture gradients, shading, vanishing points, etc. In Table 3.1, we list the set of statistics used for classification. Some of these statistics, such as perspective cues, are only helpful when computed over the appropriate spatial support (i.e., a region in a segmentation), which is provided by our multiple segmentation method (Section 3). The sundry and sometimes redundant nature of these statistics reflects our approach: compute all cues that might be useful and allow our classifier (described in Section 4) to decide which to use and how to use them.

SURFACE CUES
Location and Shape
L1. Location: normalized x and y, mean
L2. Location: normalized x and y, 10^{th} and 90^{th} percentile
L3. Location: normalized y wrt estimated horizon, 10^{th}, 90^{th} percentile
L4. Location: whether segment is above, below, or straddles estimated horizon
L5. Shape: number of superpixels in segment
L6. Shape: normalized area in image
Color
C1. RGB values: mean
C2. HSV values: C1 in HSV space
C3. Hue: histogram (5 bins)
C4. Saturation: histogram (3 bins)
Texture
T1. Leung&Malik filters: mean absolute response (15 filters)
T2. Leung&Malik filters: histogram of maximum responses (15 bins)
Perspective
P1. Long Lines: (number of line pixels)/sqrt(area)
P2. Long Lines: percent of nearly parallel pairs of lines
P3. Line Intersections: histogram over 8 orientations, entropy
P4. Line Intersections: percent right of image center
P5. Line Intersections: percent above image center
P6. Line Intersections: percent far from image center at 8 orientations
P7. Line Intersections: percent very far from image center at 8 orientations
P8. Vanishing Points: (num line pixels with vertical VP membership)/sqrt(area)
P9. Vanishing Points: (num line pixels with horizontal VP membership)/sqrt(area)
P10. Vanishing Points: percent of total line pixels with vertical VP membership
P11. Vanishing Points: x-pos of horizontal VP - segment center (0 if none)
P12. Vanishing Points: y-pos of highest/lowest vertical VP wrt segment center
P13. Vanishing Points: segment bounds wrt horizontal VP
P14. Gradient: x, y center of mass of gradient magnitude wrt segment center

Table 3.1: Statistics computed to represent superpixels (C1-C4,T1-T2,L1,L6) and segments (all). The boosted decision tree classifier selects a discriminative subset of these features.

2.1. Location

It should come as no surprise that, for recovering the rough 3D geometry of the scene, the 2D representation itself (position in the image) provides a strong cue. Figure 3.2 displays the likelihood of each geometric class given the x-y position in the image. As one might expect, ground tends to be low in the image, and sky tends to be high. The x-position (or column) in the image tells little about the main classes, but is helpful in distinguishing among some of the subclasses. For instance, planes facing left tend to be on the right side

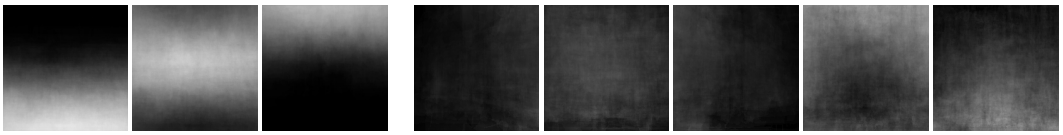


Figure 3.2: Likelihood (higher intensity is more likely) of each geometric class given location in the image. Location is highly discriminative for the main classes. Porous surfaces (often vegetation) tends to form a canopy around the center of the image, while solid surfaces often occur in the front-center of the image. The left/center/right likelihoods show the tendency to take photos directly facing walls or down passages.

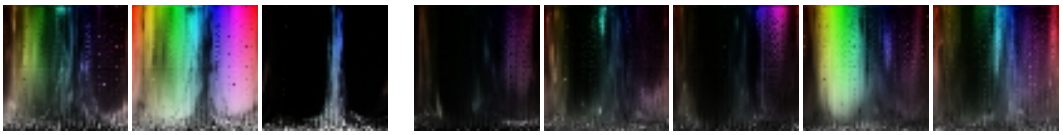


Figure 3.3: Likelihood of each geometric class given hue and saturation. Each image shows the given hue and saturation, with the label likelihood given by the intensity. Blue sky is easily distinguishable. More confusion exists between the support and vertical classes, though natural tones of brown, green, and blue lend evidence for support. Gray (low saturation) is common in all main classes. Color is not very discriminative among the subclasses, with the exception of porous which tends to be colors of vegetation.

of the image, while planes facing right tend to be positioned on the left side. The reason is simply that more photographs are taken facing down a street or path than directly into the corner of a building.

In our representation, we normalize the pixel locations by the width and height of the image. We compute the mean (set L1 in Table 3.1) and 10^{th} and 90^{th} percentile (L2) of the x- and y-position of a segment in the image. We additionally measure the number of superpixels (L5), described in Section 3.1, and pixels (L6), normalized by total image area, in each segment.

2.2. Color

By itself, color has little to do with 3D orientations or geometry. Nevertheless, by modeling color, we can implicitly identify materials and objects that correspond to particular geometric classes, making color a powerful cue. For instance, the sky is usually blue or white,

and support segments are often green (grass) or brown (dirt). In Figure 3.3, we plot the likelihoods for each of the geometric main classes and subclasses given hue or saturation.

We represent color using two color spaces: RGB and HSV (C1-C4). RGB allows the "blueness" or "greenness" of a segment to be easily extracted, while HSV allows perceptual color attributes such as hue and "grayness" to be measured.

2.3. Texture

Similarly to color, texture provides a cue for the geometric class of a segment through its relationship to materials and objects in the world. Texture also relates more directly to the geometric class through properties of surfaces in perspective, such as that a vertical plane will tend to have more vertically oriented textures than a horizontal plane.

To represent texture, we apply a subset of the filter bank designed by Leung and Malik [**78**]. We generated the filters using publicly available code with the following parameters: 19x19 pixel support, $\sqrt{2}$ scale for oriented and blob filters, and 6 orientations. In all, there are 6 edge, 6 bar, 1 Gaussian, and 2 Laplacian of Gaussian filters. Our representation includes the absolute filter response (T1) of each filter and the histogram (over pixels within a segment) of maximum responses (T2).

2.4. Perspective

Knowledge of the vanishing line of a plane completely specifies its 3D orientation relative to the viewer [**41**], but such information cannot easily be extracted from outdoor, relatively unstructured images. Our representation contains perspective information in two basic forms: a "soft" estimate and an explicit estimate of vanishing points. We illustrate our perspective cues in Figure 3.4.

By computing statistics of straight lines (P1-P2) and their intersections (P3-P7) in the image, our system gains information about the vanishing points of a surface without explicitly computing them. Our system finds long, straight edges in the image using the method of Kosecka and Zhang [**69**]. The intersections of nearly parallel lines (within $\pi/8$ radians) are radially binned, according to the direction to the intersection point from the image center

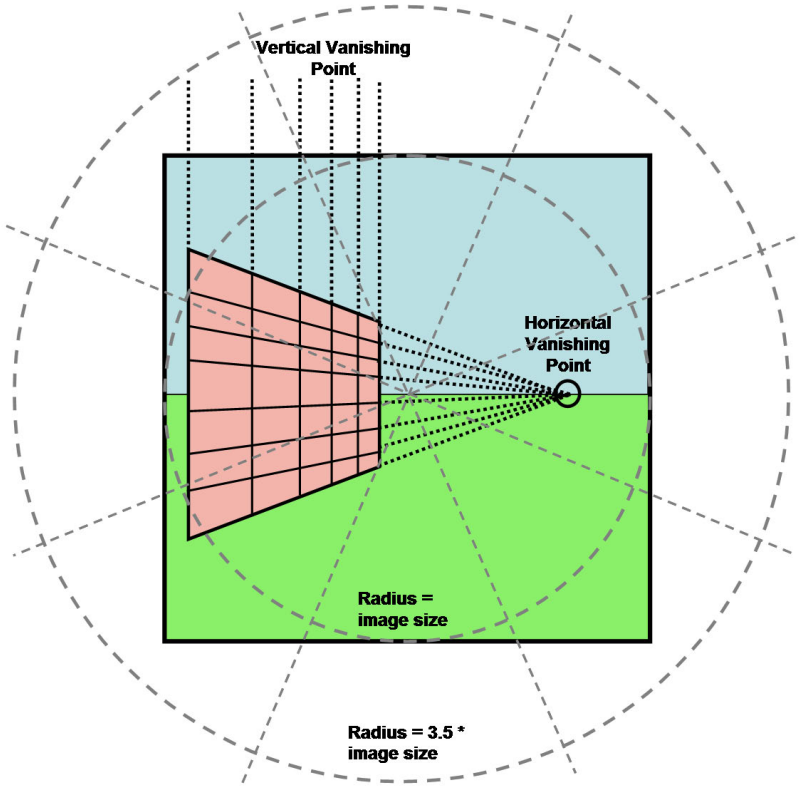
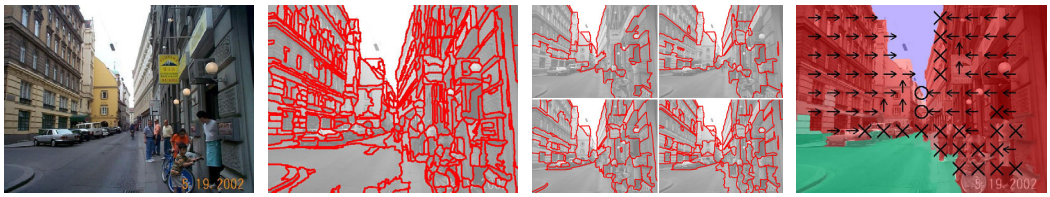


Figure 3.4: Illustration of our perspective cues. Our representation includes both a "soft" estimate, such as a radially binned histogram of edge intersections, and an explicit estimate of vanishing points.

(8 orientations) and the distance from the image center (thresholds of 1.0 and 3.5 times the image size).

We also found that computing a more explicit estimate of the vanishing points can help in some cases. We compute these vanishing points over the entire image (without regard to any segmentation), using the EM approach described by Kosecka and Zhang [**69**]. From this, we have a set of estimated vanishing points and the probability that each line in the image belongs to each vanishing point. If the expected number of lines that form a vanishing point is less than 3, we ignore that vanishing point. We then consider vanishing points that occur very high or low (outside 2.5 times the image height) in the image plane to be "vertical vanishing points", which are likely to lie on a vertical plane. We consider vanishing points that lie close to the image center (within 1.25 times the image height) to be "horizontal vanishing points", which are likely to lie on a plane parallel to the ground. Our experiments



Input Superpixels Multi Segmentations Surface Layout

Figure 3.5: Surface layout estimation algorithm. From the input image, we create an over-segmentation into superpixels. We then group those superpixels into several different segmentations. The final surface layout combines estimates from all of the segmentations.

have shown our system to be robust to reasonable variations in choices for the distance thresholds used in binning and classifying vanishing points.

For a given segment, we then compute various statistics relating to these vanishing points. For instance, the number of pixels that a segment has that contribute to a vertical or horizontal vanishing point (P8-P10) may help determine whether the surface is a support surface or vertical. Likewise, the position of the highest or lowest vanishing point (P12) formed by lines in the segment may help determine the rough surface orientation. The position of the horizontal vanishing point with respect to the segment center (P11, P13) provides evidence of whether the surface is facing to the left, center, or right of the viewer.

The texture gradient can also provide orientation cues, even for natural surfaces without parallel lines. We represent texture gradient information (P14) by computing the difference of the positional center of a segment with its gradient magnitude center of mass.

We also estimate the horizon position in the image based on the vanishing points that lie relatively close to the image center (within 75% of the image height). If more than one such vanishing point exists, a weighted average is taken of their y-positions (weighted by the inverse variance of the estimated y-position of the vanishing points). Feature set L3-L4 relates the coordinates of the segment relative to the estimated horizon, which is often more relevant than the absolute image coordinates (though we must note that this horizon estimate is often quite poor).

3. Spatial Support

Many of the cues described above can be extracted only when something is known about the structure of the scene. For instance, knowledge about the intersection of nearly parallel

lines in the image is often extremely useful for determining the 3D orientation, but only when we know that the lines belong to the same planar surface (e.g., the face of a building or the ground). Our solution is to slowly build structural knowledge of the image: from pixels to superpixels to multiple segmentations (see Figure 3.5).

3.1. Superpixels

Initially, an image is represented simply by a 2D array of RGB pixels. Without any knowledge of how those pixels should be grouped, we can compute only simple cues, such as pixel colors and filter responses. Our first step is to form superpixels from those raw pixel intensities. Superpixels correspond to small, nearly-uniform regions in the image and have been found useful by other computer vision and graphics researchers [**136, 113, 80**]. The use of superpixels improves the computational efficiency of our algorithm and allows slightly more complex statistics to be computed for enhancing our knowledge of the image structure.

Our implementation uses the graph-based oversegmentation technique of Felzenszwalb and Huttenlocher [**28**]. We use the code publicly released by those authors with the parameters $\sigma = 0.5$, $k = 100$, $min = 100$. Beginning from single-pixel regions, this method merges two regions if the minimum intensity difference across the boundary is greater than the maximum difference within the regions, with a bias towards larger regions. The advantage of this technique is that it can often group large homogeneous regions of the image together while dividing heterogeneous regions into many smaller superpixels. This often allows reasonable oversegmentations with fewer superpixels (typically around 500 for an 800x600 image). The superpixels, however, tend to be highly irregular in size and shape which could be disadvantageous for some applications. Alternative over-segmentation methods include the normalized cuts based method evaluated by Ren and Malik [**113**] and the simple watershed algorithm, which was used effectively by Li et al. [**80**]. These methods produce more regular superpixels but require much more processing time or larger numbers of superpixels.

3.2. Multiple Segmentations

Our experiments (Section 6.2) show that, while the increased spatial support of superpixels provides much better classification performance than for pixels, larger regions are required to effectively use the more complex cues, such as perspective. How can we find such regions? One possibility is to use a standard segmentation algorithm (such as normalized cuts [**124**]) to partition the image into a small number of homogeneous regions. However, since the cues used in image segmentation are themselves very basic and local, there is little chance of reliably obtaining regions that correspond to entire surfaces in the scene.

Our approach is to compute *multiple segmentations* based on simple cues and then use the increased spatial support provided by each segment to better evaluate its quality. Ideally, we would evaluate all possible segmentations of an image to ensure that we find the best one. To make our algorithm tractable, we sample a small number of segmentations that are representative of the entire distribution. We compute the segmentations using a simple method (described in Figure 3.7) that groups superpixels into larger continuous segments. Our method is based on pairwise same-label likelihoods, which are learned from training images. A diverse sampling of segmentations is produced by varying the number of segments n_s and using a random initialization. In our implementation, we generate 15 segmentations of the input image, with $n_s \in \{5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60, 70, 80, 90, 100\}$). In Figure 3.6, we show multiple segmentation examples for two images.

Our multiple segmentation method has advantages of being task-based, efficient, and empirically able to generate a reasonable sampling of segmentations. Other methods, however, are also possible. For instance, Russell et al. [**115**] generate multiple segmentations using Normalized Cuts [**124**] while varying the size of the image and the number of segments. Other possibilities include varying the cues used for the segmentation [**111**] or generating a hierarchical segmentation [**1, 123, 3**].

3.3. Labeling

Each segmentation provides a different view of the image. To arrive at a final solution, we need to evaluate the likelihood that each segment is good or *homogeneous* and the likelihood of each possible label. We can then combine estimates produced by different segmentations

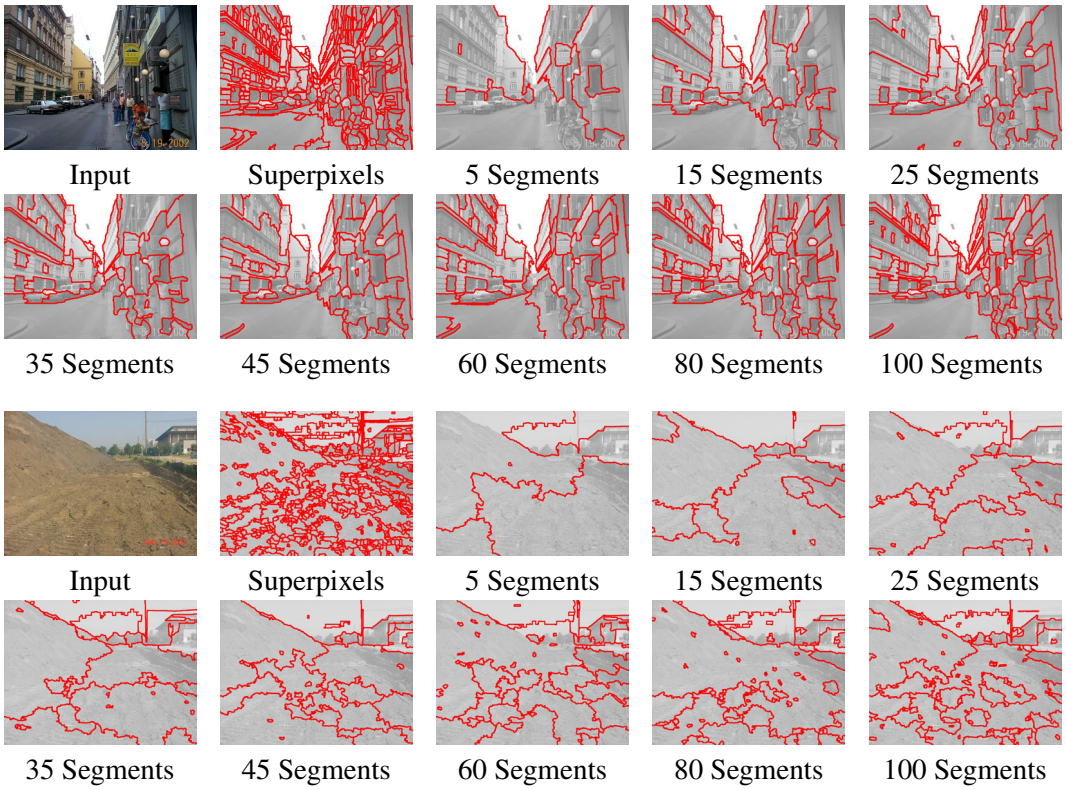


Figure 3.6: Examples of multiple segmentations.

in a probabilistic fashion. A segment s_k is homogeneous if all superpixels within it have the same label ($\forall i, j \in s_k : y_i = y_j$). We estimate the homogeneity likelihood $\mathrm{P}(s_k|\mathbf{I})$ based on all of the cues listed in Table 3.1 using boosted decision trees[1]. Homogeneity is very difficult to estimate from image data $\mathbf{I}$, however, and the learned estimate depends heavily on the number of superpixels in a segment. If we know that a segment is homogeneous, we can estimate the likelihood of its label $\mathrm{P}(\tilde{y}_k|s_k, \mathbf{I})$.

To get the label likelihood for the i^{th} superpixel, we simply marginalize over the unique sampled segments $\{s_k\}$ that contain the superpixel:

$$\mathrm{P}(y_i = l|\mathbf{I}) \propto \sum_{s_k \ni i} \mathrm{P}(s_k|\mathbf{I})\mathrm{P}(\tilde{y}_k = l|s_k, \mathbf{I}). \tag{3.1}$$

Here, y_i is the superpixel label and $\tilde{y}_k$ is the segment label. Since the sum is over segments containing the i^{th} superpixel, rather than the joint space of segmentations, we believe that

[1] In an attempt to create a better homogeneity estimator, we also tried using statistics of individual superpixel likelihood estimates and same-label likelihoods within a segment. The classifier learned from these statistics, however, performed the same as the classifier using the original cues, both in terms of its ability to classify homogeneous segments and of the overall accuracy of the system.

SEGMENTATION

Input:

- n_s: number of segments
- $\mathrm{p}_{ij} = \mathrm{P}(y_i = y_j|\mathbf{I})$: probability that adjacent superpixels r_i and r_j have same label, given the image (Section 4)

Initialize:

1. Assign n_s random superpixels to segments $1..n_s$
2. While any unassigned superpixels remain
   For each unassigned m_i with at least one assigned neighbor:
   For each assigned neighboring segment k:
   Assign $m_i = k$ with probability $\prod_j \left(\mathrm{p}_{ij}^{\mathbf{1}(m_j=k)}(1-p_{ij})^{\mathbf{1}(m_j\neq k)}\right)$

Minimize energy: $\mathrm{E}(\mathbf{m}) = -\sum_{ij}\log\left(\mathrm{p}_{ij}^{\mathbf{1}(m_i=m_j)}(1-p_{ij})^{\mathbf{1}(m_i\neq m_j)}\right)$

- For each m_i: $m_i = \mathrm{argmin}_{m_i}\mathrm{E}(\mathbf{m})$ until a local minimum of $\mathrm{E}(\mathbf{m})$ is reached (until $\mathbf{m}$ remains unchanged for one iteration)

Output:

- $m_1..m_{n_r} \in \{1..n_s\}$: assignment of superpixels to continuous segments

Figure 3.7: Pseudocode for segmentation. The product in the initialization is taken over all adjacent superpixels to r_i that have been assigned. The sum in the energy term is over all pairs of adjacent superpixels. During the energy minimization, the assignment of m_i is constrained to preserve segment continuity. Here, $\mathbf{1}(.)$ is the indicator function.

a small number of segmentations will provide a sufficiently large sample. Our experiments (Section 6.5) appear to support this conclusion.

The resulting superpixel label likelihoods can be thresholded to get a max-margin estimate of the superpixel labels (as we do in most of our evaluation) or used in a Markov random field framework to produce the jointly most probable labels. In Section 7.1, we evaluate a simple implementation of the latter, forming unary potentials from the the label likelihood and pairwise potentials from the same-label likelihoods and maximizing using graph cuts.

4. Classifiers

In this chapter, we use boosted decision trees for each classifier, using the logistic regression version of Adaboost [**16, 30**]. Decision trees make good weak learners, since they provide automatic feature selection and limited modeling of the joint statistics of data. Each decision tree provides a partitioning of the data and outputs a confidence-weighted decision

TRAINING BOOSTED DECISION TREES

Input:

- $\mathbf{D}_1..\mathbf{D}_m$: training data
- $w_{1,1}..w_{1,m}$: initial weights
- $y_1..y_m \in \{-1,1\}$: labels
- n_n: number of nodes per decision tree
- n_t: number of weak learner decision trees

For $t = 1..n_t$:

1. Learn n_n-node decision tree T_t based on weighted distribution $\mathbf{w}_t$
2. Assign to each node $T_{t,k}$: $f_{t,k} = \frac{1}{2}\log\frac{\sum_{i:y_i=1,\mathbf{D}_i\in T_{t,k}} w_{t,i}}{\sum_{i:y_i=-1,\mathbf{D}_i\in T_{t,k}} w_{t,i}}$
3. Update weights: $w_{t+1,i} = \frac{1}{1+\exp(y_i\sum_{t'=1}^{t} f_{t',k_{t'}})}$ with $k_{t'} : D_i \in T_{t',k_{t'}}$
4. Normalize weights so that $\sum_i w_{t+1,i} = 1$

Output:

- $T_1..T_{n_t}$: decision trees
- $f_{1,1}..f_{n_t,n_n}$: weighted log-ratio for each node of each tree

Figure 3.8: Boosted decision tree algorithm using logistic regression version of Adaboost.

which is the class-conditional log-likelihood ratio for the current weighted distribution. The logistic regression version of Adaboost differs from the original confidence-weighted version [**121**] by only a slight change in the weight update rule, but it results in confidence outputs that tend to be well-calibrated probabilities (after applying the simple sigmoid conversion to the log-ratio output).

The classifier training algorithm is given in Figure 3.8. For the same-label classifier, the initial weighted distribution is uniform. For the segment classifiers, the initial distribution is proportional to the percentage of image area spanned by each segment, reflecting that correct classification of large segments is more important than of small segments. When computing the log-likelihood ratio, we add a small constant ($\frac{1}{2m}$ for m data samples) to the numerator and denominator which helps to prevent overfitting and to stabilize the learning process.

The same-label classifier outputs an estimate of $\mathrm{P}(y_i = y_j|\mathbf{I})$ for the adjacent superpixels i and j and image data $\mathbf{I}$. The segment homogeneity classifier outputs $\mathrm{P}(s_k|\mathbf{I})$ for the k^{th} segment. We train separate classifiers to distinguish among the main classes and the subclasses of vertical. These are each learned in a one versus all fashion. For instance, to distinguish

Support	Vertical	Sky
31.4%	48.0%	20.6%

Left	Center	Right	Porous	Solid
5.0%	10.5%	6.3%	15.7%	10.5%

Table 3.2: Average image area for each main class and subclass of vertical.

among the main classes, we train three classifiers that estimate the probability of a segment being "support", "vertical", and "sky". These are then normalized to ensure that the estimated probabilities sum to one. Similarly, the subclassifier outputs probabilistic estimates for each subclass, given that the main class is "vertical".

5. Implementation

We use our Geometric Context dataset (described in Chapter 1) for training and testing. Table 3.2 shows the average image area of each main class and subclass in our data set.

Training and testing is performed with cross-validation. Our training algorithm is outlined in Figure 3.9, and our inference algorithm in Figure 3.10. We randomly split the set of 300 images into six subsets of 50 images each. The first subset is used to train a same-label classifier that estimates the likelihood that two superpixels have the same label. The remaining subsets are used for training and testing in five-fold cross-validation. Based on the same-label classifier, multiple segmentations are produced for each image (as in Section 3.2). Then, in each fold, four subsets are used to train the geometry and homogeneity classifiers, and the remaining subset is used for testing. Since we use the cross-validation to evaluate our algorithm, we do not use it to select parameters.

The same-label classifier is based on cue sets L1, L6, C1-C4, and T1-T2 in Table 3.1. The following statistics are computed over pairs of superpixels: the absolute differences of the mean RGB, HSV, filter response, and pixel location values; the symmetrized Kullback-Leibler divergence of the hue, saturation, and texture histograms; the ratio of the areas; the fraction of the boundary length divided by the perimeter of the smaller superpixel; and the straightness (length divided by endpoint distance) of the boundary.

To train the segment classifiers, we need to assign ground truth to the automatically created segments. If nearly all (at least 95% by area) of the superpixels within a segment have

TRAINING OVERVIEW

1. For each training image:
   (a) Compute superpixels
   (b) Compute superpixel cues (Table 3.1)
2. Train same-label classifier (Section 4)
3. For each training image:
   (a) Produce multiple segmentations for varying n_s (Section 3)
   (b) Label each segment according to ground truth
   (c) Compute cues in each segment (Table 3.1)
4. Train segment label classifier and homogeneity classifier (Section 4)

Figure 3.9: Outline of training procedure.

the same ground truth label, the segment is assigned that label. Otherwise, the segment is labeled as "mixed". The label classifier is then trained to distinguish among single-label segments, and the homogeneity classifier is trained to determine whether a segment has a single label or is mixed. The segment classifiers use all of the listed cues. Many of these cues can be quickly computed for the segments, since the superpixel cues provide sufficient statistics. In fact, all of the location/shape, color, and texture cues, with the exception of L2 and L3, can be computed for segments by taking a weighted (by area) mean of the superpixel cue values.

In our implementation, each strong classifier consists of 20 decision trees that each have 8 leaf nodes. Decision trees are learned using a weighted version of the MATLAB® `treefit` function. We first grow the tree to four times the number of nodes desired and then prune it using the MATLAB® `treeprune` function.

6. Experiments

> *Write the vision, and make it plain upon tables, that he may run that readeth it.*
> *Bible, Habakkuk* ii. 2.

The purpose of our experiments is to demonstrate the effectiveness of our proposed multiple segmentation algorithm and to gain an understanding of what makes it effective. Our analysis includes a comparison of varying levels of spatial support, the impact of different types of cues on accuracy, and choices of parameters for segmentation and classification.

SURFACE LAYOUT ESTIMATION

1. Image → superpixels via over-segmentation
2. Superpixels → multiple segmentations
   (a) For each superpixel: compute cues (Table 3.1)
   (b) For each pair of adjacent superpixels: compute same-label likelihood $\mathrm{P}(y_i = y_j|\mathbf{I})$
   (c) Create multiple segmentations for varying n_s (Section 3)
3. Multiple segmentations → superpixel labels
   (a) For each segment:
      (i) Compute cues (Table 3.1)
      (ii) For each possible label (main classes and subclasses): compute label likelihood $\mathrm{P}(\tilde{y}_k|\mathbf{I}, s_k)$
      (iii) Compute homogeneity likelihood $\mathrm{P}(s_k|\mathbf{I})$
   (b) Compute label confidences for each superpixel: $\mathrm{P}(y_i|\mathbf{I}) \propto \sum_{s_k \ni i} \mathrm{P}(\tilde{y}_k|\mathbf{I}, s_k)\mathrm{P}(s_k|\mathbf{I})$

Figure 3.10: Outline of surface layout estimation algorithm.

In each experiment, we report the average accuracy of classification among the main classes ("support", "vertical", "sky") and among the subclasses of vertical ("left", "center", "right", "porous", "solid"). The most likely label is assigned to each superpixel, and the average accuracy is weighted by the area of the image spanned by each superpixel. For instance, an accuracy of 85% means that, on average, 85% of the pixels in an image are correctly labeled. Subclass accuracy is evaluated independently of the main class accuracy. For example, if a "porous", "vertical" superpixel is mislabeled as ground, but the most likely subclass given vertical is porous, that superpixel is counted as incorrect for the main class accuracy but correct for the subclass accuracy.

Overall (Section 6.1), the performance of our method is quite good, both quantitatively and qualitatively. Key ingredients to its success are the robust spatial support (6.2) provided by multiple segmentations and the inclusion of many types of cues (6.3). The spatial support plays an especially important role in the subclassification (e.g., determining the whether a plane is facing the right or the left), which greatly benefits from perspective cues. Fortunately, the algorithm is not highly sensitive to implementation details such as the number of segmentations (6.5) and classification parameters (6.6). Our method also easily extends to indoor images (6.7), achieving excellent accuracy despite a small training set.

Main Class			
	Support	Vertical	Sky
Support	**0.84**	0.15	0.00
Vertical	0.09	**0.90**	0.02
Sky	0.00	0.10	**0.90**

Vertical Subclass					
	Left	Center	Right	Porous	Solid
Left	**0.37**	0.32	0.08	0.09	0.13
Center	0.05	**0.56**	0.12	0.16	0.12
Right	0.02	0.28	**0.47**	0.13	0.10
Porous	0.01	0.07	0.03	**0.84**	0.06
Solid	0.04	0.20	0.04	0.17	**0.55**

Table 3.3: Confusion matrices (row-normalized) for multiple segmentation method.

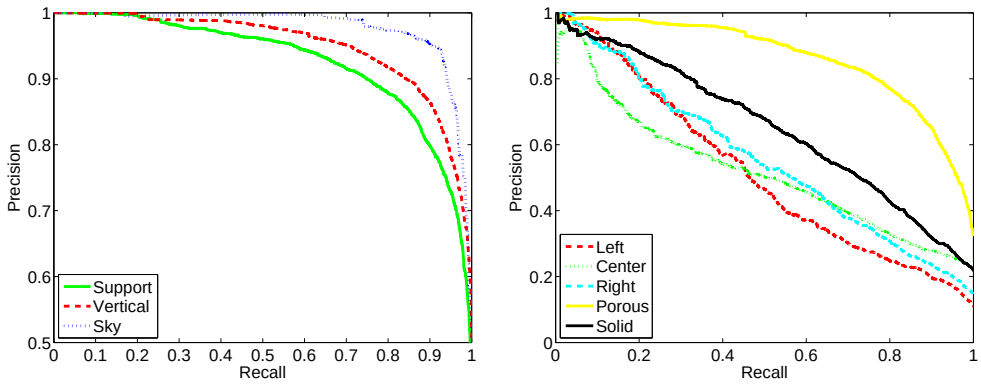


Figure 3.11: Precision-recall curves. Precision is the percent of declared labels (at some confidence threshold) that are true, and recall is the percent of true labels that are declared. The curve is generated by varying the confidence threshold.

6.1. Overall

In Table 3.3, we report the confusion matrices for our multiple segmentation method. The matrices are row-normalized. The non-normalized matrices can be computed using the class priors (Table 3.2).

For some applications, only the accuracy of the most likely label is important, but for others the confidence in that label is helpful. In Figure 3.11, we plot the precision-recall curves for the main and subclasses.

Method	Main	Sub
Pixels	82.1	44.3
Superpixels	86.2	53.5
Single Segmentation	86.2	56.6
Multiple Segmentations	**88.1**	**61.5**
Ground Truth Segmentation	95.1	71.5

Table 3.4: Average accuracy (percent of correctly labeled image pixels) of methods using varying levels of spatial support.

In Figures 3.15 and 3.16, we show qualitative results from our multiple segmentation method, and, in Figure 3.17, we display results as confidence images for each of the surface labels. We demonstrate the generality of our geometric models in Figure 3.19, where we label paintings, despite training on only real images.

6.2. Spatial Support

In Table 3.4, we report the average accuracy for increasing levels of spatial support, from pixels to multiple segmentations. When using only individual pixels as spatial support, based on color, filter responses, and position, the classification accuracies are 82.1% for the main classes and 44.3% for the subclasses. When using superpixels for spatial support, based on the simpler superpixel cues (L1, L6, C1-C4, T1-T2 in Table 3.1), the accuracies improve substantially to 86.0% and 52.9%, respectively. We find that including the more complex cues, such as perspective, in the superpixel classification improves accuracy only slightly (by 0.2% for main classes and 0.6% for subclasses). Thus, while superpixels allow more useful cues than pixels, such as histograms of color and texture, the perspective cues still cannot be utilized effectively without better spatial support.

A single segmentation provides better spatial support than superpixels but is unreliable and may result in a poor solution. Segmenting images into 100 segments each (the single value that gives the highest average accuracy in our experiments) slightly improves the subclass accuracies (over superpixels) to 56.6%. Our multiple segmentation method provides good spatial support while avoiding commitment to any particular segmentation, resulting in much better performance (88.1% and 61.5%). As can be seen, better spatial support results in higher accuracy, especially for the subclasses which rely on the more complex cues.

We also performed an experiment based on "perfect" segmentations, which were created by performing connected components on the ground truth labels. When training and testing on these ground truth segmentations, our algorithm achieves much higher accuracy (95.1% and 71.5%). This result demonstrates the value of spatial support for classification and indicates the potential improvement that could result from an improved segmentation process.

6.3. Analysis of Cues

We wish to evaluate the effectiveness of our four types of cues: location and shape, color, texture, and perspective (see Table 3.1). To do this, we train classifiers and compute average accuracies over our test images in eight trials: using each type individually and using all cues except one type. In these experiments, we applied the multiple segmentation method, using the same segmentations as were used to report accuracy on the multiple segmentations. Thus, the segmentations themselves are based on the standard set of cues, and the results in this experiment reflect how effectively the segments are classified. The results are listed in Table 3.5. In Figure 3.18, we display qualitative results, comparing the individual contribution of each cue and the overall result.

The simple location and shape cues prove to be surprisingly effective for the main classes but ineffective for distinguishing among the vertical subclasses. Texture cues are highly effective for both main classes and subclasses. Perspective seems to be highly effective for subclassification but to offer little benefit to main classification. Perhaps texture implicitly provides the useful perspective information for distinguishing between "support" and "vertical", or perhaps this is due to the low number of "manhattan" scenes in the database for which the perspective cues would be most valuable. From this table, we can also conclude that the cues have high interdependencies through the labels. Each cue by itself seems remarkably effective at discriminating among the classes (especially for the main classes), but removing a single cue never affects accuracy by more than a few percent. The effect on accuracy would be larger (as shown in our previous work [**48**]) if the segmentations were also recomputed with subsets of the cues. For instance, color plays a stronger role in the segmentation than in the classification.

Cues	Main	Sub
All	**88.1**	**61.5**
Location	82.9	42.1
Color	72.2	43.1
Texture	79.9	54.6
Perspective	68.3	51.8
No Location	84.4	59.5
No Color	87.0	60.4
No Texture	86.7	58.2
No Perspective	88.1	56.6

Table 3.5: Average accuracy under different sets of cues.

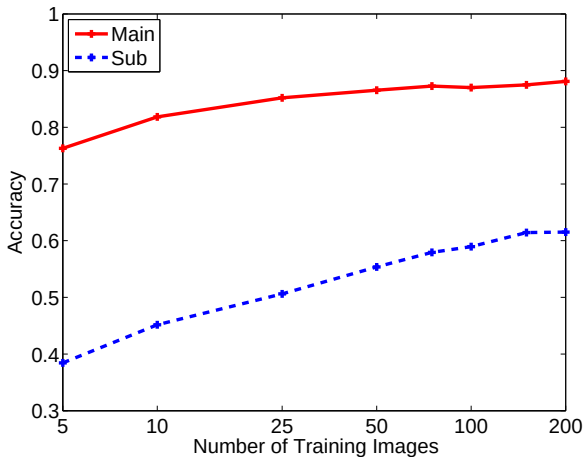


Figure 3.12: Accuracy while varying the number of training images. According to these trends, much larger training sets would probably result in significantly higher accuracy, especially for the subclasses.

6.4. Effect of Training Data Size

If we had more training images, would we be able to train more accurate classifiers? To find out, we performed an experiment, re-training our segment homogeneity and label classifiers on between 5 and 200 images. We plot the results in Figure 3.12, providing evidence that much larger training sets would probably improve accuracy, especially for the subclasses.

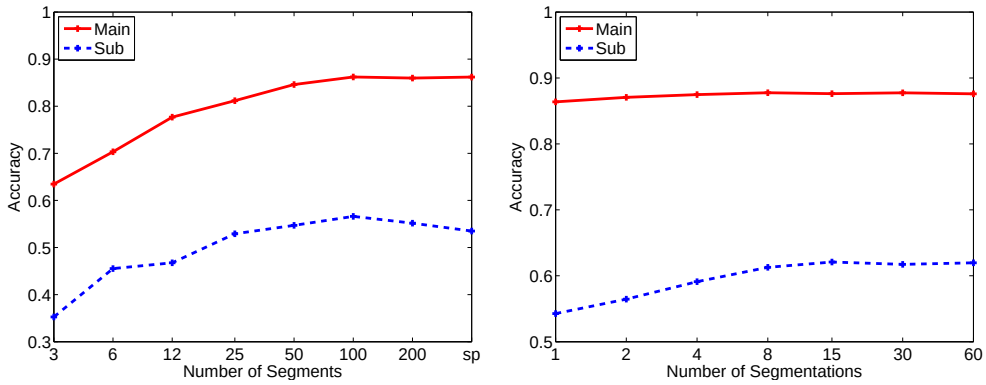


Figure 3.13: Analysis of segmentation parameters. On the left, classification is performed for single segmentations into varying numbers of segments ("sp" is the segmentation into superpixels). Peak accuracy is at 100 segments, with larger numbers of segments degrading the subclass accuracy slightly. On the right, classification is performed using the multiple segmentation method for varying numbers of segmentations. Although eight segmentations outperforms single segmentations by about 2% and 5% for main and subclasses, increasing the number of segmentations further produces no significant change in accuracy.

6.5. Analysis of Segmentations

We first evaluate the effect of the number of segments on accuracy, for the single segmentation method. We plot the results of our experiment in Figure 3.13 (left). The main class accuracy peaks at 100 segments per segmentation and remains unchanged for higher numbers. The subclass accuracy also peaks at 100 segments per segmentation but declines slightly as the number of segments increases further. This may be due to the importance of the perspective cues (which require good spatial support) to the subclassifier. These results illustrate the trade-off between bias and spatial support, showing that in the case of a single segmentation, low bias (an oversegmentation) is preferred.

We also measure the effect of the number of segmentations on accuracy in our multiple segmentation framework. Our implementation uses 15 segmentations. In Figure 3.13 (right), we compare results while varying the number of segmentations from one to sixty. In these experiments, we used the same classifiers trained under our reported results for multiple segmentations but generated new sets of segmentations for testing. A large improvement is observed for the subclassifier, with a smaller improvement for the main classifier. This

is in accordance with the results in Table 3.4, which shows that segmentation plays a more critical role in the subclassification.

6.6. Classification Parameters

Boosted decision trees sequentially carve the input space into a set of hypercubes and estimate the class-conditional likelihood ratio for each. The granularity of the hypercubes can be made arbitrarily fine (as the number of weak learners increases), but the estimations of the likelihood ratios (and the assigned classes) for datapoints falling within each cube are constrained by the type of decision tree. For instance, if the decision tree has only two leaf nodes (as is common practice in computer vision algorithms), the strong classifier is based on a sum of functions that each take only one attribute as input, as in linear logistic regression. If the decision tree can be arbitrarily large, however, any set of unique datapoints can be arbitrarily labeled.

Thus, when using boosted decision trees, there are two important parameters to consider: the number of weak learner trees and, more importantly, the number of nodes per tree. Generally, increasing the number of trees will not harm accuracy, since Adaboost is robust to overfitting. Choosing the number of nodes per tree, however, involves the complexity trade-off between the power of the classifier and its tendency to overfit. We performed experiments, comparing the average accuracy as the number of decision trees varied for eight-node trees (eight leaf nodes) and while varying the number of nodes per tree. When changing the number of nodes per tree, we keep the total number of leaf nodes fixed at 160 for each strong classifier (80 2-node trees, 40 4-node trees, 20 8-node trees, 10 16-node trees, 5 32-node trees). See Figure 3.14 for the results.

6.7. Indoor Scenes

We designed the cues and approach of our algorithm for outdoor images, thinking that indoor images may require a different approach to take advantage of the high structure of building interiors. However, after seeing results of the indoor 3D reconstruction method of Delage, Lee, and Ng [**23**], we decided to measure the accuracy of our algorithm on their indoor dataset. We annotated ground truth geometry labels for the Stanford dataset of 92 indoor images used by Delage et al.. For simplicity, we used the same geometric classes as

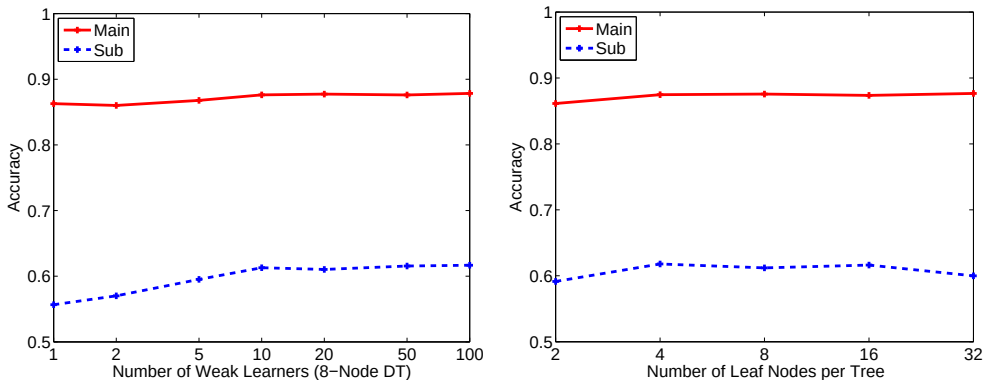


Figure 3.14: Analysis of classification parameters. The results are fairly robust to the classification parameters. When boosting eight-node decision trees, having ten or more weak learners gives the best results. When keeping the total number of decisions constant (160 nodes total), between trees of between 4 and 16 nodes produce the best results.

for outdoors, except that the "sky" class is redefined as the ceiling. We did not change any parameters in our system and re-used the same-label classifier trained on outdoor images. We then performed two experiments. In the first, all classifiers are trained on outdoor images. In the second, the homogeneity and label classifiers are trained on indoor images in five-fold cross-validation.

When trained on outdoor images, the average classification accuracy of indoor images is 76.8% for the main classes and 44.9% for the subclasses. After re-training the segment classifiers on indoor images, the test accuracy improves to 93.0% and 76.3%, respectively. Qualitative results are shown in Figure 3.20. These results are quantitatively better than our results for outdoor images, likely due to the ease of segmentation and highly structured scenes (for instance, many are hallways) in this indoor dataset. Our results show that our approach generalizes surprisingly well to new types of scenes, especially when trained appropriately.

7. Alternative Frameworks

Since our method is the first to treat surface estimation as a recognition problem, it is worth checking that other reasonable approaches will not greatly outperform our own. Here, we explore two alternative frameworks for recovering surface layout: a simple random field

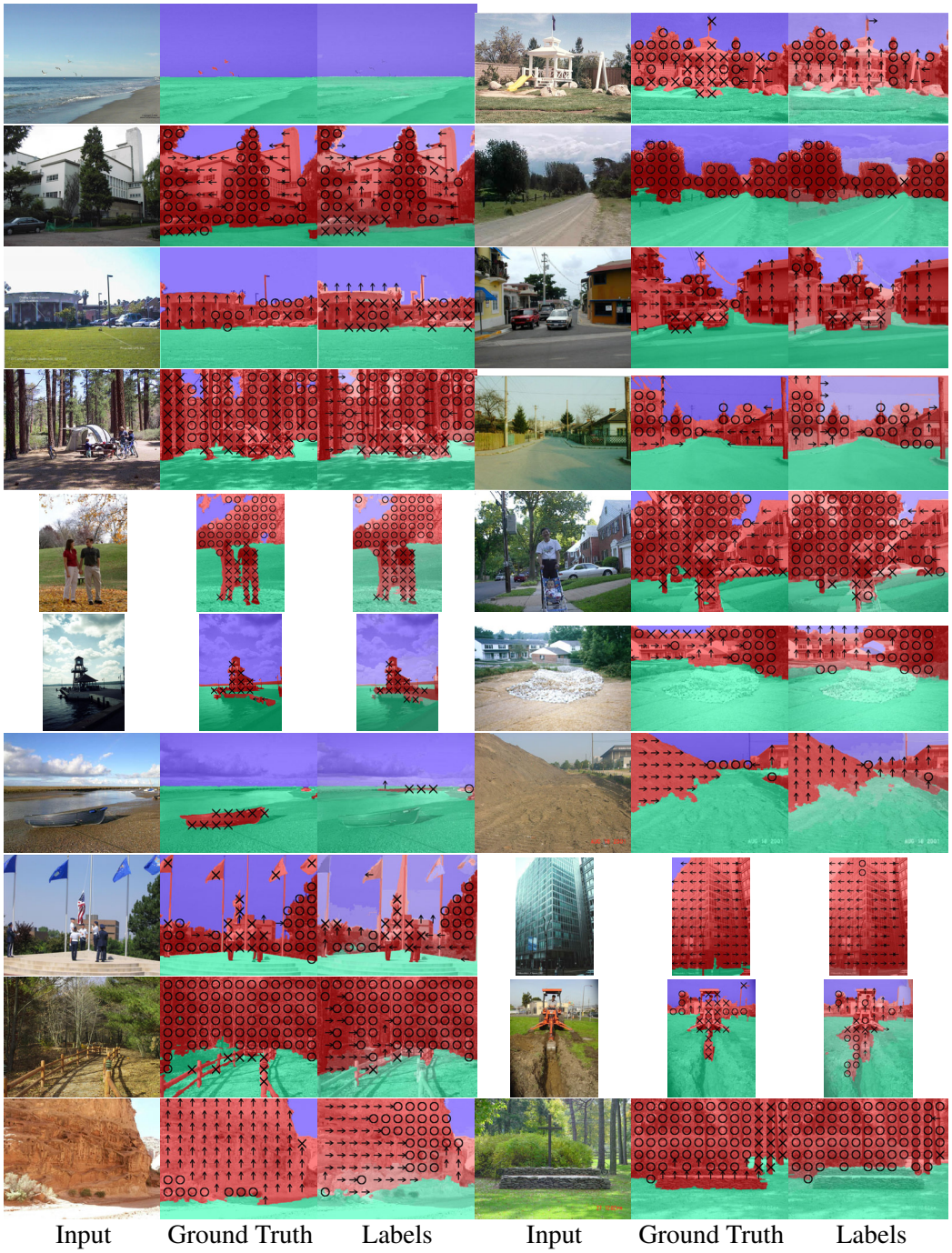


Figure 3.15: Results from multiple segmentations method. This figure displays an evenly-spaced sample of the best two-thirds of all of our results, sorted by main class accuracy from highest (upper-left) to lowest (lower-right). See Figure 3.1 for explanation of colors and markings. Brighter colors indicate higher levels of confidence for the main class labels.

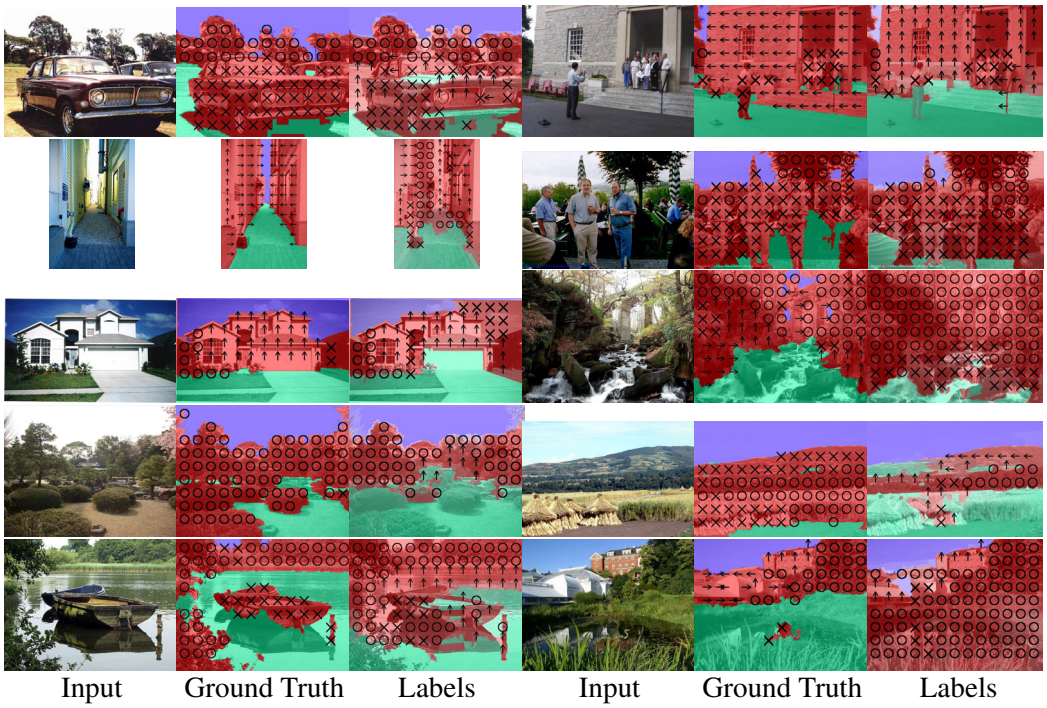


Figure 3.16: Continued results from multiple segmentations method. This figure displays an evenly-spaced sample of the worst third of all of our results, sorted by main class accuracy from highest (upper-left) to lowest (lower-right).

framework in which unary and pairwise potentials are defined over superpixels, and a simulated annealing approach that searches for the most likely segmentation and labeling. Our results from the random field framework confirm that our multiple segmentations provide more than what can be modeled by simple pairwise interactions. Our results on the simulated annealing framework highlight the difficulty of searching for a single good segmentation, reaffirming our own approach.

7.1. Random Field Framework

In image labeling problems, conditional random fields [**74**] (CRFs) are widely used to model the conditional distribution $\mathrm{P}(\mathbf{y}|\mathbf{I})$ of the labels given the image data. Using pairwise cliques, the log of of the conditional distribution can be written as a sum of unary and pairwise potential terms:

$$\log \mathrm{P}(\mathbf{y}|\mathbf{I}) = \sum_i \mathrm{f}_1(y_i, \mathbf{I}) + \sum_{ij} \mathrm{f}_2(y_i, y_j, \mathbf{I}) - \log Z \tag{3.2}$$

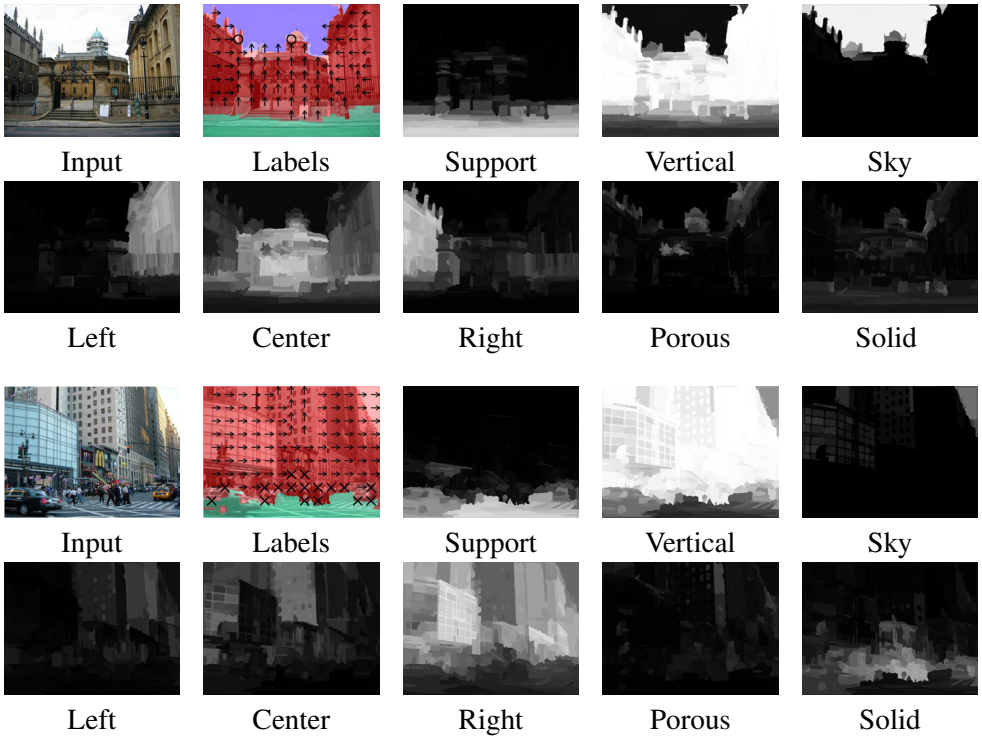


Figure 3.17: Results with confidence images for multiple segmentations method. The confidence image shows the estimated probabilities over the image for the given label.

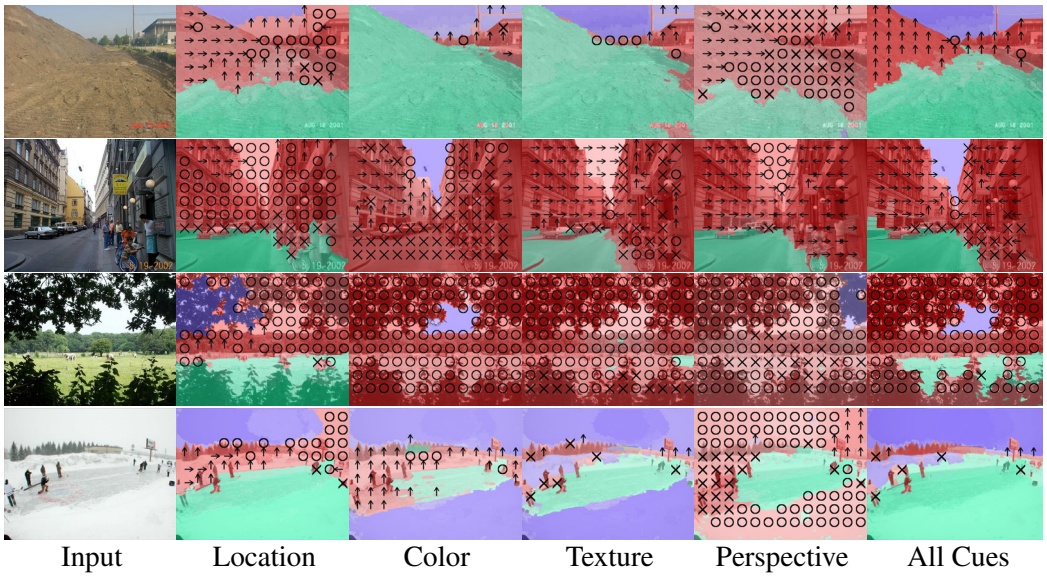
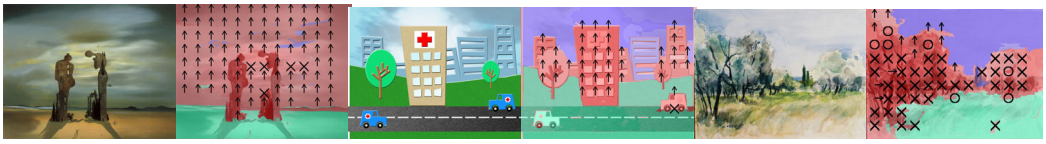


Figure 3.18: Results when performing classification based on each cue separately and using all cues. In each case, the same multiple segmentations are used (which are based on location, color, and texture), and those segments are analyzed with the given type of cues.



Input Labels Input Labels Input Labels

Figure 3.19: Results on paintings of outdoor scenes. Although the system is trained only on real images, it can often generalize to very different settings.

Input Ground Truth Labels Input Ground Truth Labels

Figure 3.20: Results on indoor scenes.

where the second sum is over all adjacent pairs of labels and Z is the partition function. Each superpixel has a label, with the unary term defined as the superpixel label log likelihood $\log \mathrm{P}(y_i|\mathbf{I})$. We set the pairwise term to be proportional to the same-label log likelihood.

We can rewrite this as an energy function:

$$\mathrm{E}(y_1..y_n) = \sum_i \mathrm{E}_i(y_i) + \sum_{ij} \mathrm{E}_{ij}(y_i, y_j) \tag{3.3}$$

The first term penalizes lack of confidence in the superpixel label: $\mathrm{E}_i(y_i) = -\log \mathrm{P}(y_i|\mathbf{I})$. The second term penalizes discontinuity between neighboring labels. Denoting $\mathrm{P}(y_i = y_j|\mathbf{I})$ as p_{ij}, we set $\mathrm{E}_{ij}(y_i = y_j) = 0$ and $\mathrm{E}_{ij}(y_i \neq y_j) = \beta(\log \mathrm{p}_{ij} - \log(1 - \mathrm{p}_{ij}))$. By forcing $\mathrm{E}_{ij}(y_i \neq y_j) >= 0$, we can find a good local minimum of this energy using the alpha-expansion graph cuts algorithm [**9**]. The parameter β controls the relative strength of the pairwise interaction term.

β	0.0	0.5	1.0	1.5	2.0	2.5	3.0
Main	86.2	86.4	86.2	86.1	85.7	85.4	84.6
Sub	53.5	54.4	54.8	54.1	53.7	52.7	52.8

Table 3.6: Average accuracies for varying levels of β using a conditional random field.

We trained the superpixel label and pairwise likelihood functions as described for the multiple segmentation method (Section 3). In Table 3.6, we show that inclusion of the smoothness term provides a slight gain in subclass accuracy over the unary superpixel classification, but the results overall are not as good as for our multiple segmentation approach.

The unary potentials could also be created from the multiple segmentation label estimates. In our experiments, however, this does not improve either main class or subclass accuracy, probably because the same-label likelihoods already impact the estimate through the segmentations. By modeling other kinds of interactions, such as positional relations of labels, the conditional random field may be able to improve the results from multiple segmentations.

7.2. Simulated Annealing

In our multiple segmentation framework, we independently generate the segmentations and then marginalize over them. But what if we search for the segmentation that gives us the highest confidence labeling and use those labels? To find out, we tried a simulated annealing approach similar to that of Barbu and Zhu [**4**].

We obtain our initial segmentation by estimating the label likelihoods for each superpixel and performing connected components after coloring with the most likely labels. Training and testing a segment classifier on this initial segmentation yields an average accuracy of 86.5% for the main classes and 55.2% for the subclasses. For each move, we select a random segment. We then merge the segment with an adjacent segment or split the segment in two. If merging, the adjacent segment is chosen with likelihood proportional to the change in $\exp(-\mathrm{E}(\mathbf{m}))$, the product of adjacent superpixel pairwise likelihoods (defined in Figure 3.7). When splitting, two superpixels are randomly selected, and the algorithm in Figure 3.7 is used to assign all superpixels in the segment to one of the two parts. Then, one part of the segment either becomes a new segment or is attached to a neighboring segment.

Our energy function is based on the estimated accuracy of the main class labeling given the current segmentation. To estimate the accuracy, we learn a regression tree (using MATLAB's `treefit`) that estimates the percentage of pixels within a segment that have the majority label, based on the set of cues in Table 3.1. We also learn a classifier that outputs the likelihood of the majority label (even if the segment contains multiple types of surfaces). The energy is the defined as

$$\mathrm{E}(\mathbf{s};\mathbf{I}) = -\log \sum_k a_k \max_l \mathrm{P}(\tilde{y}_k = l | s_k, \mathbf{I}) \mathrm{r}(\tilde{y}_k = l, s_k | \mathbf{I}) \tag{3.4}$$

where a_k is the (normalized) area of the segment s_k, $\tilde{y}_k$ is its label, and $\mathrm{r}(\tilde{y}_k, s_k|\mathbf{I})$ is the regression tree estimate. Each move is accepted with probability $\exp(-\frac{\Delta \mathrm{E}}{T_m})$ where T_m is the temperature at the current iteration.

We used a low initial temperature (T_0 set as one percent of initial energy) and a fast cooling schedule ($T_{m+1} = 0.95 * T_m$ after n_s proposals, where n_s is the number of segments at the beginning of the iteration). Even so, the simulated annealing took about 26 hours to test the 250 images. On average, the energy at the final iteration was about 40% of the initial energy.

The result of the simulated annealing is an average accuracy of 85.9% for the main classes and 61.6% for the subclasses. If, instead of using only the final segmentation, we use an average over all of the segments produced during the simulated annealing (similarly to the multiple segmentation algorithm), the main class accuracy is 85.3% and the subclass accuracy is 66.9%. Thus, we are able to achieve much higher accuracy (about 5% higher) for subclasses than with our multiple segmentation algorithm, but the main class accuracy, which is more important for many applications, is about 2% lower.

The challenge of the simulated annealing approach is in determining the majority label and the percentage of pixels in the segment that have the majority label. This is much more difficult to estimate than the likelihood that a segment is homogeneous or the label likelihood of a homogeneous segment.

8. Conclusion

By posing surface estimation as a recognition problem, we are able to recover the coarse surface layout in a wide variety of scenes. Our approach has the advantages of simplicity

and robustness, being able to generalize even to paintings and indoor images. One important aspect of our approach is the use of a wide variety of image cues including position, color, texture, and perspective. Different cues provide different types of information about a region, and, when used together, they are quite powerful. The idea of multiple segmentations is also crucial to the success of our algorithm, especially for distinguishing among the subclasses. Multiple segmentations acquire the spatial support necessary for complex cues while avoiding the risky commitment to a single segmentation.

However, a more complete notion of surface layout is required. As Gibson says, spatial perception is nothing but "edge and surface", but we have only surface orientation. Without knowledge of the occluding contours, we cannot determine whether neighboring regions in the image are physically connected. In the next chapter, we propose a method to recover the major occlusion boundaries from an image. With such knowledge, we can separate the person from the row of cars behind him and determine the rough layout of objects in cluttered scenes.

CHAPTER 4

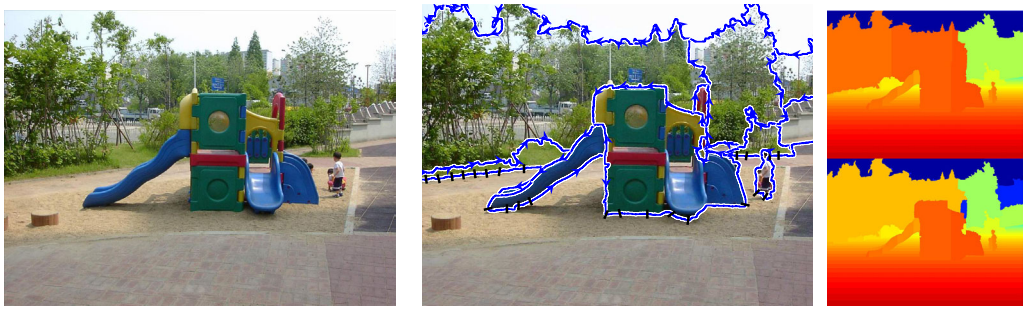
Recovering Major Occlusion Boundaries

In the country of the blind the one-eyed man is the king.
Desuderious Erasmus (1469-1536)

Surface orientations provide an incomplete description of spatial layout in all but the simplest of scenes. To provide a more complete understanding, we must reason about occlusion, the concept that spatially separated 3D objects can interfere with each other in the projected 2D image plane.

Consider the scene in Figure 4.1: nearly every object is partially occluded by some other object, and each occludes part of the ground. Yet, despite their pervasiveness, occlusions have too often been ignored. For example, in Byzantine painting, artists took great care to place objects (such as human figures) far away from each other to avoid any overlap, and only after Giotto at the start of the Italian Renaissance was the importance of occlusion in art fully realized (Figure 4.2). In computer vision, the study of occlusion reasoning has largely been confined to the context of stereo, motion and other multi-view problems. For single-view tasks, such as object recognition, occlusions are typically considered a nuisance that must be dealt with by making the algorithm more robust, and heavily occluded objects are often considered unimportant or too difficult to detect.

In this chapter, we argue that occlusion reasoning lies at the core of scene understanding and must be addressed explicitly. Our goal is to recover the boundaries and depth ordering of prominent objects in sufficient detail to provide an accurate sense of depth. Our greatest challenge is that objects are typically not defined by homogeneity in appearance, but by physical connectedness. For example, in Figure 4.1, the most prominent objects are the



Input Image Occlusion Labels Depth Range

Figure 4.1: Given an image, we recover occlusion boundaries and infer a range of possible depths that are consistent with the occlusion relationships. In the center, blue lines denote occlusion boundary estimates, arrows indicate which region (left) is in front, and black hatch marks show where an object is thought to contact the ground. On the right, we display the minimum and maximum depth estimates (red = close, blue = far).

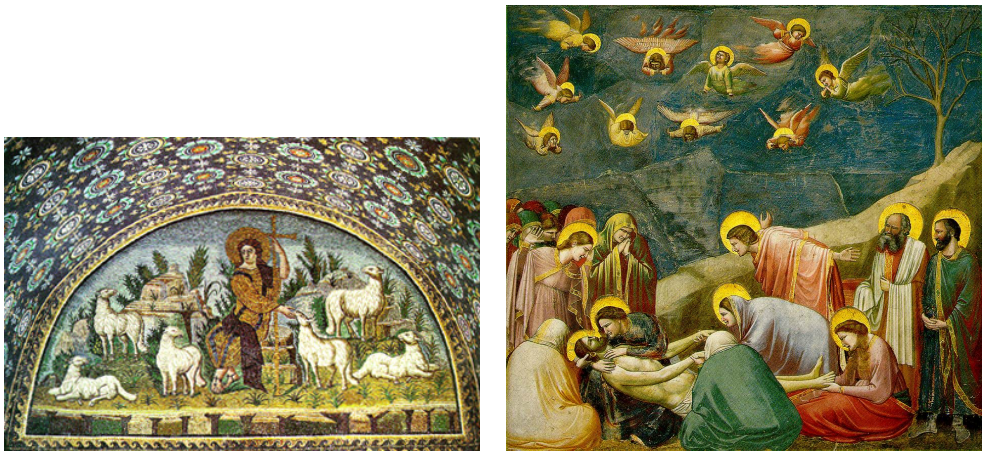


Figure 4.2: In the Byzantine era, artists took care to separate objects, such as the lambs on the left ("Christ as the Good Shepherd", 425, entrance wall of the mausoleum of Galla Placidia). Not until the start of the renaissance was occlusion in art fully appreciated (right: Giotto's "Mourning of Christ", 1305).

jungle gym, the boy, and the vegetation. Of these three, only the vegetation can be identified as a single region based on local appearance. How do we have any hope of realizing that the black shorts, white shirt, and small circular region above the shirt actually form a single object? What is coherent about the blue slides and green portal of the jungle gym?

We believe that the perception of these structures as single objects arises from a physical 3D interpretation of the scene. Correspondingly, we consider a scene to consist of a ground plane, a set of free-standing structures (objects), and the sky. In our example, the entire boy is an object because his whole body is connected to the ground through his legs. Our goal

is to determine which parts of the image correspond to ground, objects, and sky and to find the boundaries and a depth ordering of the objects. Then, with estimates of visible object-ground contact points, we can recover a *depth range*, up to a scale. Our approach is to learn models of occlusion based on both 2D perceptual cues and 3D surface and depth cues from a training set. We can then use those learned models to gradually infer the occlusion relationships, reasoning together about boundaries in the image and surfaces in the scene.

While the task we have set for ourselves is clearly very difficult, we believe that it is necessary to make progress in scene understanding. Indeed, the importance of occlusion boundaries for human scene perception has long been established by psychologists. Gibson argues that occlusion boundaries, together with surfaces, is what gives rise to the perception of the overall surface layout of a scene [**32**]. Biederman includes occlusion (which he terms *interposition*) as one of the five relational rules for a well-formed scene [**7**]. Good progress has been made in operationalizing several of Biederman's rules, including *likelihood* [**138**], *support* (Chapter 3), *size* (Chapter 5), and *position* [**133**]. Occlusion reasoning is the natural next step.

1. Background

Early computer vision successes in image understanding, such as Roberts' blocks world [**114**], encouraged interest in occlusion reasoning as a key component for a complete scene analysis system. Some success has been achieved by exploiting multiple images of the scene and apparent motion at the boundaries. This can be done based on the local surface near the boundary [**145**], the difference of the motion estimates on either sides of the boundary [**8, 127**], or the responses of spatio-temporal filters [**129, 130**]. Although these are sensible ways to exploit motion cues, we are interested here in the more difficult problem of occlusion reasoning from a single image.[1]

Traditionally, researchers have divided single-view occlusion reasoning into subtasks of segmentation and line labeling to be solved separately. Modern segmentation algorithms attempt to partition the image according to color or texture similarity or gestalt cues, but the resulting boundaries often do not to correspond to complete objects. Figure/ground labeling

[1] In collaboration with Andrew Stein, two occlusion reasoning methods were developed: the single-view work described here and a second method that focuses on motion cues [**131**].

algorithms work well, but only when given perfect segmentations. Our method is the first to both find and label occlusion boundaries in natural scenes from one image.

1.1. Segmentation

In part, our goal is similar to that of image segmentation — to partition the visual field into meaningful, coherent regions. The major difference is in how we define what makes a coherent region.

Some segmentation methods rely on 2D brightness, color, or texture cues to group the image pixels into perceptually similar regions [**124, 89, 113, 28, 3**]. Though this grouping is sometimes performed based on pre-computed affinities (e.g., [**124**]), others advocate a gradual approach, such as the hierarchical segmentation techniques developed by Ahuja [**1**] and Arbelaez [**3**], which we follow in our approach.

Other segmentation methods are based on the observation that many objects are delineated by closed, smooth contours [**70**]. These approaches typically compute affinities between edge fragments to form a graph from which contours are computed. The affinities are based on computational realizations of the Gestalt principles of continuation, proximity and closure [**149, 83, 26, 44**]. The graph can be used directly through graph-based search techniques [**26, 59, 2, 62**] to find the contours. Alternatively, a global solution can be reached by finding a dominant component in the graph with spectral methods [**128, 105, 77, 84**].

Because all of these algorithms rely on 2D perceptual grouping cues, the boundaries of such segmentations could be due to reflectance, illumination or material discontinuities, as well as occlusions, and resulting regions often do not correspond to actual objects (see Berkeley Segmentation Dataset (BSDS) [**89**]). Our physical definition of boundaries provides a more concrete objective and allows us to build on the 3D surface estimation described in Chapter 3.

1.2. Figure/Ground Line Labeling

Much research has gone into assigning occlusion labels to boundaries, once a good segmentation has been achieved. In the domain of simple polyhedral objects, Guzman in

1968 proposed an elegant algorithm for assigning occlusion labels to line segments [**37**]. The approach, fully developed by Waltz [**147**] and others [**15, 57, 58, 63, 24**], defines a set of possible line labels (convex, concave, and occluding) and a set of allowed vertex types (T-junctions, L-junctions, etc). Constraint propagation was used to efficiently rule out globally-inconsistent geometric interpretations.

This line labeling paradigm has been very influential over the years, with extensions to handle curved objects (e.g., [**60, 85**]) as well as algebraic [**134, 135**] and MRF-based [**117**] reformulations. Marill [**86**] observes that optimization of a global numerical criterion can mimic human 3D interpretation of line drawings, motivating others to formulate the 3D interpretation problem as optimization over an objective function that favors planarity, symmetry, and other "natural" properties [**76, 82, 125**]. More recent approaches also incorporate topological constraints [**11**].

However, attempts to transfer these ideas from the world of line drawings to natural images have been largely unsuccessful. The main reason is that detecting boundaries in real images is in itself an extremely hard problem. The intuition that directly detecting junctions (especially T-junctions) should help with boundary localization and labeling has not been shown to work in practice. In fact, recent psychophysics experiments [**90**] suggest that T-junctions may not be the cause of occlusion percepts, but rather their byproduct.

To our knowledge, the only successful approach for labeling occlusion boundaries in natural images is a recent paper by Ren et al. [**112**]. Following a similar strategy to the 2.1D work of Nitzberg and Mumford [**97**], they approach the problem in two stages: 1) segment the image, 2) assign a figure/ground label to each boundary fragment according to local image evidence and global MRF-learned constraints. Given a perfect segmentation, their method is able to produce impressive results on difficult natural images. But without perfect segmentation, the performance drops dramatically, showing that the main difficulty is in finding occlusion boundaries rather than in labeling them.

1.3. Single-View 3D Reconstruction

Our goal of recovering depth is similar to recently proposed methods for single-view 3D reconstruction. Our surface estimates (Chapter 3) can sometimes be used to reconstruct a coarse 3D model of a scene (see Chapter 7 for examples). Saxena et al. [**118**] train with

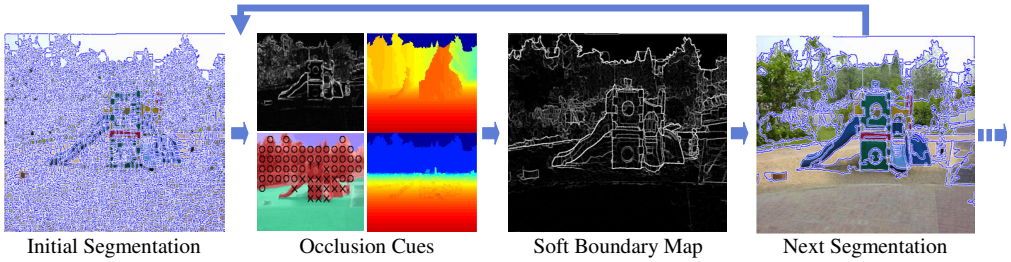


Figure 4.3: Illustration of our occlusion algorithm. Beginning with an initial oversegmentation into thousands of regions, we gradually progress towards our final solution, iteratively computing cues over boundaries and regions in the current segmentation, estimating a soft boundary map by performing inference over our CRF model, and using the boundary map to create a new segmentation. At the end of this process, we achieve the result shown in Figure 4.1.

range images to estimate depth from the image features directly. These methods, however, are likely to oversimplify the 3D model when the scene contains many foreground objects. By explicitly reasoning about occlusions, we enable much more accurate and detailed 3D models of cluttered scenes.

2. Algorithm Overview

Our strategy is to *simultaneously reason about the regions and boundaries in the image and the 3D surfaces of the scene* using learned models. Our representation includes a wide variety of cues: color, position, and alignment of regions; strength and length of boundaries; 3D surface orientation estimates; and depth estimates. In a conditional random field (CRF) model, we also encode gestalt cues, such as continuity and closure, and enforce consistency between our surface and boundary labels.

To provide an initial conservative hypothesis of the occlusion boundaries, we apply the watershed segmentation algorithm to the soft boundary map provided by the pB algorithm of Martin et al. [**88**] (skipping the non-maxima suppression step, as suggested by Arbelaez [**3**]). This produces a segmentation into thousands of regions that preserves nearly all true boundaries. In training, we assign ground truth to this initial hypothesis. Given a new image, our task is to group the small initial regions into objects and assign figure/ground labels to the remaining boundaries.

To get a final solution, we could simply compute cues over each region and boundary and perform a single segmentation and labeling step. However, the small regions from the initial

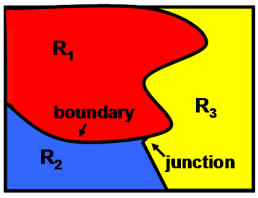


Figure 4.4: Illustration of regions, boundaries, and junctions. Beginning with an oversegmentation, we attempt to determine whether each boundary is an occlusion boundary and, if so, which region is in front. We classify the boundary based on 2D and 3D cues computed over the boundary and neighboring regions. In a conditional random field (CRF) model, we encourage continuity and enforce closure by defining appropriate energy terms over the junctions.

oversegmentation do not allow the more complicated cues, such as depth, to be reliable. Furthermore, global reasoning with these initial boundaries is ineffective because most of them are spurious texture edges.

Our solution is to gradually evolve our segmentation by iteratively computing cues over the current segmentation and using them with our learned models to merge regions that are likely to be part of the same object. In each iteration, the growing regions provide better spatial support for complex cues and global reasoning, improving our ability to determine whether remaining boundaries are likely to be caused by occlusions. See Figure 4.3 for an illustration. Each iteration consists of three steps based on the image and the current segmentation: 1) compute cues; 2) assign confidences to boundaries and regions; and 3) remove weak boundaries, forming larger regions for the next segmentation. We describe each of these steps in the following sections.

3. Cues for Occlusion Reasoning

Traditional approaches to segmentation are based on region color and texture cues or boundary cues, such as gradient strength. These 2D cues are helpful for occlusion reasoning as well, but we can also benefit from 3D cues of surface orientations and depth because our segmentations are defined according to physical boundaries. Using all of these cues, we learn to detect whether a boundary is likely to be caused by an occlusion and, if so, which side is likely to be in front. Using these learned models, we can then infer occlusion boundaries in a new image. Our occlusion cues are listed in Table 4.1 and described below. See Figure 4.4 for an illustration of regions, boundaries, and junctions.

Occlusion Cue Descriptions	Num
Region	**18**
R1. Color: distance in L*a*b* space	1
R2. Color: difference of L*a*b* histogram entropy	1
R3. Area: area of region on each side	2
R4. Position: differences of bounding box coordinates	10
R5. Alignment: extent overlap (x,y, overall,at boundary)	4
Boundary	**7**
B1. Strength: average *Pb* value	1
B2. Length: length / (perimeter of smaller side)	1
B3. Smoothness: length / (L1 endpoint distance)	1
B4. Orientation: directed orientation	1
B5. Continuity: minimum diff angle at each junction	2
B6. Long-Range: number of chained boundaries	1
3D Cues	**34**
S1. GeomContext: average confidence, each side	10
S2. GeomContext: difference of S1 between sides	5
S3. GeomContext: sum absolute S2	1
S4. GeomContext: most likely main class, both sides	1
S5. GeomTJuncts: encode event of vert-gnd-vert, vert-sky-vert	4
S6. GeomTJuncts: if S8 is true for both boundary endings	2
S7. Depth: three estimates, each side	6
S8. Depth: discontinuity along boundary	3
S9. Depth: minimum depth discontinuity (un/signed)	2

Table 4.1: Cues for occlusion labeling. The "Num" column gives the number of variables in each set. We determine which side of a boundary is likely to occlude (neither, left, right) based on estimates of 3D surfaces, properties of the boundary, and properties of the regions on either side of the boundary. Some information (such as S1) is represented several ways to facilitate learning and classification.

3.1. Region Cues

Adjacent regions are more likely to be separate objects with an occlusion boundary between them if they have different colors or textures or are misaligned. Further, as lower regions tend be closer, image position is a valuable cue for figure/ground labeling. We represent color in L*a*b* space, and we use as cues the difference of mean color (R1 in Table 4.1) and the difference between the entropy of histograms (8x8x8 bins) of the individual regions versus the regions combined (R2). We also represent the area (R3), position and differences of the bounding box and center coordinates (R4), and the alignment of the regions (R5), measured by overlap of minimum to maximum position along each axis.

3.2. Boundary Cues

Long, smooth boundaries with strong color or texture gradients are more likely to be occlusion boundaries than short boundaries with weak gradients. To represent boundary strength (B1), we take the mean Pb [**88**] (probability of boundary) value along the boundary pixels, without applying non-maxima suppression. We also provide a measure of surroundedness (B2): the ratio of boundary length to the perimeter of the smaller region (e.g., $\frac{\text{length}(e_{12})}{\text{length}(e_{12})+\text{length}(e_{23})+\text{length}(e_{24})}$ in Figure 4.7). We measure smoothness (B3) as the ratio of boundary length to Euclidean distance between endpoints and orientation (B4) as the difference between the boundary angle (simply the angle between the endpoints) and the angle of adjacent boundaries (B5). Finally, we apply a simple chaining algorithm to chain approximately (within 45 degrees) continuous boundaries together (B6).

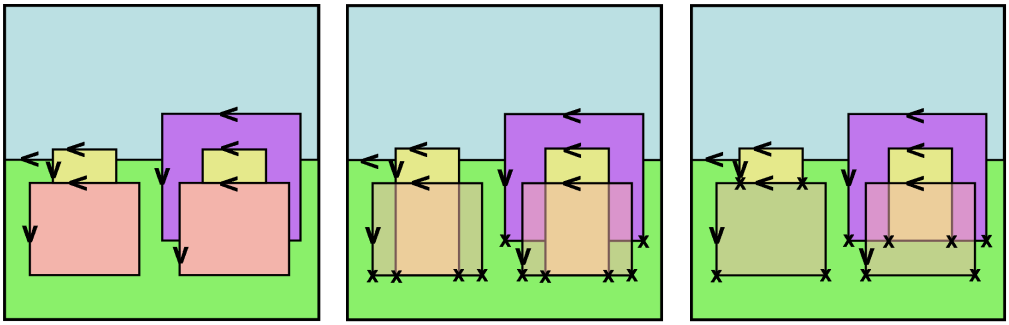
3.3. 3D Surface Cues

Many occlusion boundaries can be found by determining where two adjacent regions have different 3D surface characteristics. For instance, a woman standing in front of a building is a solid, non-planar surface occluding a planar horizontal surface (the ground) and a planar vertical surface (the building wall). We can take advantage of our work in Chapter 3 to recover surface information, which we represent as the average confidence (S1-S4) for each geometric class (horizontal support, vertical planar, vertical solid non-planar, vertical porous, and sky) over each region.

T-junctions, which occur when one boundary ends on another boundary, have long been used as evidence for an occlusion event [**37**]. Such junctions, however, are only reliable indicators when they occur at the boundaries of surfaces, not within them. As a cue, we record the event of *geometric T-junctions* (S5-S6) by finding where a boundary chain (B6) transitions from a ground-vertical or sky-vertical to a vertical-vertical boundary, according to the most likely surface labels (S4).

3.4. 3D Depth Cues

If we can determine that there is a large depth discontinuity between adjacent regions, the boundary is likely to be an occlusion boundary. Though we cannot calculate the absolute



(a) Image (b) Minimum Depth (c) Maximum Depth

Figure 4.5: Illustration of minimum and maximum depth estimates. In (a), we have a sky, support, and five vertical regions, with the occlusion relationships labeled with arrows (left side is in front). In (b), we show the interpretation for minimum depth (the frontmost region is made semi-transparent), with "X"s to indicate the ground-vertical contact points from which depth is computed. In (c), the interpretation for maximum depth is shown. The region on the left side has a minimum depth of the region occluding it and a maximum depth as if the bottom visible portion is contacting the ground. Note that the center region on the right side has a minimum depth of the region occluding it and a maximum depth of the region that it occludes. The other regions have visible ground contact points and, thus, have equal minimum and maximum depths. All depth is estimated up to a scale.

depth of a region without knowing the camera parameters, we can estimate the relative depth between two regions if we can see where each region contacts the ground. When the ground contact point of a region is occluded, we can estimate a possible depth range for that region (see Figure 4.5).

More formally, under assumptions of no camera roll, unit aspect ratio, zero skew, and an approximately level camera, the depth of a point at ground level at pixel row v_i is given by $z = \frac{fy_c}{v_i - v_0}$, where f is the camera focal length, y_c is the camera height, and v_0 is the horizon position. Therefore, given the horizon, the ground-vertical-sky surface labels, and ground-vertical contact points, we can approximate the depth of an image region, up to the scale fy_c. The depth difference of two such ground points is $\log(z_2) - \log(z_1) = -\log(v_2 - v_0) + \log(v_1 - v_0)$ in log space.

The surface labels are given by the most likely labels, according to the surface estimates described in the previous subsection. We estimate the horizon to be below the lowest sky label, above the highest ground pixel, and as close to the image center as possible. We estimate the ground-vertical contact points using a decision tree classifier based on the shape of the perimeter of a region, as described by Lalonde et al. [**75**]. For each region,

we provide three estimates of depth (S7-S9) corresponding to three guesses of the ground-contact point. The first is estimated by computing the closest ground pixel directly below the object, giving a trivial underestimate of depth. The second assigns the depth of objects without visible ground-contact points as the maximum depth of the objects that occlude it, giving a more plausible underestimate of depth. The third assigns such objects the minimum depth of objects that it occludes, giving an overestimate of depth. The depth range images displayed in our results depict the second and third of these estimates.

4. CRF Model for Occlusion Reasoning

Once we have computed cues over the boundaries and regions from the current segmentation, the next step is to estimate the likelihoods of the boundary labels (denoted 0 for no boundary or the region number of the occluding side) and of the surface labels (into ground, planar, porous, solid, and sky). Our CRF model allows joint inference over both boundary and surface labels, modeling boundary strength and continuity and enforcing closure and surface/boundary consistency.

We represent the model with a factor graph, in which the probability of the boundaries and surfaces is given by

$$\mathrm{P}(\mathbf{labels}|\mathrm{data}) = \frac{1}{\mathrm{Z}} \prod_{\mathrm{j}}^{\mathrm{N_j}} \phi_{\mathrm{j}} \prod_{\mathrm{e}}^{\mathrm{N_e}} \gamma_{\mathrm{e}} \tag{4.1}$$

where in shorthand notation we denote junction factor ϕ_j and surface factor γ_e, with N_j junctions and N_e boundaries in the graph and partition function Z.

The junction factors encode the likelihood of the label of each boundary according to the data, conditioned on its preceding boundary if there is one. They also enforce closure and continuity. Though there are 27 possible labelings of boundary triplets, there are only five valid types of three-junctions, up to a permutation. We give the terms for these five types in Figure 4.6. Four-junctions are handled in a similar manner. A prohibitively high penalty is set for invalid junctions, such as one edge leading into the junction and none leading out. In Figure 4.7, we give examples of the junction factor terms. Note that the factor graph, when given a valid labeling, decomposes into one likelihood term per boundary. This nice property allows us to learn boundary likelihoods using standard machine learning techniques, such as boosted decision trees, without worrying about CRF interactions (see Section 6 for

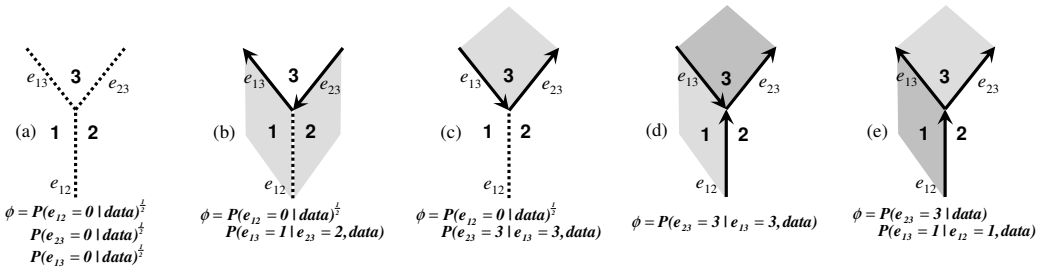


Figure 4.6: The five types of valid junctions with the expression for their corresponding potential. By convention the foreground region (shaded) is to the left of the directed edge. Dotted lines indicate non-occlusion boundaries.

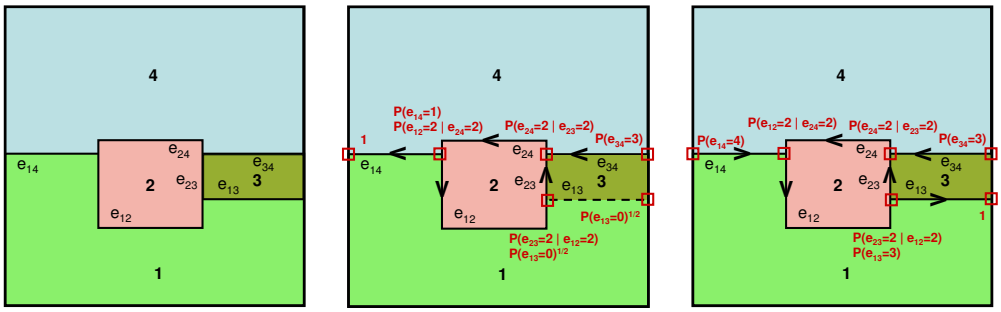


Figure 4.7: Illustration of segmentation and two possible sets of boundary labels (center and right) with junction CRF factors. Left side of arrows indicates foreground, and dashed lines indicate non-occlusion boundaries. In the CRF model, each junction has a factor term, which is shown for the given labeling. To improve clarity, we omit the data term in the likelihood expressions here.

implementation details). The reasoning over junctions and the definition of valid junctions is reminiscent of the much earlier line labeling work of Waltz [**147**]. Where Waltz used shading to resolve ambiguities in polyhedral scenes, we learn region and boundary cues to label occlusions in general scenes.

The surface factors encode the likelihood of the surface label of each region according to the data and enforce consistency between the surface labels and the boundary labels. We impose a strong penalty ($\rho_{inconsistent} = e^{-1}$) for the lack of a boundary between different geometric classes, for the ground region or sky occluding a vertical region, and for sky occluding the ground. We impose a weaker penalty ($\rho_{floating} = e^{-0.25}$) for a vertical region being entirely surrounded by another vertical region or by sky (which would imply that the former is floating). Examples of the surface factors are shown in Figure 4.8. The unary region likelihood $\mathrm{P}(\mathrm{r_i}|\mathbf{data})$ is computed as the mean geometric class confidence over the

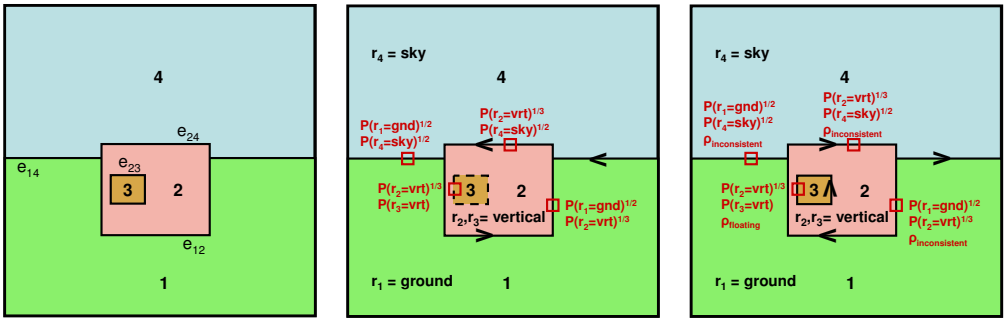


Figure 4.8: Illustration of segmentation and two possible sets of boundary labels (center and right) with surface CRF factors. Left side of arrows indicates foreground, and dashed lines indicate non-occlusion boundaries. In the CRF model, each boundary has a factor term, which is shown for the given labeling. To improve clarity, we omit the data term in the likelihood expressions here.

region (S1 in Table 4.1). To write one surface factor term per boundary, the unary region terms are set to $\mathrm{P}(r_i|\mathbf{data})^{\frac{1}{n_i}}$, where n_i is the number of boundaries surrounding region r_i.

Even approximate max-product inference over our model is intractable due to the high closure penalties, but the Heskes et al. [**45**] Kikuchi free energy-based sum-product algorithm, combined with the mean field approximation of raising factors to the $1/T$ ($T = 0.5$ in our experiments), as suggested by Yuille [**153**] efficiently provides "soft-max" likelihood estimates.

5. Segmentation from Boundary Likelihoods

Given a soft boundary map, we can compute a hierarchical segmentation and threshold it to get the initial segmentation for the next iteration (see Figure 4.9). The hierarchical segmentation is computed by iteratively merging regions with the minimum boundary strength until no boundary is weaker than the given threshold. We define the boundary strength between two regions as the maximum of 1) the value of the strongest boundary between them ($1 - \mathrm{P}(e_{12} = 0|\mathbf{data})$); and 2) a re-estimate of boundary strength computed when new regions are formed. The first of these is the value from our CRF inference. The second is computed by estimating the boundary likelihood of newly formed regions based on quickly computable cues (S1-S4, C1, R1-R5 in Table 4.1). By incorporating this second estimate, we better handle cases in which two distant regions are clearly different objects but are separated by a set of weak boundaries (as in the case of a slowly varying gradient). Our definition of total boundary strength as the maximum of the two estimates ensures that

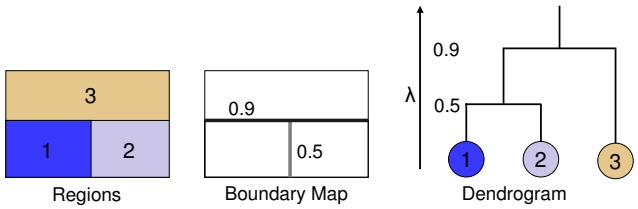


Figure 4.9: Illustration of hierarchical segmentation from soft boundary map. Here, for hierarchy threshold $\lambda > 0.5$ the two bottom regions are merged, and for $\lambda > 0.9$, all regions are merged.

our merging metric is an ultrametric [**3**], guaranteeing a true hierarchy. We threshold the hierarchy to provide our next initial segmentation.

6. Implementation Details

We train and test our method on our Geometric Context dataset (Chapter 1), consisting of a wide variety of scenes including beaches, fields, forests, hills, suburbs, and urban streets.

6.1. Assigning Ground Truth

To assign ground truth, we segment each image into thousands of regions, using watershed with pB soft boundaries, and manually group them into object regions, which could be discontinuous due to occlusion. We then label the occlusion relationships of adjacent regions. We assigned ground truth to 100 images: 50 for training and 50 for testing. For training, we use the 50 images that were originally used to train the segmentation in the surface estimation algorithm. For quantitative evaluation, we use 50 of the test images from the dataset (specifically, the images from the first fold of the five-fold cross-validation). Examples of the ground truth can be seen in Figure 4.14. A medium-complexity image will typically contain 10-15 objects according to our ground truth labels.

6.2. Training

After defining the ground truth over our dataset, we train using the algorithm outlined in Figure 4.10. We estimate the unary ($\mathrm{P}(e_1|\mathrm{data})$) and conditional ($\mathrm{P}(e_1|e_2,\mathrm{data})$) boundary classifiers using a logistic regression version of Adaboost [**16**], with 20 16-node decision trees as weak learners. This classification method provides good feature selection and probabilistic outputs. The cues used in the unary classifier are described in Section 3. For

TRAINING

Input:
- Training images
- Initial segmentations $\{\mathbf{s}^0\}$ (from watershed/pB)
- Ground truth object regions $\{\hat{\mathbf{s}}^0\}$
- Ground truth boundary labels $\{\hat{\mathbf{e}}^0\}$ (NoEdge/Side1Occludes/Side2Occludes)

For iteration t = 1..3:
1. For each image k: compute cues for segmentation ${\mathbf{s}_k}^{t-1}$ (Section 3)
2. Train boosted decision tree boundary classifiers to get $\mathrm{P}^t(e_i|\mathbf{data})$, $\mathrm{P}^t(e_i|e_j = \mathrm{on}, \mathbf{data})$
3. For each image: compute soft boundary map by CRF inference (Section 4)
4. For each image: compute hierarchical segmentation (Section 5)
5. Set hierarchy threshold by training error
6. For each image k: get next segmentation ${\mathbf{s}_k}^t$ by thresholding hierarchy
7. Update ground truth $\{\hat{\mathbf{s}}^t\}$, $\{\hat{\mathbf{e}}^t\}$ as best fit from $\{\hat{\mathbf{s}}^0\}$, $\{\hat{\mathbf{e}}^0\}$ given $\{\mathbf{s}^t\}$

Output:
- Boundary classifiers for each iteration
- Thresholds for hierarchical segmentations for each iteration

Figure 4.10: Procedure for training our occlusion recovery algorithm.

pairwise cues, we simply concatenate the unary cues for both boundaries and add cues for continuity (relative angle of the two adjacent boundaries) and boundary length (in pixels). Since cues such as depth and color histograms become more useful in the later iterations of our algorithm (with larger regions), we train separate classifiers for the initial segmentations (about 4,400 regions per image, on average) and for the segmentations obtained after the first and second iterations (300 and 100 regions per image, on average, respectively).

In the first two iterations, we set the threshold for the hierarchical segmentation to a conservative value corresponding to an "acceptable" level of pixel error in the training set (1.5%, 2%, respectively), as is typically done in cascade algorithms such as Viola and Jones object detection [**146**]. The threshold values are 0.105 and 0.25. For instance, in the first iteration, we merge two regions if we are less than 10.5% confident that there is an occlusion boundary between them. The threshold for the remaining iterations can be set to reflect the desired trade-off between the number of regions and how well the true object regions are preserved. In our experiments, we set this threshold to 0.6. To train the boundary classifiers after the first iteration, we transfer the ground truth from the initial oversegmentations to

the current segmentations by automatically labeling each region as the object that occupies the largest percentage of its pixels.

In our experiments, we set the surface factor unary term by combining, in a linear logistic model, two likelihood estimates: 1) the multiple segmentation estimate from Chapter 3; and 2) an estimate using the same cues as (1) but using the current segmentation from our occlusion algorithm. The logistic weights (1.34, 0.16) were learned to maximize likelihood of the training surface labels.

6.3. Inference

To evaluate a new image, we perform the algorithm described in the previous sections, initializing with an oversegmentation, and iteratively progressing toward our final solution. In each iteration, we compute cues over the current regions, compute boundary likelihoods in a CRF model based on those cues, and create a new segmentation by merging regions based on the boundary likelihoods. In the first iteration, we restrict our CRF model to the unary boundary likelihoods, since boundary and surface reasoning over the initial segmentation is ineffective and computationally expensive. In the second iteration, we expand our model to include the full junction factor terms. In each additional iteration, we perform inference over the full model. The algorithm terminates when no regions are merged in an iteration (typically after 4 or 5 iterations, in total). In our Matlab implementation, our algorithm takes about four minutes for a 600x800 image on a 64-bit 2.6GHz Athalon running Linux, including about 70 seconds for pB [**88**] without non-maxima suppression, about 25 seconds for our surface estimation algorithm, and about 135 seconds for the occlusion algorithm.

7. Experiments

We quantitatively evaluate our method in terms of boundary classification, figure/ground classification, and overall segmentation accuracy on 50 test images. We also provide several qualitative results, showing our the recovered object boundaries and estimated depth maps.

7.1. Boundary Classification

In Figure 4.11, we show the precision-recall curve for detecting whether a boundary in the initial oversegmentation is an occlusion boundary using only Pb [**88**], after adding region

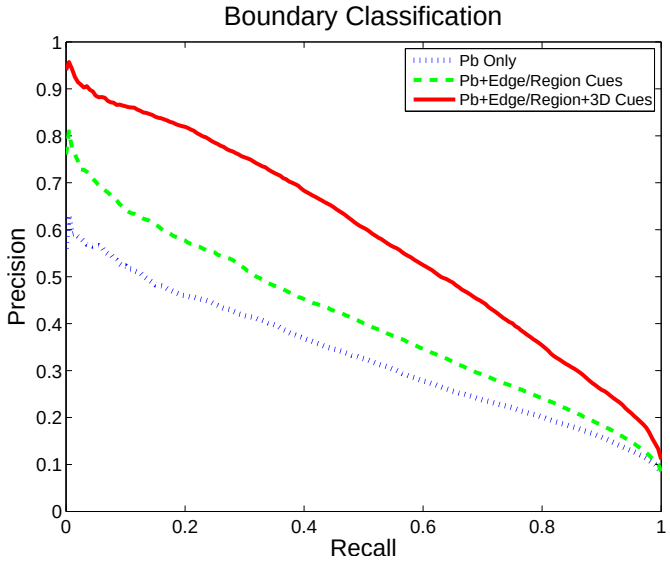


Figure 4.11: Precision-recall curve for classifying whether a boundary is an occlusion boundary in the first iteration. These results show that 3D cues are important for occlusion reasoning.

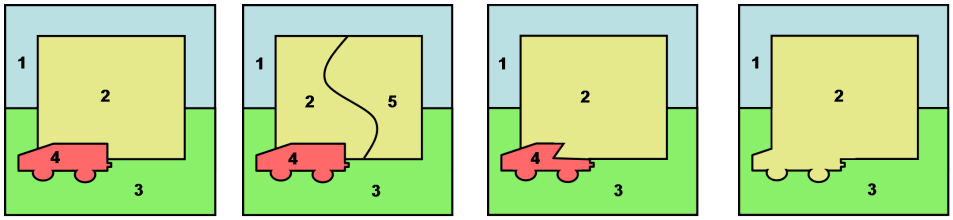
	Edge/Region Cues	+ 3D Cues	with CRF
Iter 1	58.7%	71.7%	–
Iter 2	65.4%	75.6%	77.3%
Final	68.2%	77.1%	**79.9%**

Table 4.2: Figure/ground labeling accuracy results for using edge/region cues only, all cues (including 3D cues), and after performing inference using our CRF model (only unary likelihoods were used in the first iteration).

and boundary cues, and using all cues. Our results show that the 3D cues are valuable for occlusion reasoning. In computing the precision and recall, boundaries are weighted by length in pixels. For the Pb result, the precision-recall curve is generated by ranking boundaries according to Pb confidence.

7.2. Figure/Ground Classification

In Table 4.2, we report the figure/ground classification accuracy. Accuracy is computed over all true occlusion boundaries, including those which are incorrectly classified as non-boundaries in testing. We can see that our accuracy improves in each iteration, as increasingly refined segmentations offer better spatial support for occlusion reasoning. Our final accuracy of 79.9% is remarkable, considering that on the BSDS dataset [**89**] the algorithm of Ren et al. [**112**] achieves figure/ground accuracy of 78.3% using *manual* segmentations



(a) Ground Truth (b) Less Efficient (c) Lower Conserv. (d) Lower Conserv.

Figure 4.12: Illustration of efficiency and conservation measures. The segmentation in (a) is the ground truth. In (b), since the building is split into two regions, the efficiency is reduced to $\log \frac{4}{5}$ ($\log 1 = 0$ is perfect). In (c), part of the car is assigned to the building, reducing the conservation by the area of the mistaken region (but efficiency is perfect). In (d), efficiency remains perfect ($\log \frac{3}{4-1}$), but conservation is decreased by the area of the car.

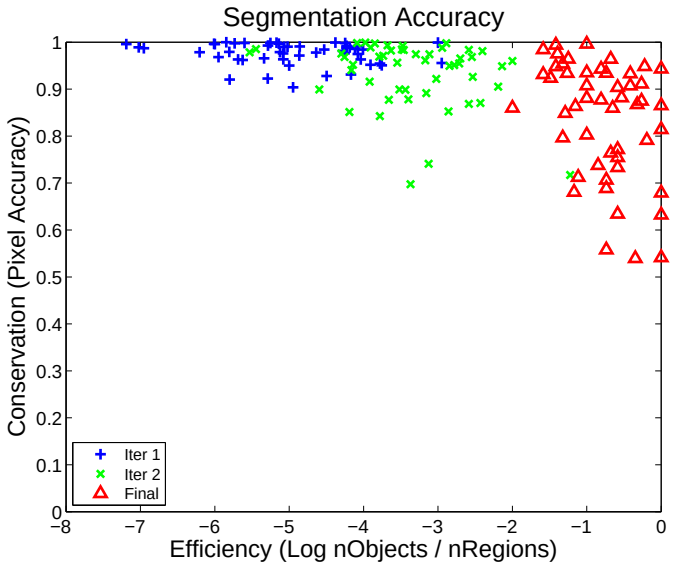
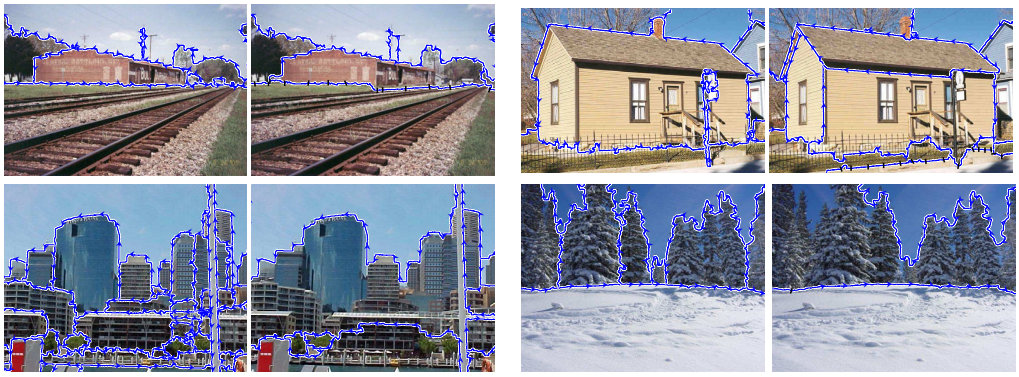


Figure 4.13: Scatter plot of "efficiency" vs. "conservation" for each test image as the segmentations become increasingly coarse. A perfect segmentation would have $\log_2$ efficiency of 0 and conservation of 1.

or 68.9% with automatically computed boundaries. Our high accuracy is due to our 3D surface and depth cues and our explicit reasoning about occlusion.

7.3. Overall Segmentation Accuracy

We measure the accuracy of a segmentation in terms of its "conservation" and its "efficiency". We report conservation as the pixel error according to the ground truth segmentation, and the efficiency as $\log_2 \frac{N_{objects} - N_{missed}}{N_{regions}}$. Here, $N_{objects}$ is the total number of objects, and N_{missed} is the number of objects that cannot be recovered from the current segmentation.



Ground Truth Result Ground Truth Result

Figure 4.14: Ground truth and final occlusion boundary results. The left side of the arrows is the occluding object. The small black lines on the result indicate estimated ground contact points. range is estimated based on constraints given by the segmentation, ground contact points, and segmentation. From top-left clockwise, the efficiency and conservation values are (-0.22, 0.95), (-1.59, 0.93), (0, 0.68), (-0.35, 0.54). The lower-left image is the result with the lowest pixel accuracy out of the test images with ground truth.

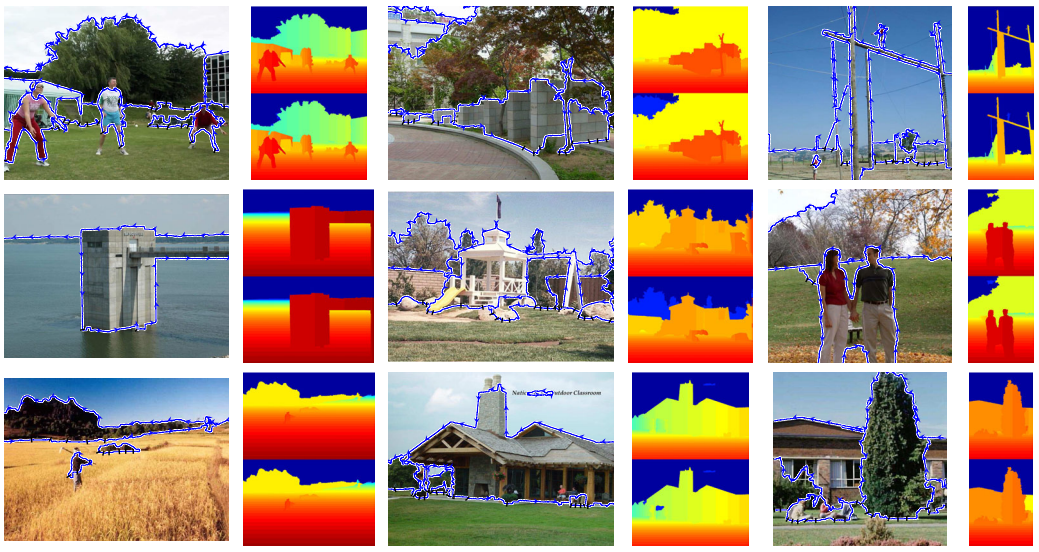


Figure 4.15: For each image, we show the recovered object boundaries with arrows indicating the foreground object (left side) and black straight lines indicating ground contact points. To the right of each image, we show an estimated depth range, with the minimum depth on top, and the maximum depth underneath. Depth ranges are computed based on the object boundaries, the surface geometry, the ground contact points, and the depth ordering.

The difference is taken so that efficiency does not increase when two ground truth objects are merged. See Figure 4.12 for an illustration. We plot the results for each stage on a scatter plot in Figure 4.13. Our results show that, as we progress, we are able to make large improvements in efficiency with moderate loss in accuracy.

	Conservation	$\log_2$ Efficiency
Our Algorithm	**83.7%**	**-0.8**
Surface-Based Segmentation	82.4%	-1.4
Ncuts Segmentation	81.7%	-1.2

Table 4.3: Comparison of our method to segmentation using only surface labels (Chapter 3) and an image-based normalized cuts segmentation algorithm [**18**]. We outperform both by using the surface cues together with additional image cues and boundary reasoning.

In Figure 4.14, we show several ground truth and result pairs, along with the conservation and efficiency scores for those segmentations. In these figures, the left side of the arrows is the occluding object, and the small black lines indicate where we believe that an object contacts the ground. In Figure 4.15, we show results for a variety of other images, together with estimated depth maps. The depth maps (red is close, blue is far) make use of surface labels (ground, vertical, sky), the segmentations, and the depth orderings.

To recover depth, our algorithm attempts to find the ground contact points for each object (see Section 3.4 for details). If it does not find a ground contact point, it constrains the object to lie behind any objects that occlude it and in front of any objects that it occludes (according to our figure/ground estimates). For each result, we show the two endpoints of that range. Our depth maps are not quantitatively accurate, since camera viewpoint and focal length are unknown, but they give a good qualitative sense of the depth of the scene.

7.4. Comparison to Baseline Algorithms

To provide further validation, we compare to two baseline methods: 1) direct segmentations from our surface labels (Chapter 3); and 2) a recent version of the Normalized Cuts image segmentation algorithm [**18**]. To implement the former, we first label the image into "ground", "planar", "porous", "solid", and "sky". We then partition the image by performing connected components on each label. Each connected component of a single label becomes one region in the segmentation. When applying the Normalized Cuts algorithm, we downsample the images to a maximum of 320 pixels per edge (due to high memory usage of the algorithm) and use the publicly available code (`http://www.seas.upenn.edu/~timothee/software_ncut/software.html`). We supply the algorithm with the number of ground truth regions, since it requires the number of segments to be specified.

In Table 4.3, we show mean conservation and efficiency for the two algorithms. In Figure 4.16, we show a representative sample of qualitative results. Overall, our algorithm achieves greater conservation and greater efficiency than both of the other algorithms. On a per-image basis, our algorithm is better in both conservation and efficiency than the purely surface-based segmentation in 48% of the images and worse in only 4% (in the remainder, our algorithm is better in only conservation or only efficiency). Also, note that only our algorithm provides foreground/background relationships.

8. Conclusions

We believe that we have made much progress on an extremely difficult problem that is crucial to scene understanding. Our method compares favorably to existing algorithms in boundary classification, figure/ground labeling, and segmentation. The recovered depth maps indicate a good sense of depth in many of the scenes. The key, we believe, is to reason together about the segmentations and figure/ground relationships, taking advantage of both 2D and inferred 3D cues in the image. Further progress can be made by including object-specific information or by extending the current color and texture similarity measures to more general measures of co-occurrence in natural scenes. Acquisition of large training sets, ideally by automatic assignment of ground truth using stereo or video cues, would allow large improvements through more effective learning.

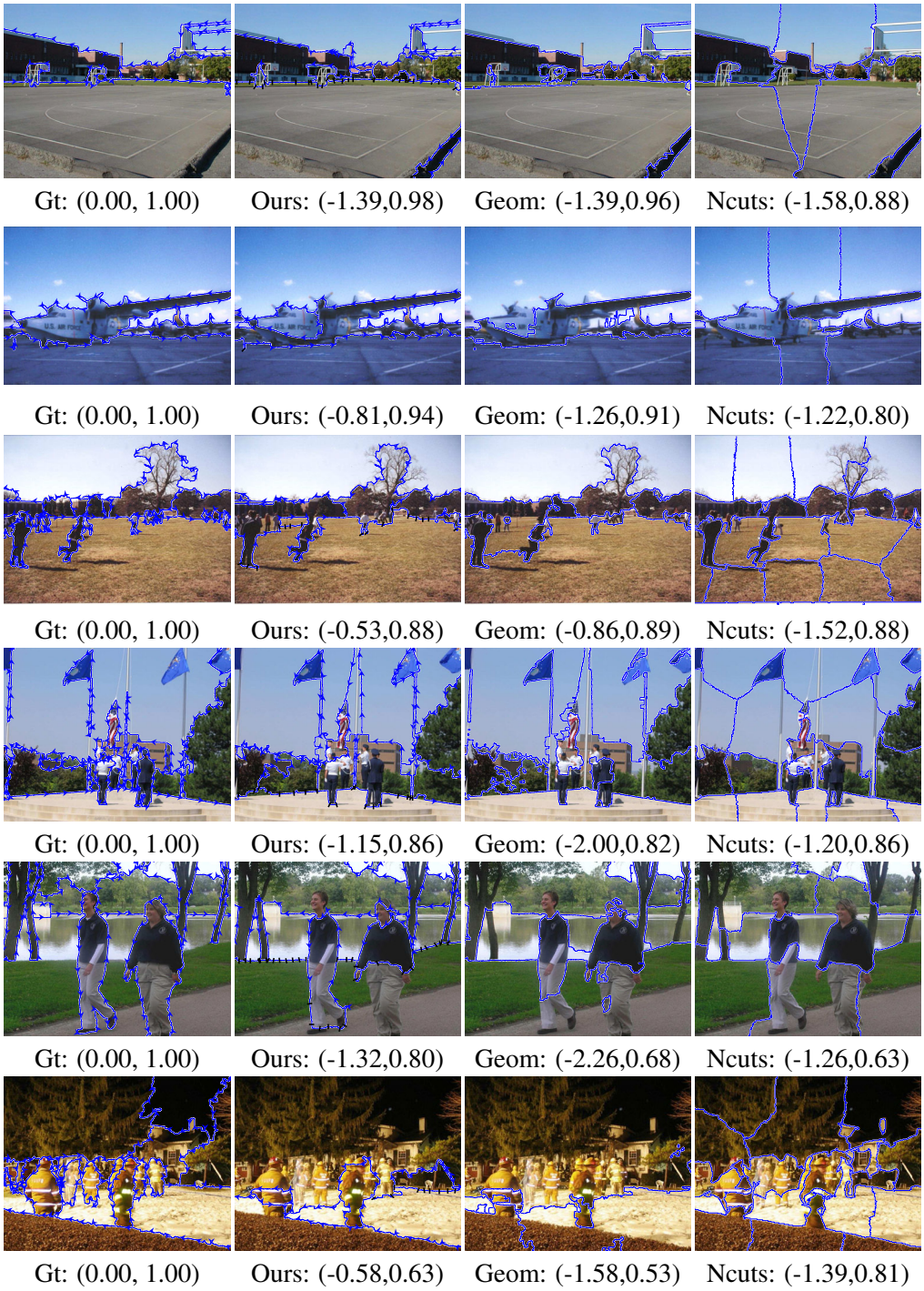


Figure 4.16: Comparison of our results with ground truth (Gt) and purely surface-based (Geom) and image-based (Ncuts) algorithms on a representative set of images. The left number shows $\log_2$ efficiency, and the right shows conservation.

CHAPTER 5

Putting Objects in Perspective

Never look down to test the ground before taking your next step; only he who keeps his eye fixed on the far horizon will find the right road.
Dag Hammarskjold (1905 - 1961)

In the previous chapters, we showed how to recover surface and occlusion information. Now, we describe how to estimate the camera viewpoint, the third component of our spatial layout. As there is a tight connection between object size and viewpoint, we reason about both of these together.

Consider the street scene depicted on Figure 5.1. Most people will have little trouble seeing that the green box in the middle contains a car. This is despite the fact that, shown in isolation, these same pixels can just as easily be interpreted as a person's shoulder, a mouse, a stack of books, a balcony, or many other things! Yet, when we look at the entire scene, all ambiguity is resolved – the car is unmistakably a car. How do we do this?

There is strong psychophysical evidence (e.g., [**7, 139**]) that context plays a crucial role in scene understanding. In our example, the car-like blob is recognized as a car because: 1) it's sitting on the road, and 2) it's the "right" size, relative to other objects in the scene (cars, buildings, pedestrians, etc). Of course, the trouble is that everything is tightly interconnected – a visual object that uses others as its context will, in turn, be used as context by these other objects. We recognize a car because it's on the road. But how do we recognize a road? – because there are cars! How does one attack this chicken-and-egg problem? What is the right framework for connecting all these pieces of the recognition puzzle in a coherent and tractable manner?

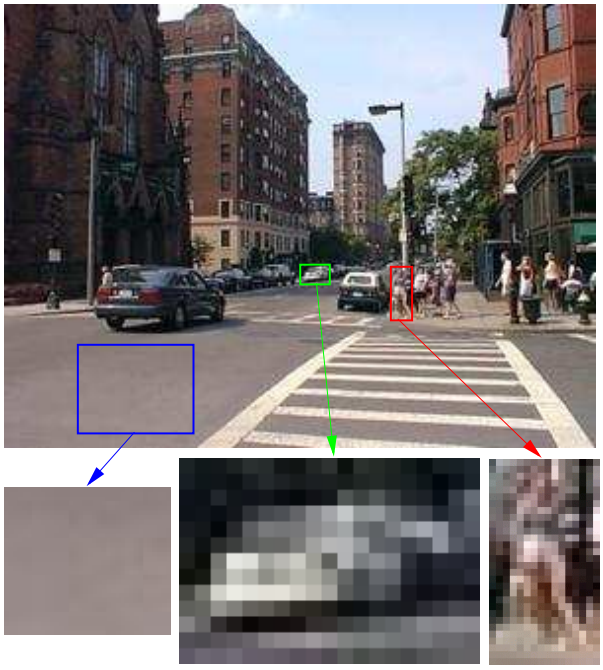
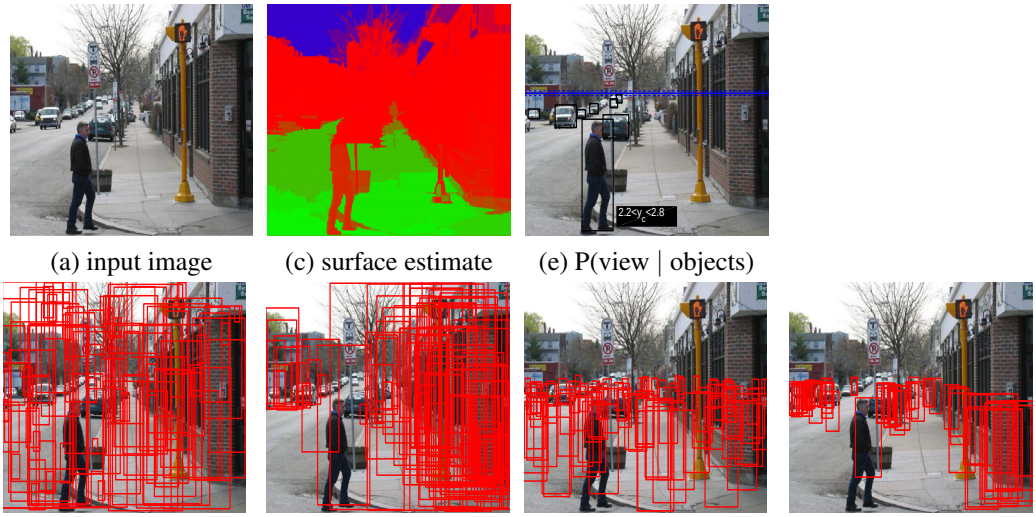


Figure 5.1: General object recognition cannot be solved locally, but requires the interpretation of the entire image. In the above image, it's virtually impossible to recognize the car, the person and the road in isolation, but taken together they form a coherent visual story.

In this chapter, we propose a unified approach for modeling the contextual symbiosis between three crucial elements required for scene understanding: low-level object detectors [**92, 21**], rough 3D scene geometry (Chapter 3), and approximate camera position/orientation. Our main insight is to model the contextual relationships between the visual elements, *not in the 2D image plane* where they have been projected by the camera, but *within the 3D world* where they actually reside. Perspective projection obscures the relationships that are present in the actual scene: a nearby car will appear much bigger than a car far away, even though in reality they are the same height. We "undo" the perspective projection and analyze the objects in the space of the 3D scene.

1. Background

By the late 1970s several image understanding systems were being developed, including such pioneering work as Brooks' *ACRONYM* [**10**], Hanson and Riseman's *VISIONS* [**40**], Ohta and Kanade's outdoor scene understanding system [**99**], and Barrow and Tenenbaum's intrinsic images [**5**]. However, the currently popular learning approaches are based on looking at small image windows at all locations and scales to find specific objects. This



(a) input image (c) surface estimate (e) P(view | objects)

(b) P(person) = uniform (d) P(person | geom) (f) P(person | view) (g) P(person | view,geom)

Figure 5.2: Watch for pedestrians! In (b,d,f,g), we show 100 boxes sampled according to the available information. Given an input image (a), a local object detector will expect to find a pedestrian at any location/scale (b). However, given an estimate of rough surface orientations (c), we can better predict where a pedestrian is likely to be (d). We can estimate the camera viewpoint (e) from a few known objects in the image. Conversely, knowing the camera viewpoint can help in predict the likely scale of a pedestrian (f). The combined evidence from surface geometry and camera viewpoint provides a powerful predictor of where a pedestrian might be (g), before we even run a pedestrian detector! Red, green, and blue channels of (c) indicate confidence in vertical, ground, and sky, respectively. Best viewed in color.

works wonderfully for face detection [**122, 146**] (since the inside of a face is much more important than the boundary) but is quite unreliable for other types of objects, such as cars and pedestrians, especially at the smaller scales.

As a result, several researchers have recently begun to consider the use of contextual information for object detection. The main focus has been on modeling direct relationships between objects and other objects [**72, 92**], regions [**43, 73, 143**] or scene categories [**92, 132**], all within the 2D image. From low-level image cues, Torralba and Oliva [**141**] get a sense of the viewpoint and mean scene depth, which provides a useful prior for object detection [**142**]. Forsyth et al. [**29**] describe a method for geometric consistency of object hypotheses in simple scenes using hard algebraic constraints. Others have also modeled the relationship between the camera parameters and objects, requiring either a well-calibrated camera [**61**], a stationary surveillance camera [**71**], or both [**35**].

2. Overview

To evaluate our approach, we have chosen a very challenging dataset of outdoor images [**116**] that contain cars and people, often partly occluded, over an extremely wide range of scales and in accidental poses (unlike, for example, the framed photographs in Corel or CalTech datasets). We will demonstrate that substantial improvement over standard low-level detectors can be obtained by reasoning about the underlying 3D scene structure.

One way to think about what we are trying to achieve is to consider the likely places in an image where an object, such as a pedestrian, could be found (Figure 5.2). Without considering the 3D structure of the scene, all image positions and scales are equally likely (Figure 5.2b) – this is what most object detectors assume. But if we can estimate the rough surface geometry in the scene, this information can be used to adjust the probability of finding a pedestrian at a given image location (Figure 5.2d). Likewise, having an estimate of the camera viewpoint (height and horizon position) supplies the likely scale of an object in the image (Figure 5.2f). Combining these two geometric cues together gives us a rather tight prior likelihood for the location and scale of a pedestrian, as in Figure 5.2g. This example is particularly interesting because this is still only a prior – we have not applied a pedestrian detector yet. Notice, as well, that the pattern of expected pedestrian detections is reminiscent of typical human eye-tracking experiments, where subjects are asked to search for a person in an image.

Of course, just as scene and camera geometry can influence object detection, so can the detected objects alter the geometry estimation. For example, if we know the locations/scales of some of the objects in the image, we can use this to better estimate the camera viewpoint parameters (see the 90% confidence bounds in Figure 5.2e). In general, our aim is to combine all these pieces of evidence into a single coherent image interpretation framework.

The rest of the chapter is devoted to exploring our two primary conjectures: 1) 3D reasoning improves object detection, even when using a single image from an uncalibrated camera, and 2) the more fully the scene is modeled (more properties, more objects), the better the estimates will be. We will first describe the mathematics of projective geometry as it relates to our problem (Section 3). We will then define the probabilistic model used for describing the relationships within the 3D scene (Section 4) and how it can be learned (Section 5). In

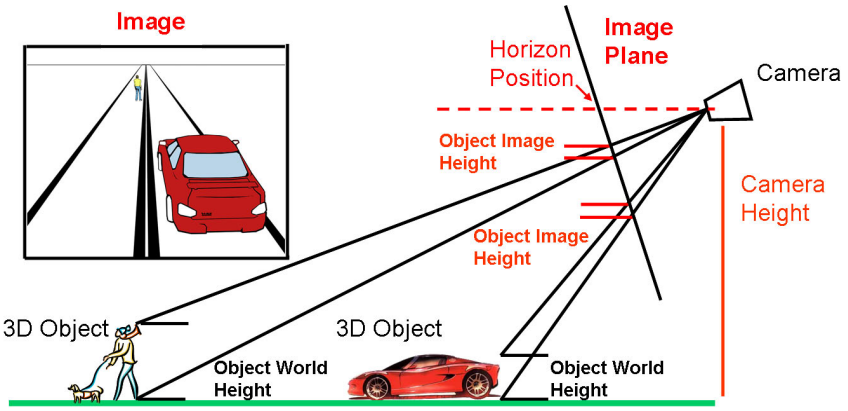


Figure 5.3: An object's height in the image can be determined from its height in the world and the viewpoint.

Section 6, we present quantitative and qualitative results demonstrating the performance of our system on a difficult dataset. Finally, in Section 7, we demonstrate a new technique for estimating camera viewpoint and show that it leads to improved accuracy in object detection.

3. Scene Projection

Under a zero-skew, unit aspect ratio perspective camera model, we can compute a grounded object's height in the scene, given only the camera height and horizon line (see Figure 5.3). First, let us rotate and translate our image coordinates (u,v) to the coordinates $(\hat{u},\hat{v})$ so that $\hat{v}=0$ for every point on the horizon and $\hat{v}>0$ for every point below the horizon. The world height y of a point can be recovered from $\hat{v}=(y_c-y)\frac{f}{z}$ where y_c is the camera height, z is the depth, and f is the camera focal length. Without loss of generality, we define the object to rest on the plane $y=0$. The object's height can be recovered from $\frac{\hat{v}_1}{\hat{v}_1-\hat{v}_2}=\frac{y_c}{y}$, where $\hat{v}_1$ is the bottom and $\hat{v}_2$ is the top of the object. To get $\hat{v}$ from pixel coordinates, we simply compute the distance of the horizon line to the point. In this dissertation, since photographs typically have little roll, we define the horizon line by the image row v_0. Letting v_i and h_i denote the bottom position and height of an object in the image (illustrated in Figure 5.4), we have the following relationship:

$$y_i=\frac{h_i y_c}{v_i-v_0}. \tag{5.1}$$

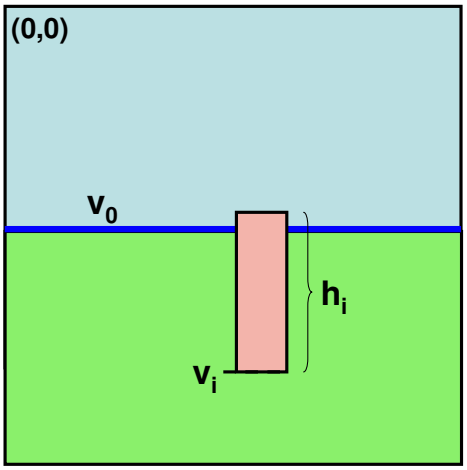


Figure 5.4: Illustration of horizon position v_0, object bottom position v_i, and object image height h_i. With these and camera height y_c, we can approximately compute object world height y_i using $y_i = \frac{h_i y_c}{v_i - v_0}$.

From equation 5.1, we can compute the image height h_i of an object given its image position v_i, 3D height y_i, and the viewpoint (v_0, y_c).[1] Of course, for an uncalibrated camera, we do not know the viewpoint or the object's true height *a priori*. However, since people do not take photos in a completely random manner and since objects have a small range of possible 3D sizes, we can estimate an informative distribution of viewpoint and object size and, from it, derive a distribution for h_i given v_i.

In this chapter, we assume that all objects of interest rest on the ground plane. While this assumption may seem restrictive (cannot find people on the rooftops), humans seem to make the same assumption (we fail to notice the security standing on the rooftops at political rallies unless we specifically look for them). If the ground is sloped, as in Figure 5.2, the coordinates and parameters are computed with respect to that slope, and the relationship between viewpoint and objects in the image still holds.

4. Modeling the Scene

We want to determine the viewpoint, object identities, and surface geometry of the scene from an image. We could estimate each independently, but our estimates will be much more accurate if we take advantage of the interactions between the scene elements. We consider

[1] This is a good approximation if the camera is close to parallel with the ground plane. More generally, y_c and v_0 could be considered a two-parameter fit to ground plane, with respect to the viewer.

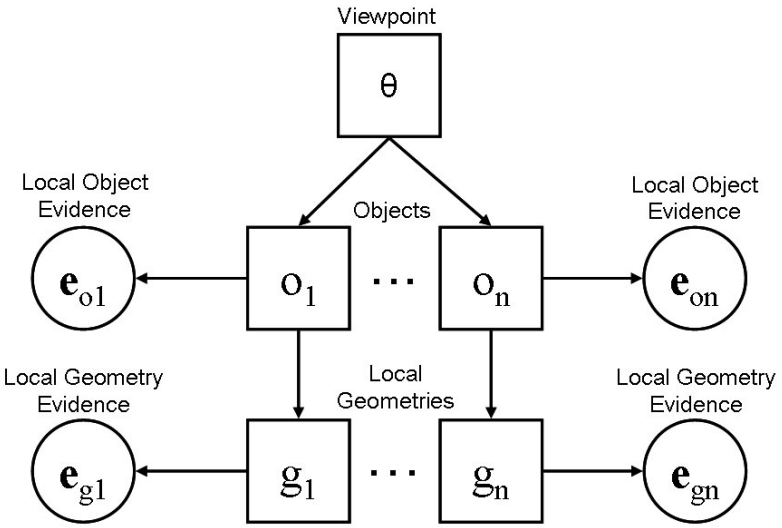


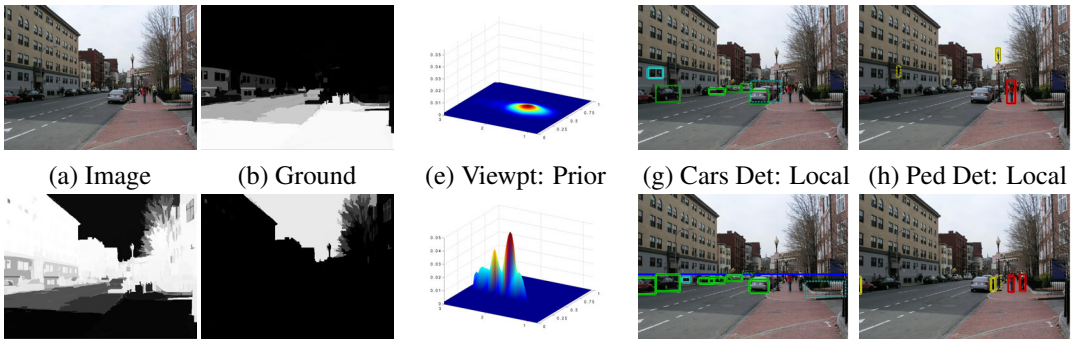
Figure 5.5: Graphical model of conditional independence for viewpoint θ, object identities $\mathbf{o}$, and the 3D geometry of surfaces $\mathbf{g}$ surrounding the objects. Viewpoint describes the horizon position in the image and the height of the camera in the 3D scene (in relation to the objects of interest). Each image has n object hypotheses, where n varies by image. The object hypothesis o_i involves assigning an identity (e.g., pedestrian or background) and a bounding box. The surface geometry g_i describes the 3D orientations of the i^{th} object surface and nearby surfaces in the scene.

the objects (e.g., cars, pedestrians, background) and geometric surfaces to each produce image evidence. The viewpoint, defined by the horizon position in the image and the camera height, directly affects the position and size of the objects in the image. In turn, the objects directly affect nearby geometric surfaces. We assume that local geometric surfaces are independent given their corresponding object identities and that the object identities are independent given the viewpoint. In Figure 5.5, we represent these conditional independence assumptions in a graphical model, denoting objects as $\mathbf{o}$, surface geometries as $\mathbf{g}$, object evidence as $\mathbf{e}_o$, geometry evidence as $\mathbf{e}_g$, and the viewpoint as θ.

Our model implies the following decomposition:

$$\mathrm{P}(\theta, \mathbf{o}, \mathbf{g}|\mathbf{e}) \propto \mathrm{P}(\theta) \prod_i \mathrm{P}(o_i|\theta) \frac{\mathrm{P}(o_i|\mathbf{e}_o)}{\mathrm{P}(o_i)} \mathrm{P}(g_i|o_i) \frac{\mathrm{P}(g_i|\mathbf{e}_g)}{\mathrm{P}(g_i)} \tag{5.2}$$

The proportionality Equation 5.2 is with respect to terms of the observed evidence ($\mathbf{e} = \{\mathbf{e}_o, \mathbf{e}_g\}$) that are constant within an image.



(a) Image (b) Ground (e) Viewpt: Prior (g) Cars Det: Local (h) Ped Det: Local

(c) Vertical (d) Sky (f) Viewpt: Full (i) Car Det: Full (j) Ped Det: Full

Figure 5.6: We begin with geometry estimates (b,c,d), local object detection confidences (g,h), and a prior (e) on the viewpoint. Using our model, we improve our estimates of the viewpoint (f) and objects (i,j). In the viewpoint plots, the left axis is camera height (meters), and the right axis is horizon position (measured from the image bottom). The viewpoint peak likelihood increases from 0.0037 *a priori* to 0.0503 after inference. At roughly the same false positive (cars:cyan, peds:yellow) rate, the true detection (cars:green, peds:red) rate doubles when the scene is coherently modeled.

Our approach allows other researchers to easily integrate their own detectors into our framework. When interactions among elements of the scene are defined, each addition to the framework adds evidence that can be used to improve estimation of the other elements.

4.1. Viewpoint

The viewpoint θ involves two variables: the horizon position in the image v_0 and the camera height (in meters) y_c. We consider camera height and horizon position to be independent *a priori* so that $\mathrm{P}(\theta) = \mathrm{P}(v_0)\mathrm{P}(y_c)$. In our initial experiments (Sections 5 and 6), we model the horizon position likelihood with a simple Gaussian prior. Similarly, for the camera height y_c, we estimate a prior distribution using kernel density estimation over the y_c values (computed based on objects of known height in the scene) in a set of training images. We will later (Section 7) show how to estimate the horizon position from image data directly, resulting in improved viewpoint estimation and object detection.

Figure 5.6 displays the viewpoint prior (e) and an example of the revised likelihood (f) when object and surface geometry evidences are considered. *A priori*, the most likely camera height is 1.67m, which is eye level for a typical adult male, and the most likely horizon

position is 0.50. While the viewpoint prior does have high variance, it is much more informative than the uniform distribution that is implicitly assumed when scale is considered irrelevant.

4.2. Objects

An object candidate o_i consists of a type $t_i \in \{object, background\}$ (e.g., "pedestrian") and a bounding box $bbox_i = \{u_i, v_i, w_i, h_i\}$ (lower-left coordinate, width, and height, respectively). The object term of our scene model is composed as follows:

$$\mathrm{P}(o_i|\mathbf{e}_o, \theta) \propto \frac{\mathrm{P}(o_i|\mathbf{e}_o)}{\mathrm{P}(o_i)}\mathrm{P}(o_i|\theta) \tag{5.3}$$

At each position and scale (with discrete steps) in the image, our window-based object detector outputs the class-conditional log-likelihood ratio

$$c_i = \log \frac{\mathrm{P}(\mathbf{I}_i|t_i = obj, bbox_i)}{\mathrm{P}(\mathbf{I}_i|t_i \neq obj, bbox_i)} \tag{5.4}$$

based on local image information $\mathbf{I}_i$ at the i^{th} bounding box. From these ratios and a prior $\mathrm{P}(o_i)$,[2] we can compute the probability of an object occurring at a particular location/scale.

$$\mathrm{P}(t_i = obj, bbox_i|\mathbf{I}_i) = \frac{1}{1 + \exp[-c_i - \log \frac{\mathrm{P}(o_i)}{1-\mathrm{P}(o_i)}]} \tag{5.5}$$

Typically, researchers perform non-maxima suppression, assuming that high detection responses at neighboring positions could be due to an object at either of those positions (but not both). Making the same assumption, we also apply non-maxima suppression, but we form a point distribution out of the non-maxima, rather than discarding them. An object candidate is formed out of a group of closely overlapping bounding boxes.[3] The candidate's likelihood $\mathrm{P}(t_i = obj|\mathbf{e}_o)$ is equal to the likelihood of the highest-confidence bounding box, and the likelihoods of the locations given the object identity $\mathrm{P}(bbox_i|t_i = obj, \mathbf{e}_o)$ are directly proportional to $\mathrm{P}(t_i = obj, bbox_i|\mathbf{I})$. After thresholding to remove detections with very low confidences from consideration, a typical image will contain several dozen object candidates (determining n of Figure 5.5), each of which has tens to hundreds of possible position/shapes.

[2] We use $\mathrm{P}(o_i)$ as shorthand for $\mathrm{P}(t_i = obj|bbox_i)\mathrm{P}(bbox_i)$, where $\mathrm{P}(bbox_i)$ is constant.

[3] Each detector distinguishes between one object type and background in our implementation. Separate candidates are created for each type of object.

An object's height depends on its position when given the viewpoint. Formally, $P(o_i|\theta) \propto p(h_i|t_i, v_i, \theta)$ (the proportionality is due to the uniformity of $P(t_i, v_i, w_i|\theta)$). From Equation 5.1, if y_i is normal, with parameters $\{\mu_i, \sigma_i\}$, then h_i conditioned on $\{t_i, v_i, \theta\}$ is also normal, with parameters $\frac{\mu_i(v_i - v_0)}{y_c}$ and $\frac{\sigma_i(v_i - v_0)}{y_c}$.

4.3. Surface Geometry

Most objects of interest can be considered as vertical surfaces supported by the ground plane. Estimates of the local surface geometry could, therefore, provide additional evidence for objects. To obtain the rough 3D surface orientations in the image, we apply the method described in Chapter 3, producing confidence maps for three main classes: "ground", "vertical", and "sky", and five subclasses of "vertical": planar, facing "left", "center", and "right", and non-planar "solid" and "porous". Figure 5.6(b,c,d) displays the confidence maps for the three main surface labels.

We define g_i to have three values corresponding to whether the object surface is visible in the detection window and, if so, whether the ground is visible just below the detection window. For example, we consider a car's geometric surface to be planar or non-planar solid and a pedestrian's surface to be non-planar solid. We can compute $P(g_i|o_i)$ and $P(g_i)$ by counting occurrences of each value of g_i in a training set. If o_i is background, we consider $P(g_i|o_i) \approx P(g_i)$. We estimate $P(g_i|\mathbf{e}_g)$ based on the confidence maps of the geometric surfaces. In experiments, we found that the average geometric confidence in a window is a well-calibrated probability for the geometric value.

4.4. Inference

Inference is well-understood for tree-structured graphs like our model (Figure 5.5). We use Pearl's belief propagation[4] algorithm [**104**] from the Bayes Net Toolbox [**91**]. Once the model is defined and its parameters estimated, as described above, it can answer queries, such as "What is the expected height of this object?" or "What are the marginal probabilities for cars?" or "What is the most probable explanation of the scene?". In this chapter,

[4]To simplify the BP algorithm, we quantize all continuous variables (v_0 and y_c into 50 and 100 evenly-spaced bins); o_i is already discrete due to sliding window detection.

we report results based on marginal probabilities from the sum-product algorithm. Figure 5.6 shows how local detections (g,h) improve when viewpoint and surface geometry are considered (i,j).

5. Training

Viewpoint. To estimate the priors for θ, we manually labeled the horizon in 60 outdoor images from the LabelMe database [**116**]. In each image, we labeled cars (including vans and trucks) and pedestrians (defined as an upright person) and computed the maximum likelihood estimate of the camera height based on the labeled horizon and the height distributions of cars and people in the world. We then estimated the prior for camera height using kernel density estimation (`ksdensity` in Matlab).

Objects. Our baseline car and pedestrian detector uses a method similar to the local detector of Murphy, Torralba, and Freeman [**92**]. We used the same local patch template features but added six color features that encode the average L*a*b* color of the detection window and the difference between the detection window and the surrounding area. The classifier uses a logistic regression version of Adaboost [**16**] to boost eight-node decision tree classifiers. For cars, we trained two views (front/back: 32x24 pixels and side: 40x16 pixels), and for pedestrians, we trained one view (16x40 pixels). Each were trained using the full PASCAL dataset [**103**].

To verify that our baseline detector has reasonable performance, we trained a car detector on the PASCAL challenge training/validation set, and evaluated the images in test set 1 using the criteria prescribed for the official competition. For the sake of comparison in this validation experiment, we did not search for cars shorter than 10% of the image height, since most of the official entries could not detect small cars. We obtain an average precision of 0.423 which is comparable to the best scores reported by the top 3 groups: 0.613, 0.489, and 0.353.

To reason together about objects and viewpoint, we need to know the 3D height distributions of objects. These distributions can be produced either from publicly available statistics, as we do here, or by learning the size distributions in a semi-supervised framework, as we will

describe in Section 7. To estimate the height distribution of cars, we used Consumer Reports (www.consumerreports.org) and, for pedestrians, used data from the National Center for Health Statistics (www.cdc.gov/nchs/). For cars, we estimated a mean of 1.59m and a standard deviation of 0.21m. For adult humans, the mean height is 1.7m with a standard deviation of 0.085m.

Surface Geometry. $P(g_i|o_i)$ was found by counting the occurrences of the values of g_i for both people and cars in the 60 training images from LabelMe. We set $P(g_i)$ to be uniform, because we found experimentally that learned values for $P(g_i)$ resulted in the system over-relying on geometry. This over-reliance may be due to our labeled images (general outdoor) being drawn from a different distribution than our test set (streets of Boston) or to the lack of a modeled direct dependence between surface geometries.

6. Evaluation

Our test set consists of 422 random outdoor images from the LabelMe dataset [**116**]. The busy city streets, sidewalks, parking lots, and roads provide realistic environments for testing car and pedestrian detectors, and the wide variety of object pose and size and the frequency of occlusions make detection extremely challenging. In the dataset, 60 images have no cars or pedestrians, 44 have only pedestrians, 94 have only cars, and 224 have both cars and pedestrians. In total, the images contain 923 cars and 720 pedestrians.

We detect cars with heights as small as 14 pixels and pedestrians as small as 36 pixels tall. To get detection confidences for each window, we reverse the process described in Section 4.2. We then determine the bounding boxes of objects in the standard way, by thresholding the confidences and performing non-maxima suppression.

Our goal in these experiments is to show that, by modeling the interactions among several aspects of the scene and inferring their likelihoods together, we can do much better than if we estimate each one individually.

Object Detection Results. Figure 5.7 plots the ROC curves for car and pedestrian detection on our test set when different subsets of the model are considered. Figure 5.8 displays

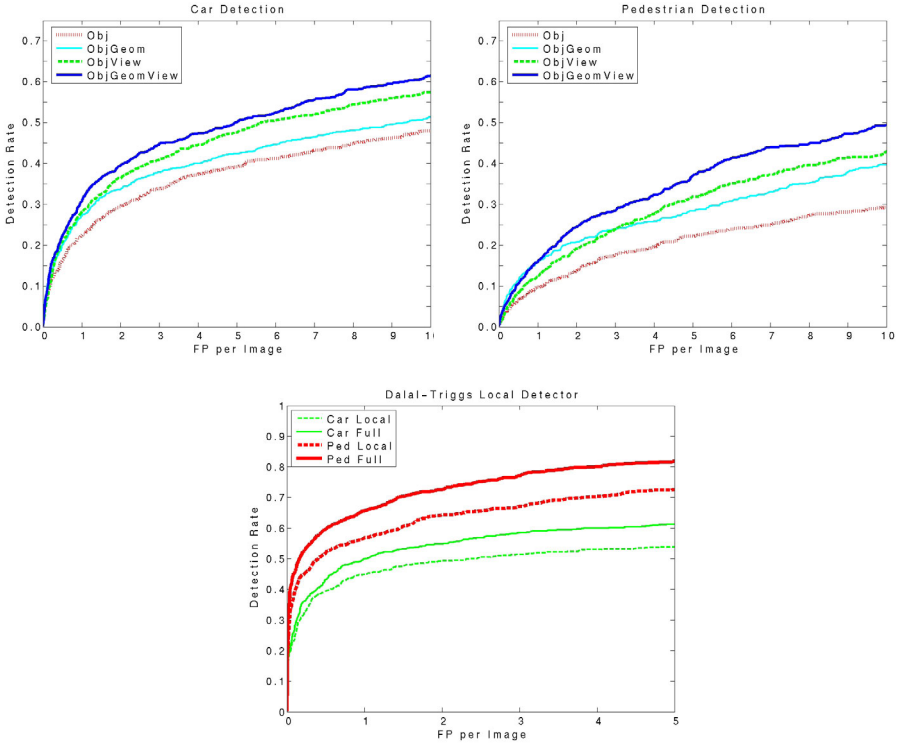


Figure 5.7: Considering viewpoint and surface geometry improves results over purely local object detection. The top two plots show object detection results using only local object evidence (Obj), object and geometry evidence (ObjGeom), objects related through the viewpoint (ObjView), and the full model (ObjViewGeom). On the bottom, we plot results using the Dalal-Triggs local detector [**21**].

	Cars			Pedestrians		
	1FP	5FP	10FP	1FP	5FP	10FP
+Geom	6.6%	5.6%	7.0%	7.5%	8.5%	17%
+View	8.2%	16%	22%	3.2%	14%	23%
+GeomView	**12%**	**22%**	**35%**	**7.2%**	**23** %	**40%**

Table 5.1: Modeling viewpoint and surface geometry aids object detection. Shown are percentage reductions in the missed detection rate while fixing the number of false positives per image.

	Mean	Median
Prior	10.0%	8.5%
+Obj	7.5%	4.5%
+ObjGeom	7.0%	3.8%

Table 5.2: Object and geometry evidence improve horizon estimation. Mean/median absolute error (as percentage of image height) are shown for horizon estimates.

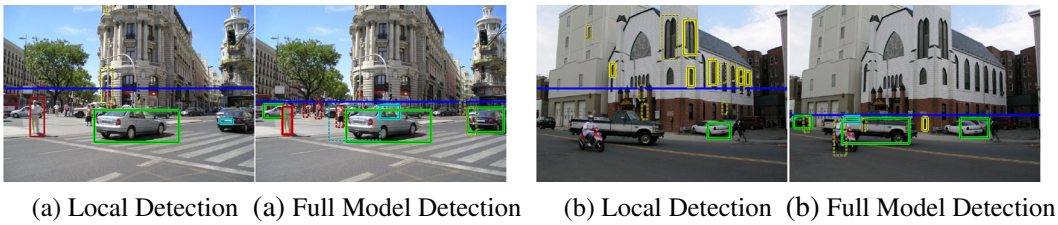
	Horizon	Cars (FP)		Ped (FP)	
Car	7.3%	5.6	7.4	—	—
Ped	5.0%	—	—	12.4	13.7
Car+Ped	3.8%	5.0	6.6	11.0	10.7

Table 5.3: Horizon estimation and object detection are more accurate when more object models are known. Results shown are using the full model in three cases: detecting only cars, only pedestrians, and both. The horizon column shows the median absolute error. For object detection we include the number of false positives per image at the 50% detection rate computed over all images (first number) and the subset of images that contain both cars and people (second number).

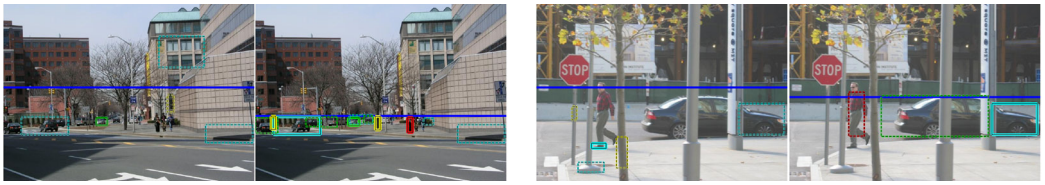
and discusses several examples. To provide an estimate of how much other detectors may improve under our framework, we report the percent reduction in false negatives for varying false positive rates in Table 5.1. When the viewpoint and surface geometry are considered, about 20% of cars and pedestrians missed by the baseline are detected for the same false positive rate! The improvement due to considering the viewpoint is amazing, since the viewpoint uses no direct image evidence. Also note that, while individual use of surface geometry estimates and the viewpoint provides improvement, using both together improves results further.

Horizon Estimation Results. By performing inference over our model, the object and geometry evidence can also be used to improve the horizon estimates. We manually labeled the horizon in 100 of our images that contained both types of objects. Table 5.2 gives the mean and median absolute error over these images. Our prior of 0.50 results in a median error of 0.085% of the image height, but when objects and surface geometry are considered, the median error reduces to 0.038%. Notice how the geometry evidence provides a substantial improvement in horizon estimation, even though it is separated from the viewpoint by two variables in our model.

More is Better. Intuitively, the more types of objects that we can identify, the better our horizon estimates will be, leading to improved object detection. We verify this experimentally, performing the inference with only car detection, only pedestrian detection, and both. Table 5.3 gives the accuracy for horizon estimation and object detection when only cars are detected, when only pedestrians are detected, and when both are detected. As predicted, detecting two objects provides better horizon estimation and object detection than detecting one.



(a) Local Detection (a) Full Model Detection (b) Local Detection (b) Full Model Detection



(c) Local Detection (c) Full Model Detection (d) Local Detection (d) Full Model Detection

(e) Local Detection (e) Full Model Detection (f) Local Detection (f) Full Model Detection

Figure 5.8: We show car and pedestrian results from our baseline local detector (from [**92**]) and after inference using our model. The blue line shows the horizon estimate (always 0.5 initially). The boxes show detection estimates (green=true car, cyan=false car, red=true ped, yellow=false ped), with the solid lines being high confidence detections (0.5 FP/Image) and the dotted lines being lower confidence detections (2 FP/Image). In most cases, the horizon line is correctly recovered, and the object detection improves considerably. In particular, boxes that make no sense from a geometric standpoint (e.g., wrong scale (d), above horizon (b), in the middle of the ground (e)) usually are removed and objects not initially detected are found. Of course, improvement is not guaranteed. Pedestrians are often hallucinated (c,e) in places where they could be (but are not). In (f), a bad geometry estimate and repeated false detections along the building windows causes the horizon estimate to become worse and the car to be missed.

Dalal-Triggs Detector. To support our claim that any local object detector can be easily improved by plugging it into our framework, we performed experiments using the Dalal-Triggs detector [**21**] after converting the SVM outputs to probabilities using the method of [**108**]. We used code, data, and parameters provided by the authors, training an 80x24 car detector and 32x96 and 16x48 (for big and small) pedestrian detectors . The Dalal-Triggs local detector is currently among the most accurate for pedestrians, but its accuracy (Figure 5.7) improves considerably with our framework, from 57% to 66% detections at 1 false positive (FP) per image. Similarly, the car detection rate improves from 45% to 50% at 1 FP per image.

Integration is Important. In our model, surface and object information is integrated through a single camera viewpoint. What if we instead simply marginalized out the viewpoint prior over each object separately, essentially providing only a position/size prior for the objects? To find out, we re-ran our experiments using the Dalal-Triggs detectors after making this change. Our fully integrated model outperforms the weaker marginalized model by 5% (cars) and 8% (pedestrians) detection rate at 1 FP per image. Similarly, we found little gain to using a less restrictive 2D prior over the object's bottom position and height that we estimated using the LabelMe dataset. Much of the power of our viewpoint and surface reasoning seems to be that it enforces detections to be *consistent*, rather than merely likely according to 2D *a priori* statistics.

7. Viewpoint by Scene Matching

One of the benefits of our proposed system is the ability to recover camera viewpoint from a single image using the detections of known objects in the scene. But what if the image does not include any easily detectable known objects? This makes the problem extremely difficult. Solutions based on edges and perspective geometry, such as methods by Kosecka and Zhang [**69**] and Coughlan and Yuille [**17**], show good results for uncluttered man-made scenes with lots of parallel lines (the so-called Manhattan worlds), but fail for less structured environments. Inspired by work in pre-attentive human vision, Oliva and Torralba [**102**] convincingly argue that simple spatial-frequency statistics (the "gist") of the image may be sufficient to recover a general sense of the space of the scene. They show some impressive results on estimating rough "scene envelope" properties such as *openness*, *smoothness* and *depth* by matching to a small number of manually labeled training images. However, using this approach to recover more precise quantities (such as camera orientation) would require 1) a much larger amount of training data, 2) a way of accurately labeling this data.

Here we propose to solve both of these issues by applying our object-viewpoint model to automatically recover camera viewpoint parameters of images in a standard object recognition database. The resulting large dataset of image/viewpoint pairs then allows us to use the gist descriptor in a simple scene matching approach to compute viewpoint estimates for a novel image. Moreover, we can use this informed viewpoint estimate in place of our simple Gaussian prior (Section 4.1) to improve the object detection results of our overall system.

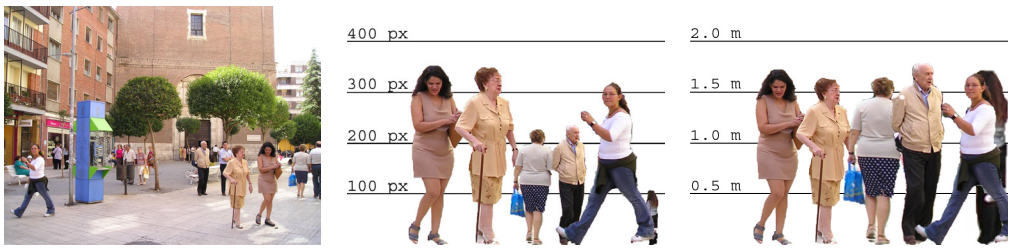


Figure 5.9: Automatic object height estimation. Objects manually segmented from a typical image in LabelMe dataset (left) are first shown in their original pixel size (center), and after being resized according to their automatically estimated 3D heights (right).

7.1. Discovery of Viewpoint and Object Size

Example-based techniques require many training samples to attain good performance. Manual labeling of camera viewpoint is tedious and error-prone, so we automatically recover the camera viewpoint and 3D object sizes in the LabelMe database [**116**]. Our method, described in [**75**], iterates between estimating the camera viewpoint for images that contain objects of known size distributions and estimating the size distributions of objects that are contained in images with known viewpoints. After initially providing only a guess of the mean and standard deviation of the height of people, we infer the camera viewpoint of over 5,000 images and heights of 13,000 object instances in roughly fifty object classes. The accuracy of our estimates are confirmed by manual measurements of camera pose on a subset of images, as well as by the inferred mean object heights (e.g., 1.67m for "women", 1.80m for "men", 1.37m for "parkingmeter", which are all within two cm of their true values). Figure 5.9 shows an example of our automatic 3D height estimation for people in an image.

Based on the inferred object heights, we re-estimate the 3D height distributions of cars (mean of 1.51m, standard deviation of 0.191m) and people (mean of 1.70m and standard deviation of 0.103m). We consider the camera viewpoint estimates to be reliable for the 2,660 images (excluding images in our test set) that contain at least two objects with known height distributions. Using the publicly available code, we compute the gist statistics (8x8 blocks at 3 scales with 8,4,4 orientations, giving 1280 variables per gist vector) over these images, providing a training set for viewpoint estimation.

	Prior	P+ObjGeom	Gist	G+ObjGeom
Mean Error	10.0%	4.3%	5.7%	3.8%

Table 5.4: We show the mean error (as a percentage of image height) in horizon position estimation using a Gaussian prior, after considering surface geometry and objects (using the Dalal-Triggs detectors), our initial gist-based estimates, and after the full inference.

7.2. Recovering the Viewpoint

In early experiments on our training set, we found that an image and its nearest neighbor (Euclidean distance in gist) tend to have similar horizon positions (correlation coefficient of 0.54). Using cross-validation, we evaluated nearest neighbor classifiers with various distance metrics, the nearest neighbor regression method of [**95**], and generalized regression neural networks (`newgrnn` in Matlab). The last of these (GRNN) provides the lowest cross-validation error, after tuning the spread $\alpha = 0.22$ on our training set. The horizon estimate from the GRNN is given by

$$\tilde{v}_0(\mathbf{x}) = \frac{\sum_i v_{0i} w_i}{\sum_i w_i} \text{ with } w_i = \exp\left[\frac{-\|\mathbf{x}-\mathbf{x}_i\|^2}{2\alpha^2}\right] \tag{5.6}$$

where $\mathbf{x}$ is the gist statistics and v_{0i} is the horizon position for the i^{th} training sample.

Empirically, the true horizon position v_0 given the regression estimate $\tilde{v}_0$ has a Laplace probability density

$$p(v_0|\tilde{v}_0) = \frac{1}{2s_v}\exp\left[\frac{-|v_0-\tilde{v}_0|}{s_v}\right] \tag{5.7}$$

where s_v is the scale parameter and is equal to the expected error. We found a correlation (coefficient 0.27) between error and Euclidean distance to the nearest neighbor in gist statistics. Therefore, we can provide better estimates of confidence by considering the nearest neighbor distance. We fit $s_v = 0.022 + 0.060\tilde{d}$ by maximum likelihood estimation over our training set, where $\tilde{d}$ is the nearest neighbor distance.

We were not able to improve camera height estimates significantly over our prior estimate, probably because a small change in camera height has little impact on the global image statistics. Therefore, we simply re-estimate the camera height prior as in Section 5 using our larger training set.

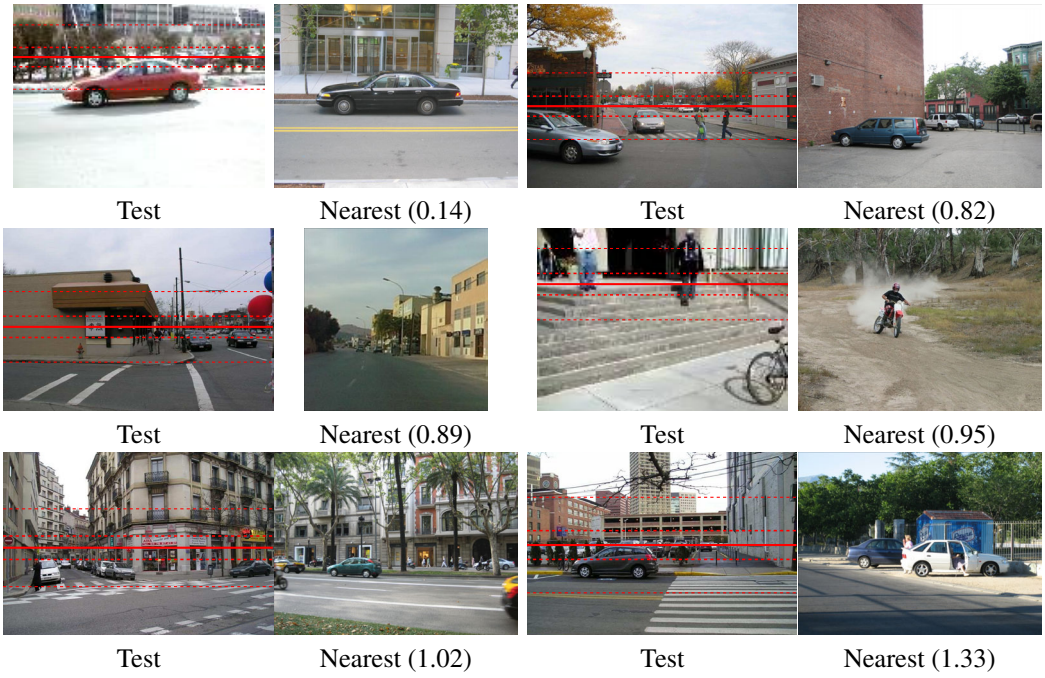
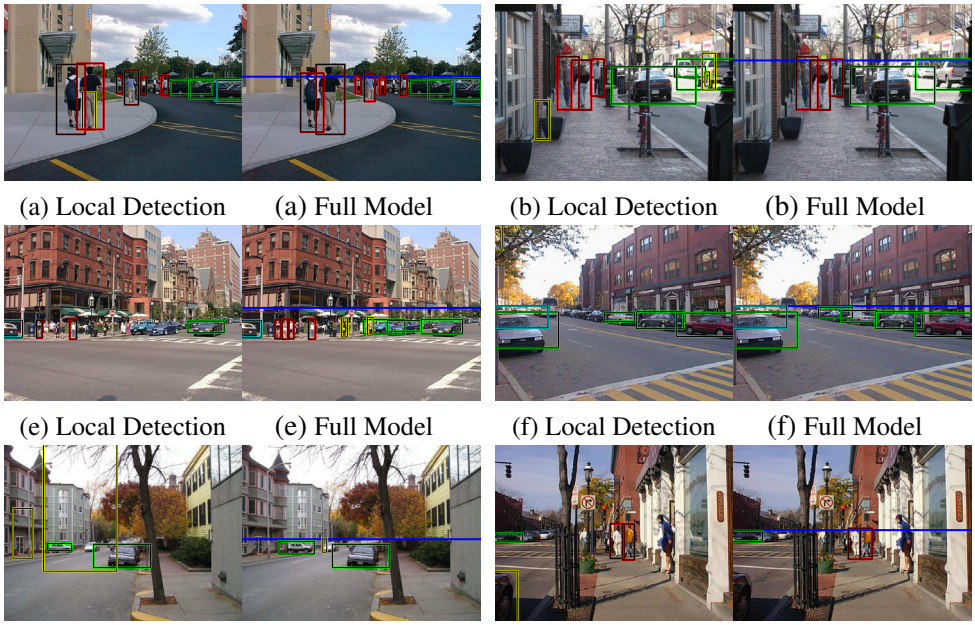


Figure 5.10: Horizon estimation. We show horizon estimation results (solid line) with 50% and 90% confidence bounds (dashed lines) for several test images with the gist nearest neighbor and distance $\tilde{d}$.

7.3. Evaluation

In Table 5.4, we show that our scene matching method for estimating the horizon position far outperforms the prior estimate (image center) and improves further when objects and surface geometry are considered. In Figure 5.10, we show several examples of test image, the nearest neighbor in our training set, and the estimated horizon. Our gist-based horizon estimates provide improvement in object detection as well, with detection rates increasing from 50% to 52% for cars and from 66% to 68% for pedestrians, at 1 false positive per image using the Dalal-Triggs object detectors. We show several examples of improved detection in Figure 5.11.

In summary, we can accurately estimate the horizon position from the gist statistics, providing: 1) a better final estimate of the horizon after considering objects and surface geometry; and 2) improved object detection. These experiments nicely reinforce a key idea of this dissertation: with appropriate integration, improvement in one task benefits the others.



(a) Local Detection (a) Full Model (b) Local Detection (b) Full Model

(e) Local Detection (e) Full Model (f) Local Detection (f) Full Model

(g) Local Detection (g) Full Model (h) Local Detection (h) Full Model

Figure 5.11: We show local object detections (left) of Dalal-Triggs (green=true car, cyan=false car, red=true ped, yellow=false ped) and the final detections (right) and horizon estimates (blue line) after considering surface geometry and camera viewpoint (initially estimated using our scene matching method). Our method provides large improvement (+7%/11% for peds/cars at 1 FP/image) over a very good local detector. Many of the remaining recorded "false positives" are due to objects that are heavily occluded (a,e) or very difficult to see (e) (i.e., missed by the ground truth labeler). In (h), a missed person on the right exemplifies the need for more robust assumptions (e.g., a person *usually* rests on the ground plane) or explanation-based reasoning (e.g., the person only looks so tall because he is standing on a step).

8. Conclusion

In this chapter, we have provided a "skeleton" model of a scene – a tree structure of camera viewpoint, objects, and surface geometry. We demonstrate our system's understanding of the 3D scene in Figure 5.12. Our model-based approach has two main advantages over the more direct "bag of features/black box" classification method: 1) subtle relationships (such as that object sizes relate through the viewpoint) can be easily represented; and 2) additions and extensions to the model are easy (the direct method requires complete retraining whenever anything changes).

To add a new object to our model, one needs only to train a detector for that object and supply the distribution of the object's height in the 3D scene. Our framework could also

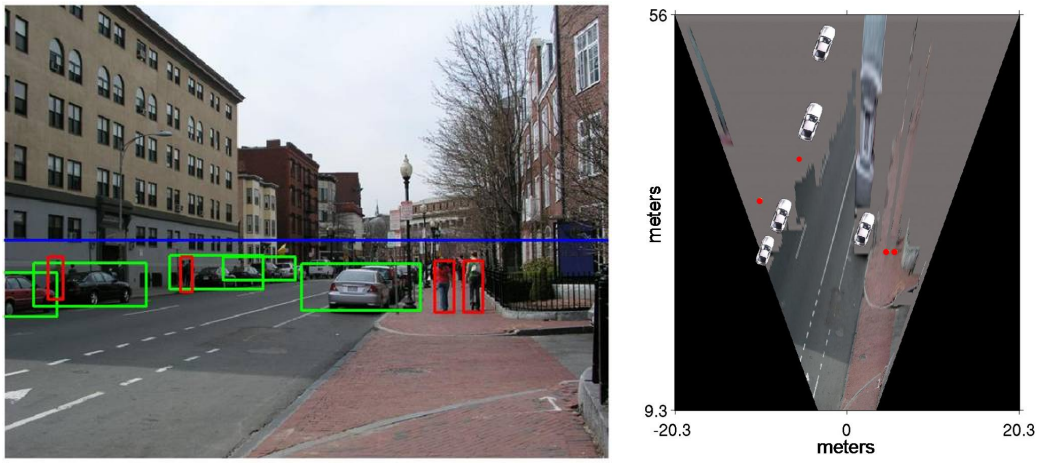


Figure 5.12: We project the estimated ground surface into an overhead view, using the estimated camera viewpoint, and plot scaled icons of objects (red dots for pedestrians) at their detected (using Dalal-Triggs) ground positions. Car orientation is estimated by a robust line fit, assuming that cars mostly face down the same line. To plot in metric scale, we assume a typical focal length.

be extended by modeling other scene properties, such as scene category. By modeling the direct relationships of objects and geometry (which can be done in 3D, since perspective is already part of our framework) further improvement is possible.

CHAPTER 6

Geometrically Coherent Image Interpretation

I was afraid that by observing objects with my eyes and trying to comprehend them with each of my other senses I might blind my soul altogether.
Socrates (469 BC - 399 BC), in "Phaedo" by Plato

One of the key ideas of this thesis is that the many visual processes should collaborate to produce a geometrically coherent image interpretation. We have already shown how surface and viewpoint information can be used together to improve object detection and how surface information can help to recover occlusion boundaries. In this chapter, we begin to investigate whether object and occlusion estimates can feed back to improve the surface estimates, and we develop an iterative algorithm for recovering a coherent spatial layout. An overview of our algorithm is illustrated in Figure 6.1. Our experimental findings reinforce some of our results from Chapter 5, as our method improves object recognition and viewpoint estimation in our widely varying Geometric Context Dataset.

1. Cues for Contextual Interaction

Using the algorithms described in the previous chapters, we estimate 3D surface orientations, occlusion relationships, object bounding boxes, and depth, which we represent as a set of confidence maps (Figure 6.2). In the following subsections, we describe how each aspect of the scene can be represented and used to help in the estimation of the others (illustrated in Figure 6.1). We summarize previously discussed contextual interactions and propose several new ones as well.

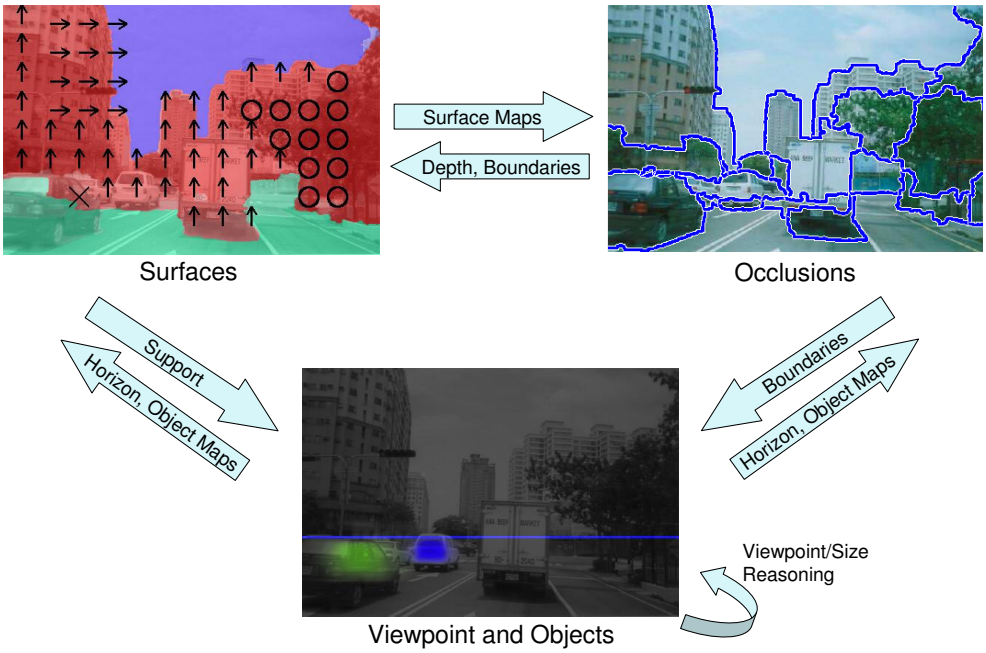


Figure 6.1: We show our final estimates for surfaces, occlusion boundaries, viewpoint, and objects and illustrate the interplay among them.

1.1. Surface Orientations

Surface orientations (Chapter 3) provide partial segmentation and depth information (with viewpoint) for occlusion reasoning and support information for object detection. We can represent our surface orientation information as a set of geometric confidence maps (Figure 6.2).

Since different objects often have different geometric classes, these geometric confidence maps can provide soft boundaries to aid segmentation. Further, the geometric class labels, with a segmentation and camera viewpoint, can be used to estimate depth.

Surface information helps with object recognition in two main ways. First, an object is typically vertical and often corresponds to a particular geometric subclass (e.g., a person is "solid, non-planar"). Regions that are thought to be ground surfaces or porous surfaces are unlikely to be people. Although much of this information is implicitly modeled in the detector itself, which locally analyzes an image patch to determine whether it is an object

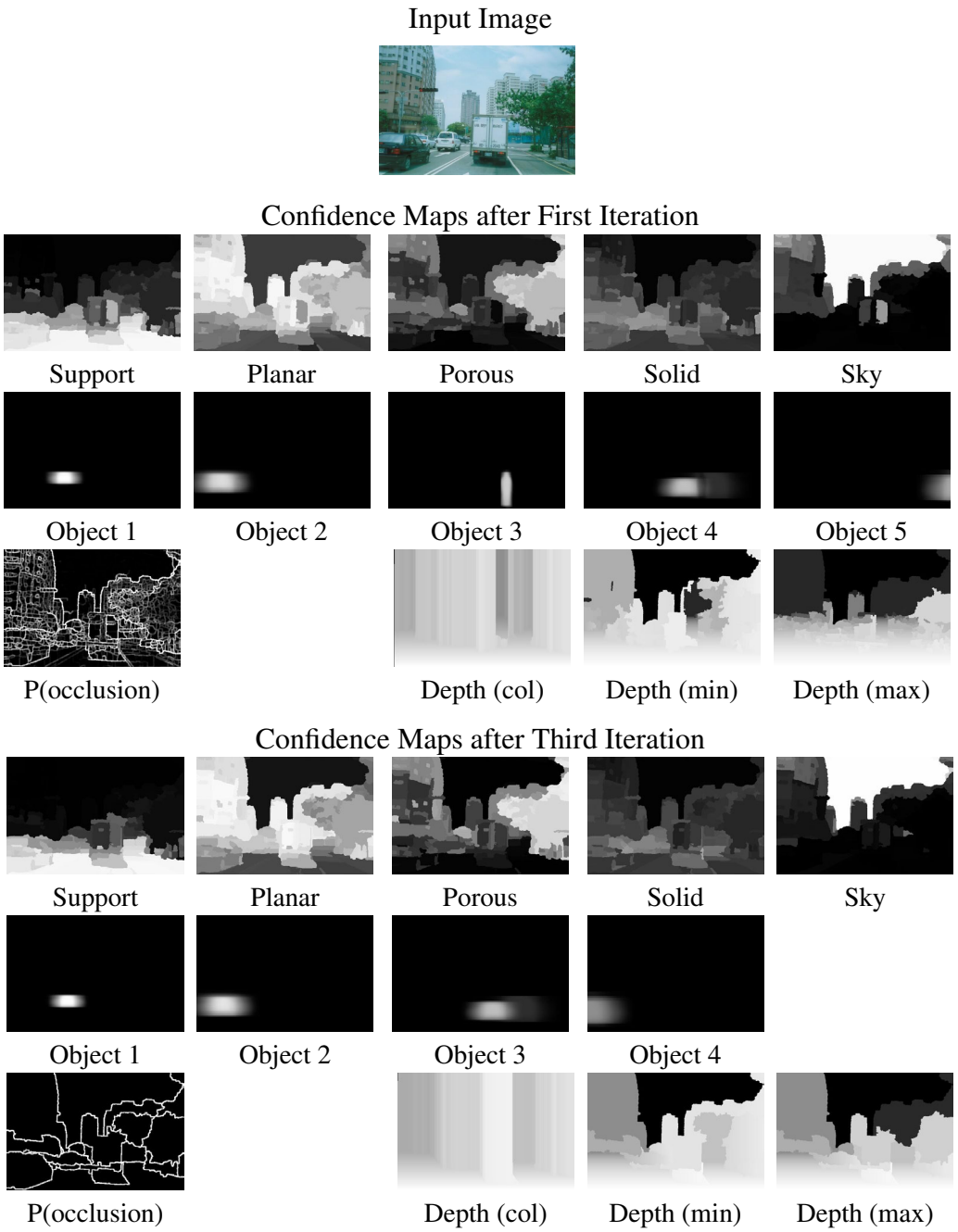


Figure 6.2: Confidence map representation. We show surface, object, and occlusion estimates after the first and third iterations. In the confidence maps, higher intensity indicates higher confidence. By the third iteration, only four object candidates remain plausible. Depth is estimated in three different ways (as in Chapter 4), and closer points are shown as brighter.

Input Image | Objects/View (Iter 1) | Occlusion (Iter 2) | Objects/View (Iter 3)

Figure 6.3: Example of the benefits of boundary reasoning. Before considering occlusion information, the many coats on a rack are mistaken for pedestrians. During the occlusion reasoning, however, it is determined that the coats form a single free-standing structure. This causes the object candidates along the coat rack to be discarded, and re-running the object/viewpoint/surface inference provides two true detections (partially obscured by the coat rack).

or not, the surface estimates incorporate more global information about the scene through the multiple segmentation process. For example, a pedestrian detector is often fooled by a patch that locally has pedestrian-like edge characteristics but globally can be seen to be part of a larger foliage region. Second, the surface information provides "support" information about an object. For instance, people and cars typically rest on the ground, so a patch with a visible support region beneath it is more likely to be a person than a region surrounded by vertical or sky regions.

1.2. Occlusion Relationships

We represent occlusion relationships with confidences over the boundaries and depth estimates (Figure 6.2). Occlusion information can help surface estimation by improving spatial support and providing depth estimates. In each segment produced by the multiple segmentation algorithm, we look up the confidence of the most likely internal boundary which helps determine whether a segment is likely to correspond to a single label. Also, the average depth and a measure of the slope in depth from left to right is computed each segment (for all three depth estimates). The average depth may help determine the label of the segment since appearance characteristics vary with distance (e.g., the texture in foliage is lost at a distance). The slope in depth may help determine whether a planar segment faces the left, center, or right of the viewer.

Occlusion information can help object detection in three ways. First, if the bottom of an object is entirely occluded, it provides an explanation for not seeing ground directly beneath the object. Second, knowledge of an occlusion can explain the unlikely appearance of an object part, preventing a false rejection that could occur if the detector expects the object to be wholly visible. Third, occlusion reasoning can help remove false object detections by showing them to be part of a larger structure. For example, the coats in Figure 6.3 individually appear to be pedestrians (and are consistent in viewpoint) but are discarded when found to be part of the larger coat rack. In our experiments, we implement only the last of these.

1.3. Camera Viewpoint

The camera viewpoint is a two-parameter fit to the ground plane (the plane that supports most objects of interest) with respect to the camera, corresponding loosely to the horizon position and the camera height. Knowledge of the camera viewpoint can be used to improve surface estimation, get more accurate relative depth maps for occlusion reasoning, and to help determine whether an image patch could be an object by measuring its 3D height.

As a general rule, support surfaces tend to be low in the image, and sky tends to be high. Knowledge of the horizon (the vanishing line of the ground plane) makes this generality more precise, aiding surface classification. As cues, we represent the differences between the horizon position and the top and bottom (10th and 90th percentile of row-position) of the segment.

With knowledge of the camera height and the horizon position, height in the image of an outlined object resting on the ground plane can be computed. We can therefore use the viewpoint to determine whether potential objects in the image are the correct 3D size. Given the viewpoint, we can also compute relative depth (up to a scale) of two points on the ground, providing a valuable clue for occlusion reasoning.

1.4. Object Detection

If we can detect an object, we often have a good idea of the surface characteristics and boundary of the object region. Further, when the object is grounded, we can estimate the

viewpoint based on the size distribution of the object class. When displaying results, we usually show only the most likely position for a detected object, but we do maintain a distribution over potential bounding boxes. We can use this distribution, along with an expected mask of the object class, to compute the likelihood that each pixel is part of an object instance, based on the bounding boxes and confidences from the detector. We compute the expected mask from manually segmented objects in LabelMe [**116**]. See Figure 6.2 for examples. The result is an "object map" for each object that is detected with confidence greater than some threshold. The intensity of an object map gives the probability that a particular pixel is part of the given object times the probability that the object exists. Objects at lower image positions are assumed to be closer to the viewer, so if two objects overlap, the confidences of the further object pixels are multiplied by one minus the confidences of the closer object pixels. If the object maps are summed, therefore, a pixel intensity will correspond to the likelihood that the pixel is generated by *any* detectable object.

Each object tends to have a corresponding geometric class. For instance, people are "solid". If we can identify the object label of a region, therefore, we are likely to know the surface label. To provide this cue, we sum pixel confidences for each type of object (cars and pedestrians) and compute the mean over each region.

In occlusion reasoning, it is helpful to know whether neighboring regions are likely to be part of the same individual object. To represent this, we first compute the mean and max confidences for each individual object with each region. As cues, we then compute: the sum (over objects) of the mean confidences (giving total object likelihood); the sum of the absolute difference of the mean confidences between two regions (an overall measure of the object confidence difference); the confidence of the most likely individual object within each region; and the maximum absolute difference between individual object confidences.

2. Training and Inference

Our training and inference are performed in a simple iterative manner, cycling through the surface estimation, object detection and viewpoint recovery, and occlusion reasoning. We outline our training and inference algorithm in Figure 6.4. Each of our algorithms are evaluated using our Geometric Context dataset. The first fifty images are used for training the surface segmentation and occlusion reasoning. The remaining 250 are used to test the

TRAINING

Initialize:
- Obtain multiple segmentations for each training image
- Estimate horizon
- Perform local object detection

For iteration t = 1..3:
1. Train and apply surface estimation
   (a) Compute features for each segment (using results of (2),(3) from iterations $1 \ldots t-1$)
   (b) Train surface classifiers and compute surface confidences with cross-validation
2. Apply object/viewpoint/surface inference (using result of (1) from iteration t and (3) from $t-1$)
3. Train and apply occlusion reasoning algorithm (using results of (1), (2) from iteration t)
   (a) Train on hold-out set
   (b) Perform occlusion reasoning on test images

Figure 6.4: Iterative training algorithm for combining surface, occlusion, viewpoint, and object information. Training and testing is performed on the Geometric Context dataset. The holdout set of 50 images used to train the surface segmentation algorithm is used to train the occlusion reasoning, and the remaining 250 images are used for testing (using five-fold cross-validation for the surface estimation).

surface, object, viewpoint, and occlusion estimators. The surface classifiers are trained and tested using five-fold cross-validation.

In training and testing the surface classifiers, the multiple segmentations are computed once. In each iteration after the first, the cues for each segment are updated with information gleaned from the latest object, viewpoint, and occlusion estimates. The object/viewpoint inference uses the surface estimates from the current iteration and the occlusion information from the previous iteration (starting in the second iteration). The occlusion algorithm uses the latest surface, object, and viewpoint estimates. The first two iterations of the occlusion algorithm correspond to the first two iterations of the original algorithm, with additional cues from the latest surface, object, and viewpoint estimates. In the third iteration, the occlusion algorithm re-iterates until convergence. In all other respects, the training and testing for the three algorithms is implemented as described in the previous chapters.

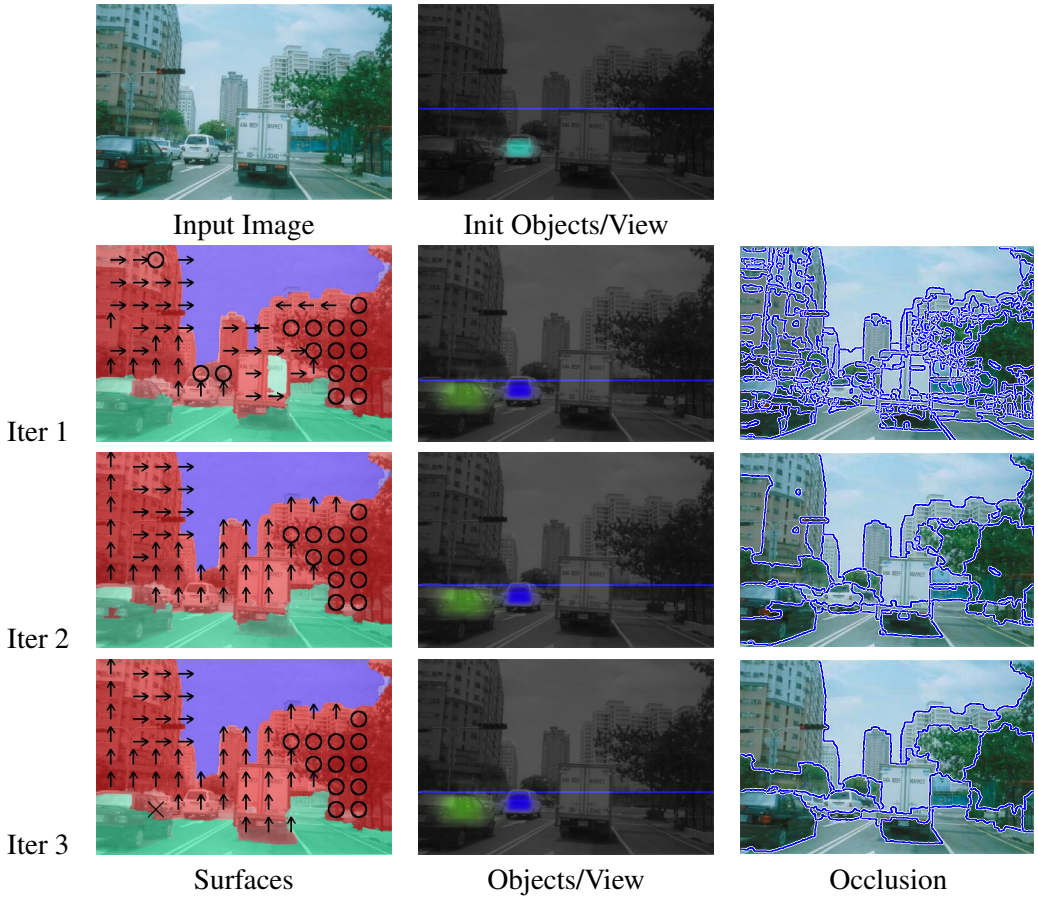


Figure 6.5: We illustrate our iterative procedure to infer surfaces, object and viewpoint, and occlusion boundaries. Our initial horizon estimate (based on scene matching) is corrected, and a missed car is detected after the first iteration. The detections and occlusion information is then used to improve the accuracy of the geometric class labels in the second iteration. Another slight improvement occurs in the third iteration, after which the final estimate of the occlusion boundaries is given. Notice that, while the occlusion boundaries are far from perfect (e.g., the truck has a building for an appendage), most of the buildings and trees and vehicles are reasonably well delineated. The direction of the occluding contour is also estimated but not shown to improve clarity.

3. Experiments

We applied the training and inference procedure described in the previous section to our Geometric Context dataset. We show an example of results after each iteration in Figure 6.5 and more examples of final results in Figure 6.6. Here, we discuss the improvement in each type of estimation. Our analysis includes qualitative assessment, inspection of the decision tree classifiers learned in the surface and occlusion estimation, and quantitative performance comparison. In the decision tree learning, early and frequent selection of a cue indicates

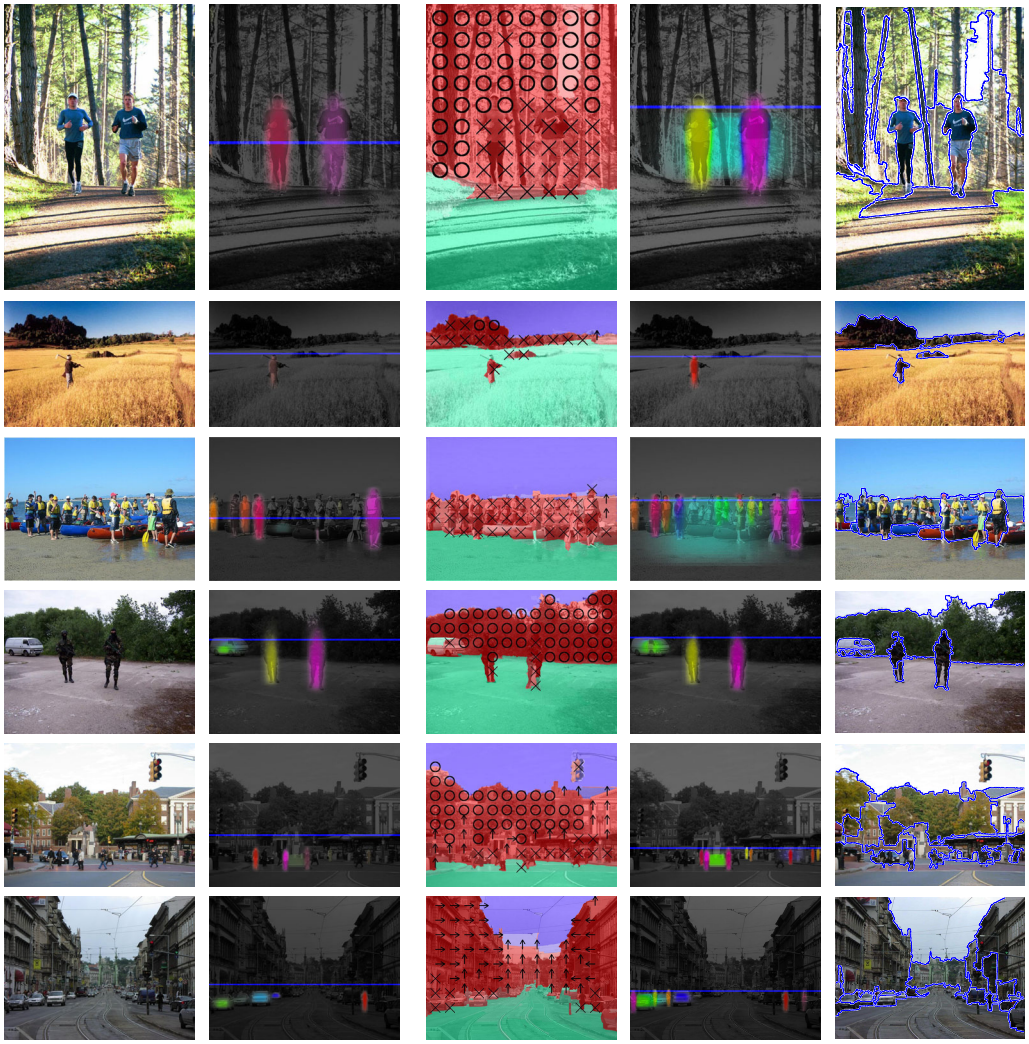


Figure 6.6: Geometrically coherent image interpretation results. In each row, we show the input image, the initial detections and viewpoint estimate and the final estimates for surfaces, objects, viewpoint, and occlusion boundaries. Row 1: The occlusion reasoning does well in a difficult image, and both pedestrians are detected (with a false car detection). Row 2: The man is correctly identified and parsed. Row 3: The few initial detections lead to detecting most of the people (with a false car detection). Row 4: the soldiers need better camouflage as the algorithm is able to identify and segment them away from the background. Row 5 and 6: A reasonably good job is done in complicated scenes.

that the cue is valuable but does not necessarily imply great value beyond the other cues, as there may be redundant information.

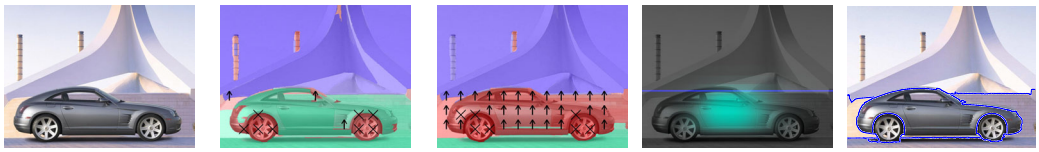


Image Initial Surfaces Obj/View Occlusion

Figure 6.7: Here, initially poor surface estimates (the worst in our test set) are greatly improved after considering object and occlusion information. The rightmost three images show our final results.

3.1. Surfaces

To train our surface classification algorithm, we train: a pairwise classifier to determine whether two superpixels are likely to be of the same geometric class; a segmentation classifier to determine whether all superpixels (or a large majority) within a given segment are likely to have the same label; a main classifier to distinguish among "support", "vertical", and "sky" regions; and a subclassifier to label vertical surfaces as planar, "left", "center", or "right", or non-planar, "porous", or "solid". When more than one class is concerned, the classifier is trained as several binary logistic classifiers (one per class). The main difference from our experiments in Chapter 3 is that occlusion, object, and viewpoint information is available to these classifiers.

The decision tree learning indicates that the boundary likelihood from occlusion reasoning is the most powerful cue for the segmentation classifier, as it is the first and most frequently selected. For the "solid" classifier, the pedestrian confidence value from the object detection is the first feature selected. The learning algorithm determines that regions with high pedestrian confidence are very likely to be solid, but regions without pedestrian confidence are not much less likely (i.e., all pedestrians are solid, but not all solids are pedestrians). Our measure of depth slope from the occlusion reasoning is used frequently by the planar "left" and "right" classifiers.

We also find that the position cues relative to the estimated horizon position are used overall twice as frequently as the absolute position cues. In a separate experiment, in which we train and test surface classification with manually assigned (ground truth) horizon estimates, we find that both the main (support, vertical, sky) classification and subclassification of vertical (left, center, right, porous, solid) each improve by 2% (from 87% to 89% and 61% to

63%, respectively). Thus, knowledge of the horizon is an important cue, but the potential improvement in surface estimation by improving the horizon estimate is limited.

In Figure 6.7, we show one example in which poor initial surface estimates were greatly improved due to the object and occlusion information. We find a modest quantitative improvement in surface classification with the inclusion of object and occlusion information. Following the original algorithm from Chapter 3, the main classification accuracy is 86.8% and subclassification accuracy is 60.9% (these are slightly different than the numbers reported earlier, due to a differences in the random segmentations and a change in the number of segmentations and number of segments per segmentation). These accuracies improve to 87.2% and 62.6% in the second iteration. In the third iteration, main classification improves to 87.6%, but subclassification accuracy decreases to 61.8%. The overall quantitative improvement is about 1%.

We do notice significant qualitative improvement in the results. We note that, while correcting the label of a 1,000 pixel region will improve quantitative results by little more than 0.1% for the image, the qualitative improvement could be large. Looking at results on an image-by-image basis, of the 57% of images that change in error by more than 1%, 61% improve. Of the 18% that change in error by more than 5%, 74% improve. Thus, while large changes are rare, changes made after considering the new object and occlusion information are much more likely to improve results than not.

Larger gains are possible. In our experiments, we estimate surfaces based on the same segmentations throughout our iterative algorithm and merely adjust the confidences or weights of the different segmentations. One potentially more effective way to use the occlusion information would be to generate new segmentations as well. We also note that the object detection and occlusion reasoning already consider the surface information and, therefore, using them to try to improve surface classification may largely reinforce the initial surface estimates.

3.2. Objects

Our object detection is based on the local Dalal-Triggs [**21**] object detectors, which were engineered particularly for pedestrian detection. As in Chapter 5, we perform inference in

	Car	Ped
Local	41%	54%
Iter 1	41%	66%
Iter 2	46%	66%
Iter 3	51%	61%

Table 6.1: Detection rates at 1 false positive per image for cars and pedestrians on the Geometric Context dataset.

a graphical model solving for objects, viewpoint, and surfaces (a simple, local representation for the last of these). The main difference in these experiments is that we remove candidates when they are shown to be a small part of a larger structure, according to the occlusion information. Here, we report the detection rate at 1 false positive per image for cars and pedestrians in our Geometric Context dataset, ignoring cars smaller than 24 pixels tall and pedestrians smaller than 48 pixels tall (these are the minimum sizes of the respective detector windows). In total, the test portion (excluding the occlusion training set portion) of the dataset contains 82 cars and 379 pedestrians.

In Table 6.1, we show the detection rates at one false positive per image for cars and pedestrians in our Geometric Context dataset. When viewpoint and surfaces are considered (iteration 1), pedestrian detection improves considerably. When occlusion information is considered (iterations 2 and 3) car detection improves but pedestrian detection drops slightly, likely due to the difficulty of maintaining occlusion boundaries for distant pedestrians. Along with many false positives, 11% of true pedestrian and 8% of true car detections are discarded by the occlusion-based filtering (see Figure 6.8 for an example). Overall, the car detection improves by 10% and the pedestrian detection by 7% when considering surface, viewpoint, and occlusions. These results are encouraging, as the Geometric Context dataset used in these experiments has much greater scene variety (forests, mountains, oceans, etc.) than than the primarily urban images of the LabelMe dataset.

We expect that larger improvement would be observed if the occlusion reasoning and object detection processes were more fully integrated. We also note that the base detector had a maximum recall of 65% for cars and 89% for pedestrians, after using a low confidence threshold. For cars, the limited recall is partially due to using a single detector for cars at

Obj/View (Iter 2) Occ (Iter 2) Obj/View (Final)

Figure 6.8: In this case, one of the initially detected small pedestrians (left) are incorrectly grouped together with the background (center), causing the pedestrian to be discarded in the final estimates (right). Fuller integration between object detection and occlusion estimation is needed.

all viewpoints. The non-maximum suppression could also prevent recall of very closely spaced objects.

3.3. Viewpoint

We evaluate viewpoint estimation based on the horizon position, since it is difficult to obtain ground truth for camera height. Using the mean horizon position as a constant estimate yields an error of 12.8% (percentage of image height difference between the manually labeled horizon and the estimated horizon). This error drops to 10.4% when using the data-driven horizon estimate with the LabelMe training set. It is encouraging that we are able to get this significant improvement on our Geometric Context dataset, since many of the types of scenes are not well represented in LabelMe. After considering objects and surfaces in the first iteration, the error drops further to 8.5%. In the second iteration, the error rises slightly to 9.1% for reasons that are not clear, since an overall increase in object detection occurs. One possibility is that false positives tend to be of similar 3D height to the objects and aid the horizon estimation by chance.

3.4. Occlusion

Our occlusion experiment is performed as in Chapter 4, except that object information is used in the boundary classification. Inspection shows that individual cars and people are correctly delineated much more frequently when the object information is considered, though the mistake of segmenting the head from the body is still common. We measure no

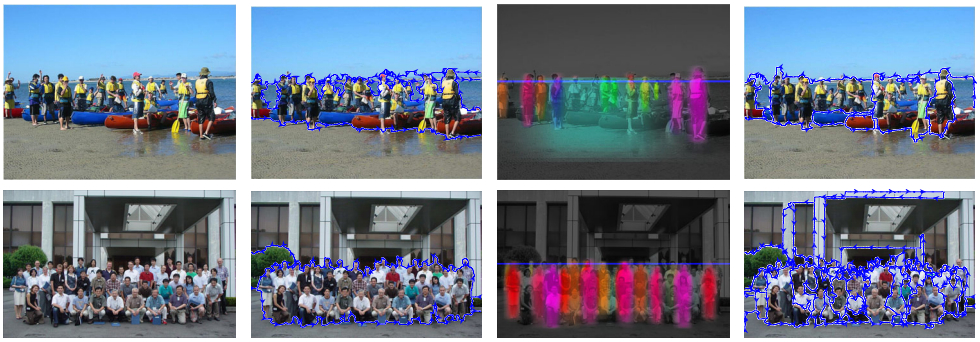


Image | Occ Ground Truth | Objects/View | Occ Result

Figure 6.9: By reasoning together about objects and occlusions, we are sometimes able to find the occlusion boundaries of tightly crowded individuals. In these cases, our occlusion estimates are more precise than the ground truth, which leads to an artificially low quantitative measure of efficiency.

overall quantitative improvement, which may be due to ground truth labeling of a crowd of people or row of cars as a single object (see Figure 6.9 for examples).

We believe that using the object detections to influence the original superpixels and performing an object-based segmentation would improve occlusion reasoning for those objects further.

4. Remarks

We note that maximum potential quantitative gain in these experiments is limited by several factors. Primarily, quantitative improvement in surface estimates are limited by the small total area of detected objects, the limited influence of occlusion estimates on spatial support, and the fact that surface estimates contribute to the detection and occlusion algorithms. Object detection improvement due to occlusion reasoning is limited by the feed-forward decision manner in which inconsistent object hypotheses are discarded. Occlusion reasoning quantitative improvement is limited by the ground truth annotation which often groups together a crowd of people or a line of cars. Thus, sometimes we may recover the occlusion boundaries for some of the individuals standing in a group, but our quantitative measure of efficiency will decrease when the entire group is given a boundary. In cases like this (Figure 6.9), we are arguably doing better than ground truth.

Even so, our results confirm that our object/surface/viewpoint inference improves detection results considerably, even on a dataset with great variety of scenes and viewpoint. We

achieve modest quantitative and more substantial qualitative improvement due do modeling interactions among the various aspects of the scene. We believe that additional effort toward fuller integration would improve results further. We conclude in Figure 6.10 by showing several failure cases with suggestions for how to improve the algorithm to better handle these cases.

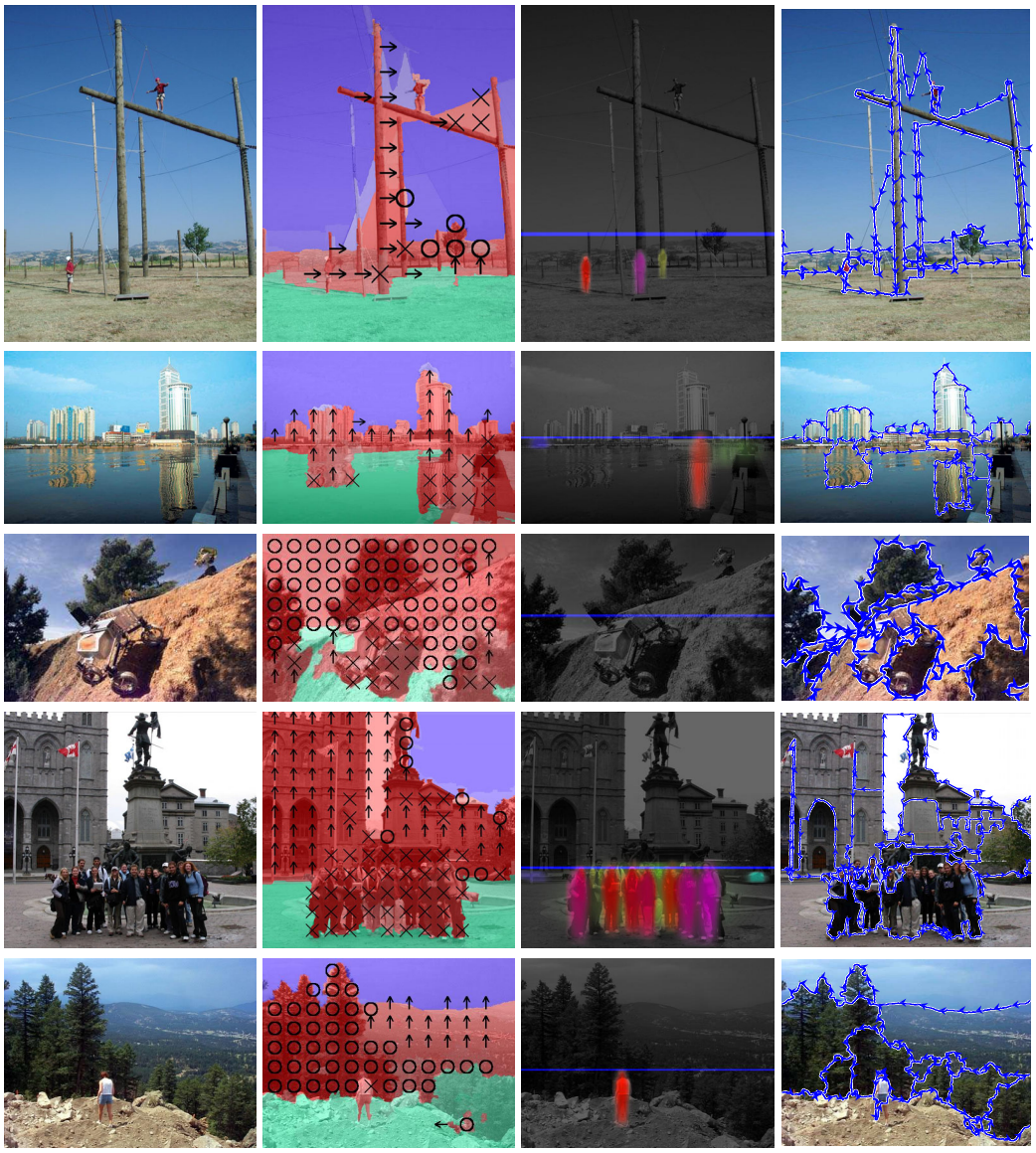


Figure 6.10: Examples that illustrate the need for improvement. Row 1: The "objects rest on the ground plane" assumption needs to be relaxed (e.g., "objects *usually* rest on the ground plane"). Row 2: Reflections cause surface errors. Also, reasoning about water (people tend not to walk on water) could improve detection. Row 3: More extensive surface labels would provide a more precise description of this steep slope. Row 4: Detection and occlusion need to be more tightly integrated. Here, many pedestrians are detected, but they are grouped together in the final occlusion estimates. Row 5: Our occlusion (and depth) models need to be extended to handle the case of a supporting surface occluding vertical objects, as in the ledge here.

CHAPTER 7

Applications

Spatial layout is foundational to scene understanding. Consequently, we can accomplish many useful tasks with simple operations on our spatial layout representation. Here, we demonstrate applications of single-view 3D reconstruction, object pasting, object discovery, and robot path-planning.

1. Automatic Photo Pop-up

Here, we show that surface and viewpoint estimates are sometimes sufficient to automatically reconstruct a simple 3D model of the scene. Our approach is similar to the creation of a pop-up illustration in a childrens book: the image is laid on the ground plane and then the regions that are deemed to be vertical are automatically popped up onto vertical planes. Just like the paper pop-ups, our resulting 3D model is quite basic, missing many details. Nonetheless, a large number of the resulting walkthroughs look surprisingly realistic and provide a fun "browsing experience".

1.1. Background

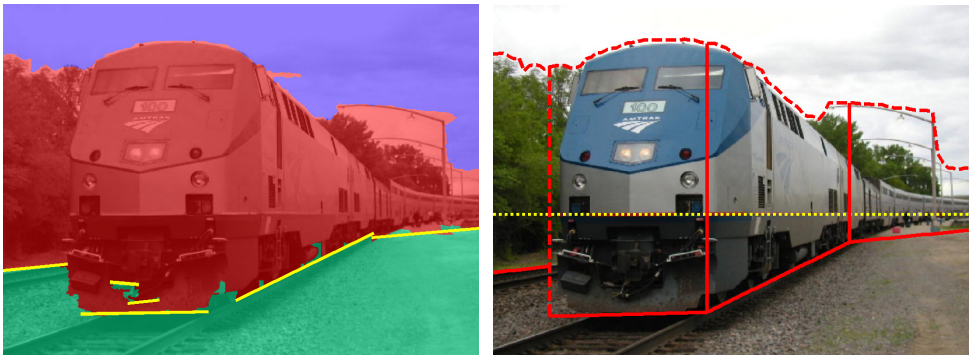
The most general image-based rendering approaches, such as Quicktime VR [**12**], Light-fields [**79**], and Lumigraph [**33**] all require a huge number of photographs as well as special equipment. Popular urban modeling systems such as Façade [**22**], PhotoBuilder [**14**] and REALVIZ ImageModeler greatly reduce the number of images required and use no special equipment (although cameras must still be calibrated), but at the expense of considerable user interaction and a specific domain of applicability.

Several methods are able to perform user-guided modeling from a single image. Liebowitz, Criminisi and Zisserman [**81, 19**] offer the most accurate (but also the most labor-intensive) approach, recovering a metric reconstruction of an architectural scene by using projective geometry constraints [**41**] to compute 3D locations of user-specified points given their projected distances from the ground plane. The user is also required to specify other constraints such as a square on the ground plane, a set of parallel "up" lines and orthogonality relationships. Most other approaches forgo the goal of a metric reconstruction, focusing instead on producing perceptually pleasing approximations. Zhang et al. [**154**] model free-form scenes by letting the user place constraints, such as normal directions, anywhere on the image plane and then optimizing for the best 3D model to fit these constraints. Ziegler et al. [**155**] find the maximum-volume 3D model consistent with multiple manually-labeled images. *Tour into the Picture* [**54**], the main inspiration for this work, models a scene as an axis-aligned box, a sort of theater stage, with floor, ceiling, backdrop, and two side planes. An intuitive "spidery mesh" interface allows the user to specify the coordinates of this box and its vanishing point. Foreground objects are manually labeled by the user and assigned to their own planes. This method produces impressive results but works only on scenes that can be approximated by a one-point perspective, since the front and back of the box are assumed to be parallel to the image plane. This is a severe limitation (that would affect most of the images in our results), which has been partially addressed by Kang et al. [**65**] and Oh et al. [**98**], but at the cost of a less intuitive interface.

While automatic methods to reconstruct certain types of scenes from multiple images or video sequences (e.g., [**96, 109**]) existed previously, ours was the first attempt, to the best of our knowledge, to attempt single-view modeling of general scenes. Recently, Delage et al. [**23**] has proposed a method to reconstruct indoor scenes, with some restrictions, and Saxena et al. [**119**] have proposed a method to reconstruct outdoor scenes that also provides metric depth estimates and can handle sloping surfaces.

1.2. Algorithm

We construct a simple 3D model by making "cuts" and "folds" in the image based on the geometric labels (see Figure 7.1). Our model consists of a ground plane and planar objects at right angles to the ground. Even given these assumptions and the correct labels, many



(a) Fitted Segments (b) Cuts and Folds

Figure 7.1: From the noisy geometric labels, we fit line segments to the ground-vertical label boundary (a) and form those segments into a set of polylines. We then "fold" (red solid) the image along the polylines and "cut" (red dashed) upward at the endpoints of the polylines and at ground-sky and vertical-sky boundaries (b). The polyline fit and the estimated horizon position (yellow dotted) are sufficient to "pop-up" the image into a simple 3D model.

Partition the pixels labeled as vertical into connected regions
For each connected vertical region:

1. **Find ground-vertical boundary** (x,y) locations $\mathbf{p}$
2. **Iteratively find best-fitting line segments** until no segment contains more than m_p points:
   (a) Find best line L in $\mathbf{p}$ using Hough transform
   (b) Find largest set of points $\mathbf{p}_L \in \mathbf{p}$ within distance d_t of L with no gap between consecutive points larger than g_t
   (c) Remove $\mathbf{p}_L$ from $\mathbf{p}$
3. **Form set of polylines from line segments**
   (a) Remove smaller of completely overlapping (in x-axis) segments
   (b) Sort segments by median of points on segment along x-axis
   (c) Join consecutive intersecting segments into polyline if the intersection occurs between segment medians
   (d) Remove smaller of any overlapping polylines

Fold along polylines
Cut upward from polyline endpoints, at ground-sky and vertical-sky boundaries
Project planes into 3D coordinates and texture map

Figure 7.2: Procedure for determining the 3D model from labels.

possible interpretations of the 3D scene exist. We need to partition the vertical regions into a set of objects (especially difficult when objects regions overlap in the image) and determine where each object meets the ground (impossible when the ground-vertical boundary is obstructed in the image). We have found that, qualitatively, it is better to miss a fold or a cut than to make one where none exists in the true model. Thus, here, we do not attempt to segment overlapping vertical regions, placing folds and making cuts conservatively. In the next

section, we show some preliminary work in handling foreground objects by incorporating occlusion estimates.

As a pre-processing step, we set any superpixels that are labeled as ground or sky and completely surrounded by non-ground or non-sky pixels to the most common label of the neighboring superpixels. In our tests, this affects no more than a few superpixels per image but reduces small labeling errors.

Figure 7.2 outlines the process for determining cuts and folds from the geometric labels. We divide the vertically labeled pixels into disconnected or loosely connected regions using the connected components algorithm (morphological erosion and dilation separate loosely connected regions). For each region, we fit a set of line segments to the region's boundary with the labeled ground using the Hough transform [**25**]. Next, within each region, we form the disjoint line segments into a set of polylines. The pre-sorting of line segments (step 3(b)) and removal of overlapping segments (steps 3(a) and 3(d)) help make our algorithm robust to small labeling errors.

If no line segment can be found using the Hough transform, a single polyline is estimated for the region by a simple method. The polyline is initialized as a segment from the left-most to the right-most boundary points. Segments are then greedily split to minimize the $L1$-distance of the fit to the points, with a maximum of three segments in the polyline.

We treat each polyline as a separate object, modeled with a set of connected planes that are perpendicular to the ground plane. The ground is projected into 3D coordinates using the horizon position estimate and fixed camera parameters. The ground intersection of each vertical plane is determined by its segment in the polyline; its height is specified by the region's maximum height above the segment in the image and the camera parameters. To map the texture onto the model, we create a ground image and a vertical image, with non-ground and non-vertical pixels having alpha values of zero in each respective image. We feather the alpha band of the vertical image and output a texture-mapped VRML model that allows the user to explore his image.

To obtain true 3D world coordinates, we would need to know the intrinsic and extrinsic camera parameters. We can, however, create a reasonable scaled model by estimating the horizon line (giving the angle of the camera with respect to the ground plane) and setting

(a) (b)

(c) (d)

Figure 7.3: Failures. Our system creates poor 3D models for these images. In (a) some foreground objects are incorrectly labeled as being in the ground plane; others are correctly labeled as vertical but are incorrectly modeled due to overlap of vertical regions in the image. In (b), the reflection of the building is mistaken for being vertical. Major labeling error in (c) results in a poor model. The failure (d) is due to the inability to segment foreground objects such as close-ups of people.

the remaining parameters to constants. We assume zero skew and unit affine ratio, set the field of view to 0.768 radians (the focal length in EXIF metadata can be used instead of the default, when available), and arbitrarily set the camera height to 5.0 units (which affects only the scale of the model).

1.3. Results

Figure 7.4 shows the qualitative results of our algorithm on several images. For these experiments, we trained our geometric labeling system on a smaller set of 82 images. On a test set of 62 novel images, 87% of the pixels were correctly labeled into ground, vertical, or sky. Even when all pixels are correctly labeled, however, the model may still look poor if object boundaries and object-ground intersection points are difficult to determine. In our experiments, we found that our system produces reasonable models for about 30% of input images of outdoor scenes.

Figure 7.3 shows four examples of typical failures. Common causes of failure are 1) labeling error, 2) polyline fitting error, 3) modeling assumptions, 4) occlusion in the image, and 5) poor estimation of the horizon position. Under our assumptions, crowded scenes (e.g. lots of trees or people) cannot be easily modeled. Additionally, our models cannot account for slanted surfaces (such as hills) or scenes that contain multiple ground-parallel planes

Figure 7.4: Input images and novel views taken from automatically generated 3D models.

(e.g. steps). Since we do not currently attempt to segment overlapping vertical regions, occluding foreground objects cause fitting errors or are ignored (made part of the ground plane). Additionally, errors in the horizon position estimation (our current method is quite basic) can cause angles between connected planes to be overly sharp or too shallow.

2. Automatic Photo Pop-up with Occlusion Boundaries

In the previous section, we used only surface estimates and the horizon position to create a simple 3D model of the scene. As a consequence, the method does not handle foreground objects or cluttered scenes well. In this section we incorporate the occlusion, viewpoint, and object information, which are estimated using the algorithm described in Chapter 6. The object detections provide better occlusion boundaries for cars and pedestrians. The occlusion boundaries allow foreground objects to be separated from the background and modeled as individual "billboards". The viewpoint estimates provide the correct perspective. Overall, we are better able to handle cluttered scenes.

2.1. Algorithm

When the ground-contact points of a region are visible, we can fit a ground-vertical boundary polyline to those points (as in the last section) to model the region with a series of planes. When the ground-contact points are obscured, however, we cannot measure the depth or orientation of the region directly. Our solution is to repeat the hierarchical merging process from Chapter 4, attaching each region of unknown depth to a region of known depth, while preventing merges between regions that already have 3D plane estimates. To do this merging, we need to estimate boundary likelihoods. We estimate these likelihoods using the occlusion third stage unary classifiers from Chapter 6 and the latest object and surface information. The occlusion process yields surface estimates that are consistent with the occlusion boundaries but may differ from our original estimates from the algorithm of Chapter 3. In our implementation, we simply average the two surface estimates to label regions as "support", "vertical", or "sky". In other respects, the 3D scene reconstruction is performed as described in the previous section.

2.2. Results

We show several preliminary results in Figure 7.5, with wireframe views of the top-row result shown in Figure 7.6. The 3D models demonstrate a good understanding of the spatial layout of the scene. Overall, in the 250 images we processed (from the Geometric Context

Figure 7.5: Input images (left) and novel views taken from automatically generated 3D models, using estimated occlusion boundaries to separate foreground objects. In row 3, we view the scene from behind the man on the left and from behind the crowd on the right. In rows 4 and 5, we many objects are segmented from the background, including cars, people, and goats. In row 6, the sign and arch are well-modeled. Note that none of these scenes, except the second row, could be modeled with the original Automatic Photo Pop-up algorithm, due to the foreground objects. These are preliminary results.

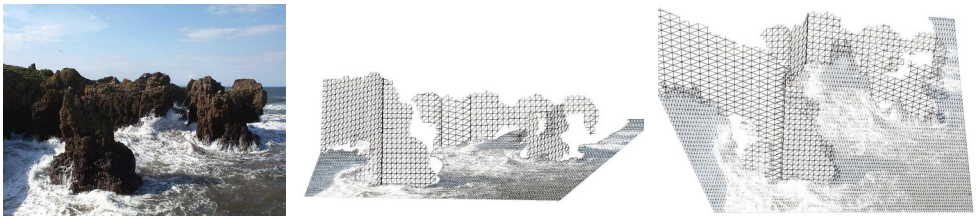


Figure 7.6: Wireframe (without texture) views of 3D model of coastal scene. Our 3D models capture the basic 3D structure of the scene. The texture-mapped surfaces can fool human observers into thinking that the models are much more detailed than they are, indicating that monocular cues are important even for small geometric details.

dataset), about 40% of the models were "pretty good".[1] In earlier work [**47**], we reported a success rate of about 30% on a slightly simpler subset of the data (the full dataset had not yet been collected).

Many further improvements are possible. For instance, we could determine the 3D position of detected objects, even when they are occluded, and propagate those estimates through the rest of the model using the estimated foreground/background labels. More work is also needed to make the models more visually appealing, such as performing texture synthesis to fill in occluded regions. Finally, quantitative depth estimates, such as those provided by Saxena et al. [**118**], or 3D scene priors [**55**] could be used to guide the reconstruction process.

[1] In sorting through the images, we listed about 95 of 250 images as accurate enough for snapshots or videos to include as positive results, though some of these many have errors in a small part of the scene or the image itself might be insufficiently attractive.

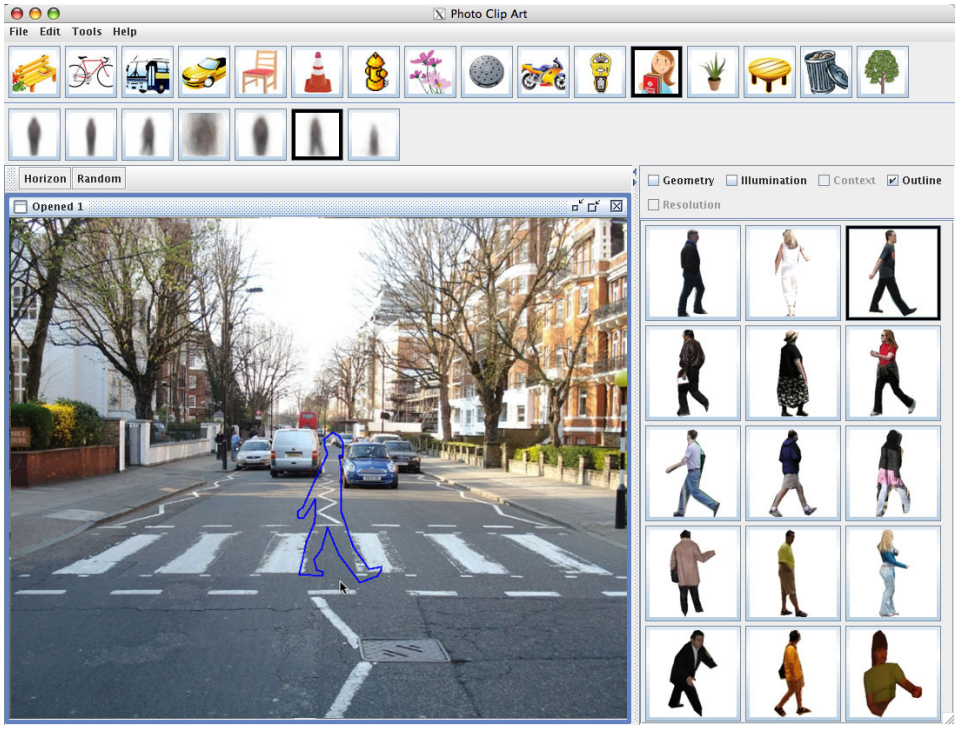


Figure 7.7: User Interface for our system. The two-level menu on the top panel allows users to navigate between different classes and subclasses of the photo clip art library. The side panel on the right shows the top matched objects that can be inserted into the scene.

3. Object Pasting

Many artists and designers frequently need to insert objects into a photograph to improve the realism or to illustrate a particular concept. Normally, such object pasting is an extremely painstaking process, as there are issues of segmentation, lighting, sizing, and shadowing among others. With Lalonde and others [**75**], we developed a system that finds objects that will match a user-provided image (according to lighting) and allow the user to drag-and-drop a selected object into the scene, automatically blending and segmenting the object, resizing it according to the true 3D height, and transferring the shadow from the original image.

Our method has two phases. In the first offline phase, we automatically recover the camera viewpoint based on labeled objects in the LabelMe dataset [**116**], giving us the true height of thousands of objects in a wide variety of object classes. We also compute statistics for matching based on lighting condition and scene geometry to help determine whether an object from one image is likely to look natural in another image. In the second online phase,

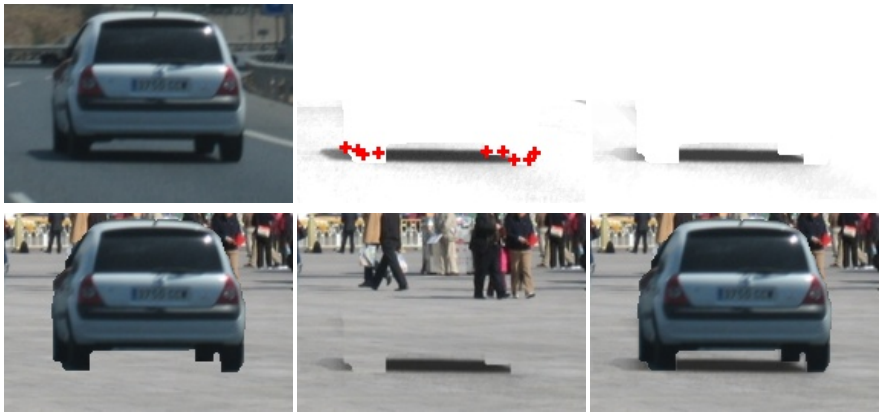


Figure 7.8: Shadow Transfer. On the top row, we show the source image (left) for a car with initial (center) and final (right) shadow estimates. Our initial estimate, simply based on intensity relative to surrounding area, is refined by enforcing that the shadow is darker closer to the contact points (red '+'). On the bottom row, we show the background patch with the object (left), with the matted shadow (center), and the combination (right), before blending.

the user provides an image and its horizon line and inserts objects into her photograph using our intuitive user interface (Figure 7.7). Based on viewpoint and object height, our system automatically resizes the object as it is dragged through the scene. Once the user selects a position, the system refines the segmentation, performs blending, transfers the shadow from the original image (Figure 7.8), and handles occlusions with other inserted objects. In Figure 7.9, we show two example results.

Figure 7.9: Object pasting results. On the top, we take a conceptual model of the new Gates building at Carnegie Mellon and add real people, making the model look more natural. On the bottom, we show an example of photorealistic storyboarding, creating an accident scene out of an empty intersection.

Figure 7.10: Several of the regions thought most likely to correspond to "solid" foreground regions from our Geometric Context dataset.

4. Object Discovery

Unsupervised visual learning from large datasets is a growing field in computer vision. For instance, Russell et al. [**115**] propose that correctly segmented objects will be more similar than incorrectly segments and, based on that, suggest a method for automatically extracting groups of segmented objects from a large dataset and classifying them. Such algorithms have had some success in constrained datasets, but would greatly improve with better segmentation algorithms and grouping that takes into account physical characteristics of the objects.

Using our occlusion reasoning algorithm, we can often obtain good segmentations of entire objects or groups of objects that could serve as a basis for object discovery. As a proof-of-concept, we show several examples of regions that are likely to be foreground objects in Figure 7.10 and Figure 7.11. These are generated simply simply by ranking regions according to the probability that all boundaries are foreground times the probability that the region is "solid" or "porous". In addition to the object boundaries, it is also possible to

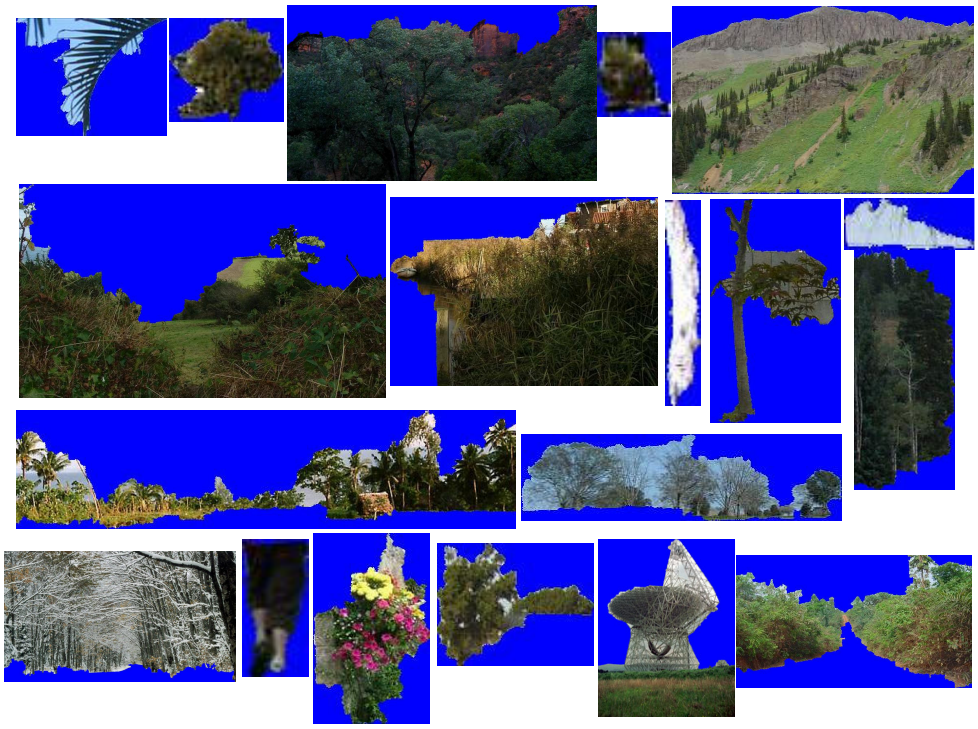


Figure 7.11: Several of the regions thought most likely to correspond to "porous" foreground regions from our Geometric Context dataset.

estimate the 3D object sizes and the contexts in which they are found (e.g., on the ground, or in the background just below the sky). Such information would be of great use when grouping objects into classes, since many objects have a limited physical size range and tend to appear in particular geometric contexts.

5. Robot Path Planning

Mobile robot path planning requires a geometric representation of the environment, often represented as a 2.5D map. Typically, this map is obtained using Laser Radar (LADAR) or passive stereo vision sensors. The limited range and acquisition methods of these sensors provide an incomplete or sparsely sampled map, which leads to poor path planning. With Bart Nabbe, we show that our spatial layout representation provides a dense, long-range environment map, allowing improved path planning [**94**].

Approximating the terrain as locally planar, we can recover a navigation map from the estimated surface layout using a simple algorithm. In each image column, we trace from bottom to top until the "support" label confidence drops below 0.5 (indicating an obstruction). Each pixel is then projected into a cell of an overhead navigation map using a perspective projection and the known camera parameters. The cell is marked with traversability value (0 to 1, where 0 indicates high confidence of traversability), which is set simply by one minus the "support" geometric label confidence of the pixel. No assumptions are made about cells that are occluded by an obstacle.

In Figure 7.12, we show several examples of navigation maps that are each recovered from one image using our surface layout representation and the simple algorithm described above. In an extension of the dissertation work of Nabbe [**93**], we are able to plan efficient paths using these maps. As an example, we performed a field experiment, in which our robot navigated around an obstacle (a large tent) on the Carnegie Mellon campus (Figure 7.13). Based on an initial observation, the robot found a clear path to the left of the obstacle. Upon approaching the left of the obstacle, the robot made another observation, updated its navigation map, and continued to its goal.

Failures can be attributed to poor surface layout estimation or highly non-planar terrain. For instance, in Figure 7.13, the sidewalk is mistaken as vertical in the initial observation, causing the region to be incorrectly labeled as non-traversable in the navigation map. Such mistakes will become increasingly rare as our surface layout estimation improves and can be rectified from subsequent observations. By incorporating additional information from stereo or LADAR sensors, we can relax our assumptions of planarity and become more robust to layout error while benefiting from the range and completeness of vision-based

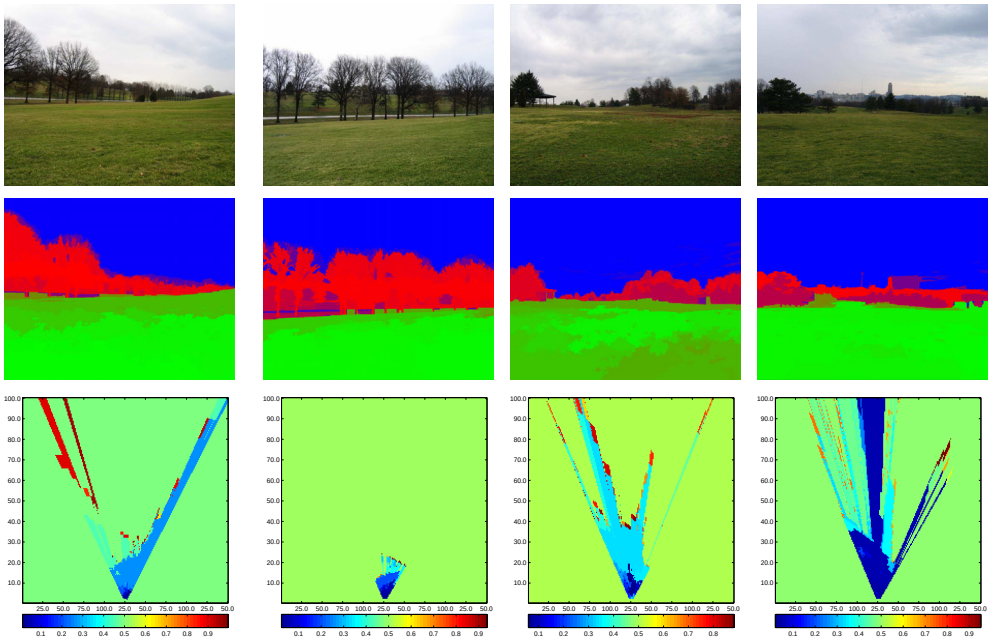


Figure 7.12: Geometric labeling and navigation maps for example unstructured environments. We show top: the original image, center: the estimated geometric class confidences for “support”, “vertical”, and “sky” as intensity in the green, red, and blue channels, respectively, and bottom: the corresponding navigation maps with 0 denoting traversable terrain and 1 representing an obstacle. Note that the range computed from our surface layout is not limited to the maximum range displayed in the range maps.

navigation. Overall, our experiments have shown that even error prone unreliable data from monocular vision can aid significantly to improve long range navigation performance by allowing to plan a path past the traditional sensing horizon.

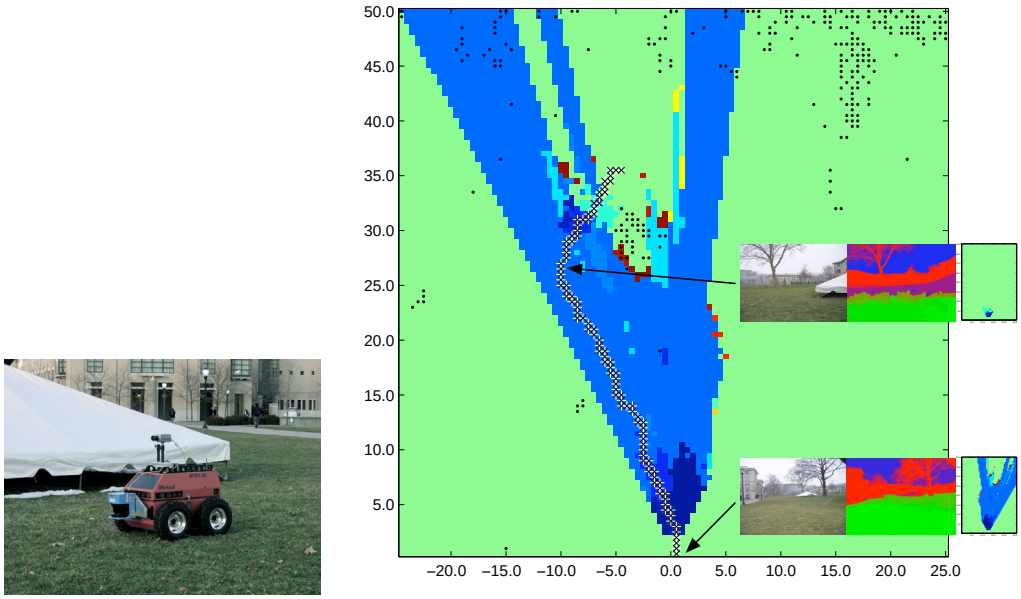


Figure 7.13: Field experiment of robot navigation using surface layout. On the left, we show our robot equipped with a camera (1024x768 resolution images), a SICK laser range finder, and a GPS and laser ring gyro for localization. On the right, we show navigation maps estimated from two observations and the final map (in meters). For comparison, we plot the readings from the laser range finder as black dots.

CHAPTER 8

Discussion

The time has come, the Walrus said,
To talk of many things:
Of shoes–and ships–and sealing-wax–
Of cabbages–and kings–
And why the sea is boiling hot–
And whether pigs have wings.
Lewis Carol (1872), *Through the Looking-Glass and What Alice Found There*

Our work is just one small step toward a 3D understanding solution. Much remains to be done, both in improving and extending our current framework and in developing a broader understanding that combines visual experience with semantic knowledge and analytical reasoning to improve interaction in the world. In this final chapter, we discuss the role of context in computer vision, future directions for recovery of spatial layout, and further steps toward scene understanding.

1. Context

A continuing theme of our work is that 3D spatial context is an important aspect of scene understanding. Here, we give our views on several issues of context: Is context helpful? Should context be implicit or explicit? Should contextual reasoning be 2D or 3D?

We define context as visual sensing or semantic knowledge that is not directly produced by the object itself. Such context could have direct representations, such as when a road label is assigned to the region under a car, or an implicit representation, such as when an object detector uses image features from outside the object itself. The context could also be

largely semantic, such as the knowledge that cows rarely wander off into the ocean or that people have a certain height range.

The Need for Context. Few would argue that context has no role in computer vision. Clearly, the world is more than a random jumble of objects, and many psychophysical studies (for instance, [**46, 53**]) show that humans make great use of contextual cues. In this dissertation, we show that simple contextual models can bring great improvements to tasks such as object detection, even in fairly unconstrained environments (our Geometric Context dataset). The extent to which contextual reasoning can improve segmentation, recognition, and tracking has hardly been explored in computer vision. We have described only a few contextual cues, but many more exist.

Arguments against specific uses of context often concern the generality of contextual models or the need to for vision to work even in rare situations. Our own work is susceptible to this criticism, since we make an assumption of a single ground plane, which can cause our contextual models to rule out true detections that have overwhelming low-level evidence. It is important for any use of context to allow for surprises. For instance, the single ground plane assumption could be made more flexible by assigning a non-zero probability to pedestrians that are not on the ground plane. Such flexible contextual models would then allow the system to detect when an unexpected situation occurs.

An Argument for Explicit Contextual Reasoning. Wolf and Bileschi [**151**] argue that explicit, high-level models of context are unnecessary. In the supporting experiment, a comparison is made between detectors that take estimated labels of the surrounding area as input and those that use only low-level image features of the surrounding area. Wolf and Bileschi show that the implicit contextual use of low-level features works at least as well as the more elaborate high-level approach of multi-stage detection.

Though Wolf and Bileschi make an interesting point, we argue that high-level contextual models allow simpler object models and, consequently, smaller training sets because they represent more directly the underlying structure of the world. For example, cars, buses, and motorcycles are all typically found on the road, and an object detector would benefit from knowing whether the area underneath these vehicles looks like a road surface. Decoupling the appearance-based detection of the vehicles and the road has several benefits: 1) the road model is learned once (instead of implicitly learned for each object; 2) the high-level

representation is more compact allowing more sophisticated reasoning (for example, a high confidence bus detection increases road likelihood, providing additional evidence for a distant car); and 3) a person working on a car detector can benefit from the road detection efforts of another and vice versa.

2D vs. 3D Context. We believe that contextual models should acknowledge the third dimension. Objects live and interact in the world, obeying forces such as gravity and collision in a 3D space. Contextual reasoning should, therefore, reflect a sense of depth. For instance, it is more accurate to say that a car is supported by the road than that the car is always above the road in the image. From a higher viewpoint it is quite possible to see the road above the car. Furthermore, modeling a computer mouse as being a certain number of pixels below and to the right of the monitor is not very powerful, as such measurements will vary wildly with viewpoint changes. Instead, it would be better to say that a mouse typically lies on the same desk as the monitor, within one or two feet of it.

However, the researcher planning to use 3D spatial context should remember that measurement error occurs in 2D. Depths computed from monocular cues are not reliable at great distances. For instance, a single-pixel error in measuring the position of a car will be insignificant at close range but may increase the depth estimate by several meters if the car is far away. This can be partially accounted for by measuring depth error in log space, as is suggested by Saxena et al. [**118**]. It is also possible to apply spatial reasoning that is inherently 3D without actually projecting back into world coordinates. For example, given the viewpoint, sizes of objects at known heights can be estimated without knowledge of depth (as we have shown). As a general rule, if 2D reasoning sufficiently models a relationship, there is no need to estimate depth or develop a more convoluted 3D model. For instance, 3D models of human pose are unlikely to greatly improve detection, as the 3D information itself is unreliable and 2D models are nearly as expressive and concise.

In summary, contextual cues such as object size and scene layout should reflect the inherent three-dimensionality of the world, but simpler 2D models are preferred when they are equally expressive.

2. Future Directions

In this dissertation, we proposed a set of methods to recover a rough spatial layout of the scene. We hope that others will build on our ideas and other related work in the community to provide more precise and complete interpretations. Here, we propose several ideas for future work that extend our current framework.

Designing Better Classifiers. It is likely that much more accurate surface and occlusion classifiers can be developed with additional effort in devising a good feature set, trying various classification methods, and increasing the number of training examples. Over the past ten years, face and pedestrian detection has improved dramatically, mostly due to such an engineering effort. Such gains are probably also possible in our algorithms, as we are among the first to attempt image-based classification of 3D surfaces and occlusions.

Improving Spatial Layout. Our current representation of spatial layout is quite limited in that surfaces are approximated either as horizontal or vertical. In many of our algorithms, we make the strong assumption of a single ground plane. Being able to represent arbitrary orientations and multiple supporting surfaces would improve reconstructed models and allow more sophisticated reasoning. Also, our occlusion reasoning ignores folds (e.g., corners of buildings) and supporting surfaces that are part of an object. After finding object boundaries, it may be necessary to refine geometry estimates with closer inspection.

Using Motion and Stereo Cues. In this dissertation, we focus on monocular cues, in part because images are more ubiquitous and easier to manage than videos or stereo pairs, but mostly out of the desire to replicate the amazing human ability to understand the 3D scene from an image. Of coarse, when additional information is available, it should be used. Stereo cues are likely to improve close-range surface estimation and occlusion reasoning. For instance, Saxena et al. [**119**] find that stereo cues are better than monocular cues at estimating depth in the range of 2 to 9 meters, while monocular cues are better outside of that range. Using both types of cues together produces the best performance overall. Optical flow and parallax can also be used to provide better segmentations and to estimate the 3D structure of close and mid-range surfaces. Stereo and motion will be especially helpful to determine more precise 3D models of nearby objects.

Recognizing More Objects. We have incorporated car and pedestrian detection into our framework to demonstrate the synergy between object detection and spatial layout estimation. The ability to detect more objects would improve spatial layout. For instance, detected windows in a city scene or tree trunks in a forest scene could provide a good sense of depth and layout, as well as providing an indication of the viewpoint and camera parameters. Identifying large surfaces in the scene, such as roads, sidewalks, dirt paths, grass, buildings, and water, could provide much additional information for contextual reasoning.

Reasoning Among Objects. In our current models, objects depend on each other only through the viewpoint. However, it is also possible to model the relationships among the objects themselves. For instance, cars are often parked in a line on the side of the road. Pedestrians often walk in small groups on the sidewalk. Buildings line the edges of the sidewalk. In more rural scenes, relationships among objects are weaker but they still exist. For instance, cows tend to graze in herds on a plain. In the past, these contextual relationships were modeled only as the likelihood of object appearance in a scene (e.g., [**92**]) or restricted to 2D relationships in the image, such as road below car below building (e.g., [**73, 140**]). With spatial layout, reasoning can occur within the 3D space of the scene, providing more powerful contextual models for objects. Specific contextual models (or schemas) may be developed for particular scenes. For instance, reasoning within a city scene and a forest scene is likely to be quite different.

3. Toward Scene Understanding

Spatial layout is only one aspect of scene understanding. Here, we outline some of the other large challenges and speculate on how to proceed. We discuss three broad problems: acquisition and use of visual experience, reasoning with real-world knowledge, and task-driven vision. Humans use years of visual experience to simplify tasks and focus on less familiar situations. With the enormous quantities of digital imagery now available, we can finally begin attempts to provide similar levels of experience to computers. No amount of experience can obviate the need for careful reasoning to disambiguate aspects of the scene or to observe new phenomena. Such reasoning will require spatial understanding to determine how objects and surfaces interact in the scene. Finally, sensing should eventually

be coupled with action, providing a context for sensing and a feedback mechanism for learning.

3.1. Visual Experience

By most estimates, human visual development requires several months to attain full functionality and continues throughout childhood. In contrast, computer vision algorithms typically begin with a clean slate and are expected to learn to recognize objects based on clever engineering and, at most, a few thousand images. Such limited training sets are a severe handicap.

With the ubiquity of digital cameras, people are taking billions of photographs every month. Many of these photos are publicly available – every day one million images are uploaded to Flickr, a photo sharing website. We may soon have the entire visual world on a hard drive. We can use this massive amount of data to more fully learn the underlying structure of our visual world. Work by Hoiem et al. [**52**] shows that even a very poor approximation of the statistics of natural images (distributions of pairs of wavelet coefficients) leads to improved object recognition from few examples. Hays and Efros [**42**] fill large holes in images by finding similar regions in a database of millions of images. This work demonstrates that, while the space of all images is infinite, the space of natural images is finite and can possibly be represented with currently available image databases.

Simple analysis of millions of images could yield non-parameteric representations of image patches and correlations among patches. Local expectations could be conditioned on the global scene or on the surrounding area. It will be important to effectively match scenes or surrounding surfaces in a way that makes the context effective. In human vision, scene matching seems to correspond to a general sense of content and spatial layout that can be recovered from global image statistics, rather than semantic categories such as "kitchen" or "birthday party" that require close inspection to confirm. These global scene representations could be derived from the "gist" of Oliva and Torralba [**102**] and from the spatial layout work presented in this dissertation. Our spatial layout representations may be especially effective for matching more local surrounding areas, such as a table-top or a patch of sidewalk near a building, as it provide more localized and detailed spatial information than the gist.

Representing and acquiring visual experience is only part of the challenge; using it effectively is difficult. One possibility is to identify coherent regions based, not on color or intensity differences, but on how commonly parts of the image are seen together. For instance, a window and a patch of bricks look nothing alike, yet we feel that they belong together. Such experience could also be used to provide a prior for image appearance as in the work by Hoiem et al. [**52**]. This would avoid having to implicitly learn a background model every time a new object is trained and may allow recognition from fewer examples. Further, visual synonyms that commonly occur next to the same patches could be discovered, improving generalization. Finally, it would be possible to create an interactive learning system in which the trainer presents a single example followed by a cycle of proposed objects in the database and reinforcement.

3.2. Explaining the Unexpected with Real-World Knowledge

Every day, we encounter many events that, while not abnormal, are not wholly expected, such as a person crossing the road or dismounting from a bicycle. We could try learn a comprehensive probabilistic model of all possible events and appearances, but such an effort is doomed because rare events are the most difficult to model empirically.

Instead, we should seek to come up with an explanation of the unexpected that is *plausible*, according to our understanding of the world. For instance, the strange pose of the person dismounting from a bicycle can be explained by the presence of a nearby bicycle and knowledge of the manner in which people use bicycles for transportation. In applying real-world knowledge to images, it will be crucial to reason within the true 3D space of the world, rather than the 2D space of the image, which is a mere accident of projection. We believe that the work described in this dissertation serves as a good first effort toward this end, providing approximate recovery of height and depth and coarse occlusion reasoning.

Further development will require more detailed notions of spatial layout. For instance, identifying a cluttered tabletop or a shelf as a supporting surface is important for everyday activities. Such reasoning sometimes requires closer inspection of objects and other times simply requires more extensive experience. It is also important to renew the effort toward integrating computer vision into the larger artificial intelligence field. Key problems will include knowledge representation, spatial reasoning, inference of intent, and coordination.

3.3. Focusing on the Task

Understanding is proven only through appropriate action. If we want to have computer vision algorithms that truly understand the scene, they must be part of a larger system that interacts with the scene or provides relevant analysis to a human operator. Tasks such as recognition, segmentation, and depth recognition should eventually be evaluated in terms of the improvement that they give to a desired activity, such as security analysis, navigation, or cleaning the house. Such activities provide a natural context for tasks, such as recognition, that are typically pursued in the abstract. For example, a typical person walking down the street is unlikely to care if someone is standing on the balcony across the way. If the person goes unnoticed, no harm will come of it. On the other hand, a security agent protecting a politician would inspect all potential threats. In this example, the pedestrian can rely on context heavily and focus on closer obstacles, but the security agent must treat all possibilities as plausible. Recovery of spatial layout is especially important when a physical agent needs to interact with the world. Spatial awareness of surroundings, localization of possible supporting surfaces, and detection of obstacles are important faculties for almost any undertaking.

A large barrier to development of task-based vision is the hassle of dealing with a physical system (such as a robot). An alternative is to apply vision to simulations that are approaching photo-realism. For instance, video games could be used to provide a ground truth for scene geometry and viewpoint recovery. Games could also provide a good environment to develop ideas of visual attention [**106**] or to integrate visual sensing with semantic reasoning. Physical simulators would allow software agents to manipulate an artificial world and learn how surfaces interact and appear to change with movement. Of course, utility in the real world should be the ultimate goal of computer vision, but a broad sketch of complete scene understanding system could be developed through simulation.

Conclusion

In this dissertation, we have taken a good first step toward recovering a coarse spatial layout of the scene and using it to improve object recognition. We hope that our work will inspire others to develop more advanced spatial and contextual models that will provide a better understanding of the 3D scene.

We conclude with a statement of the philosophy that has driven our research:
The image is but a projection of a scene, which is itself subject to a strict structure imposed by nature and man. To interpret an image, we must tease out that underlying structure while acknowledging the third dimension. We cannot possibly enumerate all the constraints of the world but instead must learn those constraints and apply statistical reasoning to find the most plausible solutions. The keys to success in a statistical algorithm are redundancy, procrastination, and integration. Use all available cues. Avoid hard decisions wherever possible. Iterate.

REFERENCES

[1] N. Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. *PAMI*, 18(12), 1996.

[2] A. Amir and M. Lindenbaum. A generic grouping algorithm and its quantitative analysis. *PAMI*, 20(2), 1998.

[3] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *Proc. POCV*, 2006.

[4] A. Barbu and S. Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *PAMI*, 27(8):1239–1253, 2005.

[5] H. Barrow and J. Tenenbaum. Recovering intrinsic scene characteristics from images. In *Comp. Vision Systems*, 1978.

[6] H. Barrow and J. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *Artificial Intelligence*, 17:75–116, 1981.

[7] I. Biederman. On the semantics of a glance at a scene. In M. Kubovy and J. R. Pomerantz, editors, *Perceptual Organization*, chapter 8. Lawrence Erlbaum, 1981.

[8] M. J. Black and D. J. Fleet. Probabilistic detection and tracking of motion discontinuities. *IJCV*, 38(3):231–245, 2000.

[9] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 23(11):1222–1239, 2001.

[10] R. Brooks, R. Greiner, and T. Binford. Model-based three-dimensional interpretation of two-dimensional images. In *IJCAI*, 1979.

[11] L. Cao, J. Liu, and X. Tang. 3D object reconstruction from a single 2D line drawing without hidden lines. In *ICCV*, 2005.

[12] E. Chen. QuickTime VR - an image-based approach to virtual environment navigation. In *ACM SIGGRAPH 95*, pages 29–38, 1995.

[13] K. Chua and M. Chun. Implicit scene learning is viewpoint dependent. *Perception & Psychophysics*, 65(1):72–80, 2003.

[14] R. Cipolla, D. Robertson, and E. Boyer. Photobuilder – 3D models of architectural scenes from uncalibrated images. In *IEEE Int. Conf. on Multimedia Computing and Systems*, volume I, pages 25–31, June 1999.

[15] M. Clowes. On seeing things. *Artificial Intelligence*, 2(1):79–116, 1971.

[16] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and Bregman distances. *Machine Learning*, 48(1–3), 2002.

[17] J. Coughlan and A. Yuille. Manhattan world: orientation and outlier detection by Bayesian inference. *Neural Computation*, 15(5), 2003.

[18] T. Cour, F. Benezit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *CVPR*, 2005.

[19] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *IJCV*, 40(2), 2000.

[20] J. E. Cutting and P. M. Vishton. Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth. In W. Epstein and S. Rogers, editors, *Perception of Space and Motion*, pages 69–117. Academic Press, San Diego, 1995.

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

[22] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *ACM SIGGRAPH 1996*, pages 11–20, 1996.

[23] E. Delage, H. Lee, and A. Y. Ng. A dynamic Bayesian network model for autonomous 3D reconstruction from a single indoor image. In *CVPR*, 2006.

[24] S. Draper. The use of gradient and dual space in line-drawing interpretation. *Artificial Intelligence*, 17:461–508, 1981.

[25] R. Duda and P. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, 1972.

[26] J. Elder and S. Zucker. Computing contour closure. In *ECCV*, 1996.

[27] M. R. Everingham, B. T. Thomas, and T. Troscianko. Head-mounted mobility aid for low vision using scene classification techniques. *Int. J. of Virt. Reality*, 3(4), 1999.

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

[29] D. A. Forsyth, J. L. Mundy, A. Zisserman, and C. A. Rothwell. Using global consistency to recognise euclidean objects with an uncalibrated camera. In *CVPR*, 1994.

[30] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2), 2000.

[31] J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, Boston, USA, 1950.

[32] J. Gibson. The perception of surface layout: A classification of types. Unpublished "Purple Perils" essay, November 1968.

[33] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *ACM SIGGRAPH 96*, pages 43–54, 1996.

[34] C. V. Gottesman. Viewpoint changes affect priming of spatial layout. *Journal of Vision*, 3(9), 2003.

[35] M. Greienhagen, V. Ramesh, D. Comaniciu, and H. Niemann. Statistical modeling and performance characterization of a real-time dual camera surveillance system. In *CVPR*, 2000.

[36] C. Guo, S. Zhu, and Y. N. Wu. Towards a mathematical theory of primal sketch and sketchability. In *ICCV*, 2003.

[37] A. Guzman. Computer recognition of three-dimensional objects in a visual scene. Technical Report MAC-TR-59, MIT, 1968.

[38] F. Han and S. Zhu. Bayesian reconstruction of 3D shapes and scenes from a single image. In *Int. Work. on Higher-Level Know. in 3D Modeling and Motion Anal.*, 2003.

[39] F. Han and S. Zhu. Bottom-up/top-down image parsing by attribute graph grammar. In *ICCV*, 2005.

[40] A. Hanson and E. Riseman. VISIONS: A computer system for interpreting scenes. In *Computer Vision Systems*, 1978.

[41] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2nd edition, 2004.

[42] J. Hays and A. A. Efros. Scene completion using millions of photographs. In *ACM SIGGRAPH 2007*.

[43] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *CVPR*, 2004.

[44] L. Herault and R. Horaud. Figure-ground discrimination: A combinatorial optimization approach. *PAMI*, 15, 1993.

[45] T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimization. In *Proc. UAI*, 2003.

[46] H. Hock, L. Romanski, A. Galie, and C. Williams. Real-world schemata and scene recognition in adults and children. *Memory & Cognition*, 6:423–431, 1978.

[47] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. In *ACM SIGGRAPH 2005*.

[48] D. Hoiem, A. A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005.

[49] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006.

[50] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 75(1):151–172, 2007.

[51] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. In *ICCV*, 2007.

[52] D. Hoiem, R. Sukthankar, H. Schneiderman, and L. Huston. Object-based image retrieval using the statistical structure of images. In *CVPR*, 2004.

[53] A. Hollingworth and J. M. Henderson. Does consistent scene context facilitate object perception? *Journ. of Experimental Psychology: General*, 127(4):398–415, 1998.

[54] Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *ACM SIGGRAPH*, pages 225–232, 1997.

[55] J. Huang, A. B. Lee, and D. Mumford. Statistics of range images. In *CVPR*, 2000.

[56] M. Hucka. *Approximate Spatial Layout Processing in the Visual System: Modeling Texture-Based Segmentation and Shape Estimation*. PhD thesis, Computer Science and Engineering, University of Michigan, 1998.

[57] D. Huffman. Impossible objects as nonsense sentences. *Machine Intelligence*, 6:295–323, 1971.

[58] D. Huffman. Realizable configurations of lines in pictures of polyhedra. *Machine Intelligence*, 8:493–509, 1977.

[59] D. Jacobs. Robust and efficient detection of convex groups. In *CVPR*, 1993.

[60] R. Jain and J. Aggarwal. Computer analysis of scenes with curved objects. *Proc. of the IEEE*, 67(5):805–812, May 1979.

[61] S. G. Jeong, C. S. Kim, D. Y. Lee, S. K. Ha, D. H. Lee, M. H. Lee, and H. Hashimoto. Real-time lane detection for autonomous vehicle. In *ISIE*, 2001.

[62] I. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *PAMI*, 23(10):1075–1088, 2001.

[63] T. Kanade. A theory of the Origami world. *Artificial Intelligence*, 13:279–311, 1980.

[64] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *Artificial Intelligence*, 17:409–460, 1981.

[65] H. Kang, S. Pyo, K. Anjyo, and S. Shin. Tour into the picture using a vanishing line and its extension to panoramic images. In *Proc. EuroGraphics*, pages 132–141, 2001.

[66] J. J. Koenderink. Pictorial relief. *Phil. Trans. of the Roy. Soc.*, pages 1071–1086, 1998.

[67] J. J. Koenderink, A. J. V. Doorn, and A. M. L. Kappers. Pictorial surface attitude and local depth comparisons. *Perception and Psychophysics*, 58(2):163–173, 1996.

[68] S. Konishi and A. Yuille. Statistical cues for domain specific image segmentation with performance analysis. In *CVPR*, 2000.

[69] J. Kosecka and W. Zhang. Video compass. In *ECCV*. Springer-Verlag, 2002.

[70] I. Kovacs and B. Julesz. A closed curve is much more than an incomplete one: Effect of closure in figure-ground discrimination. *Proc. Nat'l Academy of Science USA*, 90, 1993.

[71] N. Krahnstoever and P. R. S. Mendona. Bayesian autocalibration for surveillance. In *ICCV*, 2005.

[72] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *ICCV*, 2003.

[73] S. Kumar and M. Hebert. A hierarchical field framework for unified context-based classification. In *ICCV*, 2005.

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

[75] J. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. In *ACM SIGGRAPH 2007*.

[76] Y. Leclerc and M. Fischler. An optimization-based approach to the interpretation of single line drawings as 3D wire frames. *IJCV*, 9(2), 1992.

[77] T. Leung and J. Malik. Contour continuity in region based image segmentation. In *ECCV*, 1998.

[78] T. Leung and J. Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001.

[79] M. Levoy and P. Hanrahan. Light field rendering. In *ACM SIGGRAPH 1996*, pages 31–42, 1996.

[80] Y. Li, J. Sun, C. Tang, and H. Shum. Lazy snapping. *ACM Trans. on Graphics*, 23(3):303–308, 2004.

[81] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating architectural models from images. In *Proc. EuroGraphics*, volume 18, 1999.

[82] H. Lipson and M. Shpitalni. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design*, 28(8), 1996.

[83] D. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer, 1985.

[84] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *PAMI*, 25(4), April 2003.

[85] J. Malik. Interpreting line drawings of curved objects. *IJCV*, 1(1):73–103, 1987.

[86] T. Marill. Emulating the human interpretation of line-drawings as three-dimensional objects. *IJCV*, 6(2), 1991.

[87] D. Marr. *Vision*. Freeman, San Francisco, 1982.

[88] D. Martin, C. Fowlkes, and J. Malik. Learning to find brightness and texture boundaries in natural images. *NIPS*, 2002.

[89] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, volume 2, pages 416–423, July 2001.

[90] J. McDermott. Psychophysics with junctions in real images. *Perception*, 33(9):1101–1127, 2004.

[91] K. Murphy. The Bayes Net Toolbox for Matlab. In *Computing Science and Statistics*, volume 33. 2001.

[92] K. Murphy, A. Torralba, and W. T. Freeman. Graphical model for recognizing scenes and objects. In *NIPS*. 2003.

[93] B. Nabbe. *Extending the Path-planning Horizon*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2005.

[94] B. Nabbe, D. Hoiem, A. A. Efros, and M. Hebert. Opportunistic use of vision to push back the path-planning horizon. In *Proc. IROS*, 2006.

[95] A. Navot, L. Shpigelman, N. Tishby, and E. Vaadia. Nearest neighbor based feature selection for regression and its application to neural activity. In *NIPS*. 2006.

[96] D. Nistér. *Automatic dense reconstruction from uncalibrated video sequences*. PhD thesis, Royal Institute of Technology KTH, 2001.

[97] M. Nitzberg and D. Mumford. The 2.1-D sketch. In *ICCV*. 1990.

[98] B. M. Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *ACM SIGGRAPH 2001*. ACM Press.

[99] Y. Ohta. *Knowledge-Based Interpretation Of Outdoor Natural Color Scenes*. Pitman, 1985.

[100] Y. Ohta, T. Kanade, and T. Sakai. An analysis system for scenes containing objects with substructures. In *IJCPR*, pages 752–754, 1978.

[101] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[102] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, 155, 2006.

[103] The PASCAL object recognition database collection. Website, 2005. http://www.pascal-network.org/challenges/VOC/.

[104] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[105] P. Perona and W. Freeman. A factorization approach to grouping. In *ECCV*, 1998.

[106] R. Peters and L. Itti. Beyond bottom-up: Incorporating task-dependent influences into a computational model of spatial attention. In *CVPR*, 2007.

[107] K. Pezdek, T. Whetstone, K. Reynolds, N. Askari, and T. Dougherty. Memory for real-world scenes: The role of consistency with schema expectation. *Journ. of Experimental Psychology: Learning, Memory, and Cognition*, 15(4):587–595, 1989.

[108] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 2000.

[109] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *Int. J. of Computer Vision*, 59(3):207–232, 2004.

[110] M. Pollefeys, R. Koch, and L. J. V. Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *ICCV*, 1998.

[111] A. Rabinovich, S. Belongie, T. Lange, and J. M. Buhmann. Model order selection and cue combination for image segmentation. In *CVPR*, 2006.

[112] X. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *ECCV*, 2006.

[113] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, 2003.

[114] L. Roberts. Machine perception of 3-D solids. In *OEOIP*, pages 159–197, 1965.

[115] B. C. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[116] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. Technical report, MIT, 2005.

[117] E. Saund. Logic and MRF circuitry for labeling occluding and thinline visual contours. In *NIPS*. Cambridge, MA, 2006.

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

[119] A. Saxena, J. Schulte, and A. Y. Ng. Depth estimation using monocular and stereo cues. In *IJCAI*, 2007.

[120] A. Saxena, M. Sun, R. Agarwal, and A. Ng. Learning 3-D scene structure from a single still image. Unpublished manuscript, 2007.

[121] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[122] H. Schneiderman. Learning a restricted Bayesian network for object detection. In *CVPR*, 2004.

[123] E. Sharon, A. Brandt, and R. Basri. Fast multiscale image segmentation. In *CVPR*, 2000.

[124] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. PAMI*, 22(8), August 2000.

[125] K. Shoji, K. Kato, and F. Toyama. 3-D interpretation of single line drawings based on entropy minimization principle. In *ICCV*, 2001.

[126] A. Singhal, J. Luo, and W. Zhu. Probabilistic spatial context models for scene content understanding. In *CVPR*, 2003.

[127] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *PAMI*, 26(4):479–494, April 2004.

[128] S.Sarkar and P. Soundararajan. Supervised learning of large perceptual organization: graph spectral partitioning and learning automata. *PAMI*, 22(5), 2000.

[129] A. N. Stein and M. Hebert. Local detection of occlusion boundaries in video. In *BMVC*, 2006.

[130] A. N. Stein and M. Hebert. Using spatio-temporal patches for simultaneous estimation of edge strength, orientation, and motion. In *Beyond Patches Workshop at CVPR*, 2006.

[131] A. N. Stein, D. Hoiem, and M. Hebert. Learning to find object boundaries using motion cues. In *ICCV*, 2007.

[132] E. Sudderth, A. Torralba, W. T. Freeman, and A. Wilsky. Learning hierarchical models of scenes, objects, and parts. In *ICCV*, 2005.

[133] E. Sudderth, A. Torralba, W. T. Freeman, and A. Wilsky. Depth from familiar objects: A hierarchical model for 3D scenes. In *CVPR*, 2006.

[134] K. Sugihara. An algebraic approach to the shape-from-image-problem. *Artificial Intelligence*, 23:59–95, 1984.

[135] K. Sugihara. A necessary and sufficient condition for a picture to represent a polyhedral scene. *PAMI*, 6(5):578–586, September 1984.

[136] H. Tao, H. S. Sawhney, and R. Kumar. A global matching framework for stereo computation. In *ICCV*, pages 532–539, 2001.

[137] J. Tenenbaum and H. Barrow. Experiments in interpretation guided segmentation. *Artificial Intelligence*, 8(3):241–274, June 1977.

[138] A. Torralba. Contextual priming for object detection. *IJCV*, 53(2), 2003.

[139] A. Torralba. *Contextual Influences on Saliency*, pages 586–593. Academic Press / Elsevier, 2005.

[140] A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *NIPS*. 2005.

[141] A. Torralba and A. Oliva. Depth estimation from image structure. *PAMI*, 24(9), 2002.

[142] A. Torralba and P. Sinha. Statistical context priming for object detection. In *ICCV*, 2001.

[143] Z. Tu, X. Chen, A. L. Yuille, and S. C. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *IJCV*, 63(2):113–140, 2005.

[144] Z. Tu and S. Zhu. Image segmentation by data-driven markov chain monte carlo. *PAMI*, pages 657–673, May 2002.

[145] R. Vaillant and O. Faugeras. Using extremal boundaries for 3D object modeling. *PAMI*, 14(2):157–173, February 1992.

[146] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2), 2004.

[147] D. L. Waltz. Understanding line drawings of scenes with shadows. In P. Winston, editor, *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, New York, 1975.

[148] R. M. Warren and R. P. Warren. *Helmholtz on Perception: its physiology and development*. John Wiley & Sons, 1968.

[149] M. Wertheimer. Laws of organization in perceptual forms. In W. D. Ellis, editor, *A Sourcebook of Gestalt Psychology*. Routledge and Kegan Paul, 1938.

[150] C. Wheatstone. On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, pages 371–394, 1838.

[151] L. Wolf and S. Bileschi. A critical view of context. *IJCV*, 2006.

[152] Y. Yakimovsky and J. A. Feldman. A semantics-based decision theory region analyzer. In *IJCAI*, pages 580–588, 1973.

[153] A. L. Yuille. CCCP algorithms to minimize the bethe and kikuchi free energies: convergent alternatives to belief propagation. *Neural Computation*, 14(7), 2002.

[154] L. Zhang, G. Dugas-Phocion, J. Samson, and S. Seitz. Single view modeling of free-form scenes. In *CVPR*, 2001.

[155] R. Ziegler, W. Matusik, H. Pfister, and L. McMillan. 3D reconstruction using labeled image regions. In *SGP '03: Proceedings of the Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 248–259. Eurographics Association, 2003.

[156] G. Zimmerman, G. Legge, and P. Cavanagh. Pictorial depth cues: a new slant. *Journ. of the Optical Soc. of America A*, 12:17–26, Jan 1995.