International Journal of Computer Vision 75(1), 151–172, 2007
© 2007 Springer Science + Business Media, LLC. Manufactured in the United States.
DOI: 10.1007/s11263-006-0031-y

Recovering Surface Layout from an Image

DEREK HOIEM*, ALEXEI A. EFROS AND MARTIAL HEBERT
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
dhoiem@cs.cmu.edu
efros@cs.cmu.edu
hebert@ri.cmu.edu

Received April 28, 2006; Accepted December 19, 2006

First online version published in February, 2007

Abstract. Humans have an amazing ability to instantly grasp the overall 3D structure of a scene—ground orientation, relative positions of major landmarks, etc.—even from a single image. This ability is completely missing in most popular recognition algorithms, which pretend that the world is flat and/or view it through a patch-sized peephole. Yet it seems very likely that having a grasp of this "surface layout" of a scene should be of great assistance for many tasks, including recognition, navigation, and novel view synthesis.

In this paper, we take the first step towards constructing the surface layout, a labeling of the image into *geometric* classes. Our main insight is to learn appearance-based models of these geometric classes, which coarsely describe the 3D scene orientation of each image region. Our multiple segmentation framework provides robust spatial support, allowing a wide variety of cues (e.g., color, texture, and perspective) to contribute to the confidence in each geometric label. In experiments on a large set of outdoor images, we evaluate the impact of the individual cues and design choices in our algorithm. We further demonstrate the applicability of our method to indoor images, describe potential applications, and discuss extensions to a more complete notion of surface layout.

Keywords: surface layout, spatial layout, geometric context, scene understanding, context, object detection, model-driven segmentation, image understanding, multiple segmentations, object recognition

1. Introduction

Consider the photograph in Fig. 1. Modern computer vision techniques could be used to recover a lot of useful information about the image: detect people's faces, find bicycles and cars, perform optical character recognition, find shadows and recover illuminant direction, segment the image based on color, texture, or contours, and much much more. Yet, does the computer actually *understand* the scene depicted in the photograph? Can it reason about 3D relationships between objects (e.g. car on road), find walkable free-space, or capture the spatial layout of the scene? Unfortunately, the answer today is largely no.

Alas, this information, while trivial for us with but a quick glance at the photograph, is largely beyond today's computer vision algorithms. If we want to bring machine vision capabilities closer to those of a human, we need to connect the various pieces of information in the image and globally reason about the scene and its contents.

In this paper, we address a small but important component of the scene understanding problem. Our goal is to recover the rough *surface layout* of a scene, a sort of theater stage 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.

We take the first steps toward constructing this surface layout by proposing a technique to estimate the orientations of large surfaces in outdoor images. 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 (Fig. 2). 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

*Corresponding author.

Figure 1. The streets of Beijing. Scene understanding requires knowledge of the spatial layout, content, setting, and situation of the scene. In this paper, we take the first steps by estimating the surface layout of the scene, coarsely labeling the major surfaces.

subdivided into planar surfaces facing left, right or toward the camera and non-planar surfaces, either porous (e.g. leafy vegetation or a mesh of wires) or solid (e.g. a person or tree trunk).

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 (Ohta, 1985; Everingham et al., 1999; Konishi and Yuille, 2000; Singhal et al., 2003), 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. Pollefeys et al., 1998), manual labeling (Criminisi et al., 2000; Liebowitz et al., 1999), or very specific scenes (Han and Zhu, 2003), 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 estimated scene structure to compute more complex image features and using these more complex image features 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. We conclude by suggesting ways to improve and complete the surface layout and by describing potential applications.

2. Background

In the past twenty years, computer vision researchers have made tremendous progress in object detection, face recognition, structure from motion, matching, and tracking. When it comes to general understanding of the visual scene, or "seeing" as humans think of seeing, however, capabilities are virtually non-existent. Though much of the early work in computer vision focused on scene understanding, it eventually became clear that the algebraic and rule-based approaches of the day could not handle the complexity and irregularity of the world. We believe that now is the time to resume such efforts, combining intuitions from the early years of computer and human vision research with the statistical, data-driven tools and computational power of today.

2.1. *Theories of Vision*

For centuries, scholars of the body and mind have pondered the mental metamorphosis from the visual field (2D retinal image) to the visual world (our perception of 3D environment). Hermann von Helmholtz, the most notable of the 19th century empiricists, believed in an "unconscious inference", in which 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 (Warren and Warren, 1968). This inference is based on an accumulation of evidence from a variety of cues, such as the horizon, shadows, atmospheric effects, and familiar objects.

In the opening pages of his book *Perception of the Visual World* (Gibson, 1950), James Gibson declared, "The elementary impressions of a visual world are those of surface and edge." Gibson laid out 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* (Barrow and Tenenbaum, 1978), 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 (Marr, 1982):

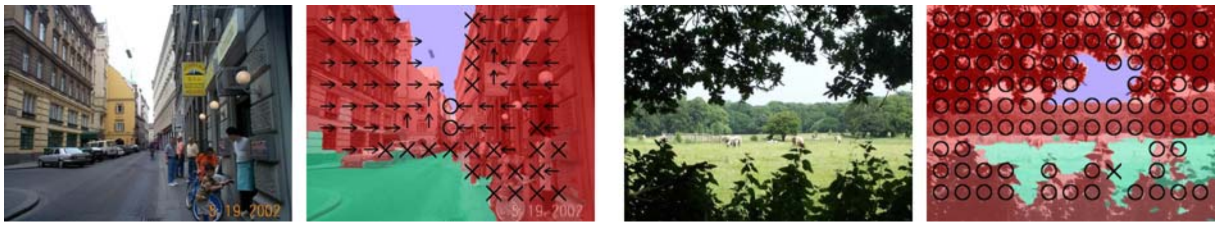


Figure 2. Surface layout. On these images and elsewhere, main class labels are indicated by colors (green=support, red=vertical, blue=sky) and subclass labels are indicated by markings (left/up/right arrows for planar left/center/right, 'O' for porous, 'X' for solid).

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.

Koenderink and colleagues (Koenderink et al., 1996; Koenderink, 1998) experimentally measured 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 provided evidence that people cannot determine the relative depth of two points unless there is some visible and monotonic surface that connects them. 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.

Irving Biederman (1981) characterized a well-organized scene as one that satisfies five relational constraints: support, interposition, probability, position, and size. Though unable to explain how these relationships are perceived, Biederman supplied convincing evidence that they are extremely valuable for scene understanding and, indeed, are heavily used in the human vision process.

Our current work has been heavily influenced by all of these ideas. Our approach shares Helmholtz' intuition and empiricist philosophy by learning models of surfaces from the "experience" of a training set and by drawing from a large and diverse set of visual cues. Our classification of an image into support, vertical, and sky regions corresponds strongly with Gibson's notions of basic surface type, though we differ from his belief in the primacy of gradient-based methods. Our surface 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 surface layout is to be used *with* the original image data, not as a substitute for it.

Overall, our work can be seen as an attempt to offer a partial solution to the spatial understanding of the scene, providing at least a computational explanation for what remains a physiological phenomenon—recovery of the 3D world from the 2D image.

2.2. *Early Computer Vision and AI*

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. While initial efforts in scene understanding focused largely on toy "blocks worlds" (Roberts, 1965; Guzman-Arenas, 1968), by the 1970s, several extremely sophisticated approaches were proposed for handling real indoor and outdoor images. For instance, Yakimovsky and Feldman (1973) 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* (Tenenbaum and Barrow, 1977) which labeled image regions, using constraint propagation to arrive at a globally consistent scene interpretation. Ohta et al. (1978) and Ohta (1985) 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 (Fig. 3(a)) and as a set of production rules in the top-down process (Fig. 3(b)). Results of applying this semantic interpretation to an outdoor image are shown on Fig. 3(c). By the late 1970s, several complete image understanding systems were being developed including such pioneering work as Brooks' *ACRONYM* (Brooks et al., 1979) and Hanson and Riseman's *VISIONS* (1978). As an example, *VISIONS* was an extremely 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, none of them were particularly successful, mainly because of the heavy use of hand-tuned heuristics which did not generalize well to new data. This, in turn, lead people to doubt the very goal of complete image

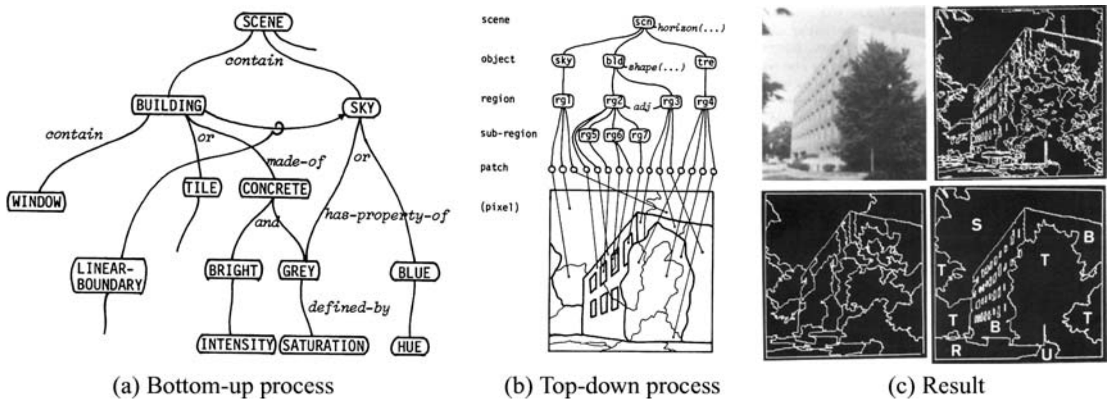


Figure 3. A system developed in 1978 by Ohta, Kanade and Sakai (1978) and Ohta (1985) for knowledge-based interpretation of outdoor natural scenes. The system is able to label an image (c) into semantic classes: S-sky, T-tree, R-road, B-building, U-unknown. Figure used with permission.

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. The advancement of learning methods in the last decade brings renewed hope for a complete scene understanding solution.

2.3. 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 et al. (2003) propose an implementation of Marr's primal sketch, and Han and Zhu (2005) describe a grammar for parsing image primitives. Tu and Zhu (2002) describe a segmentation method based on a generative image representation that could be used to sample multiple segmentations of an image. Barbu and Zhu (2005) propose a model-driven segmentation with potential applications to image parsing (Tu et al., 2005).

Oliva and Torralba (2001) 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 Torralba and Oliva (2002), used similar algorithms to estimate mean depth of the scene. These concepts have since been applied to object recognition (Murphy et al., 2003; Sudderth et al., 2005), and Sudderth et al. (2006) have demonstrated the ability to recover some depth information based on the objects in an image.

Some recent work has also attempted recovery of 3D information from a single image. Han and Zhu (2003) reconstruct wire-like objects in simple scenes. Saxena et al. (2005) present a method to learn depth from single outdoor images based on low-level features in an MRF model, and Delage et al. (2006) present a method for reconstructing indoor scenes from a single image.

We draw inspiration from Oliva and Torralba's global scene representation and the model-driven stochastic segmentation processes of Barbu, Tu, and Zhu. The depth estimation work of Saxena et al. could be used to complement our surface layout by providing evidence for the viewpoint and occluding contours.

3. 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 (e.g. material) with view-centric cues (e.g. perspective). Figure 2 shows images labeled with these geometric classes.

3.1. *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 are solid surfaces that are too steep to support an object, such as walls, cliffs, the curb sides, people, trees, or cows. The sky is simply the image region corresponding to the open air and clouds.

3.2. *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.

These subcategories are intended to provide a more detailed labeling of the vertical class in a manner that will be useful for scene understanding. For instance, consider the task of navigation. Planar surfaces often make up the boundaries of the traversable area of a scene. A vehicle may be able to drive through porous objects, but avoidance would be preferred. Solid objects are often not fastened to the ground but require avoidance. The solid vertical subclass also contains many objects that would be of interest in the object recognition task.

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.

4. 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 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 5). 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 6) to decide which to use and how to use them.

4.1. *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 4 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 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 then compute the mean (set L1 in Table 1) and 10th and 90th 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 5.1, and pixels (L6), normalized by total image area, in each segment.

4.2. *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

Table 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.

SURFACE CUES
Location and Shape
L1. Location: normalized x and y, mean
L2. Location: normalized x and y, 10th and 90th pctl
L3. Location: normalized y wrt estimated horizon, 10th, 90th pctl
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. LM filters: mean absolute response (15 filters)
T2. LM 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

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 Fig. 5, 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

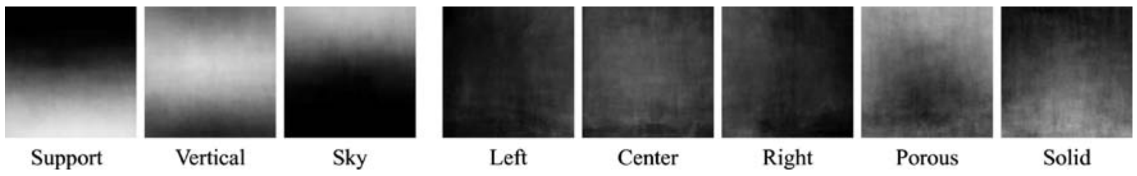


Figure 4. Likelihood (higher intensity is more likely) of each geometric class given location in the image. Location is highly discriminative for the main classes. Porous surfaces (often vegetation) tends to form a canopy around the center of the image, while solid surfaces often occur in the front-center of the image. The left/center/right likelihoods show the tendency to take photos directly facing walls or down passages.

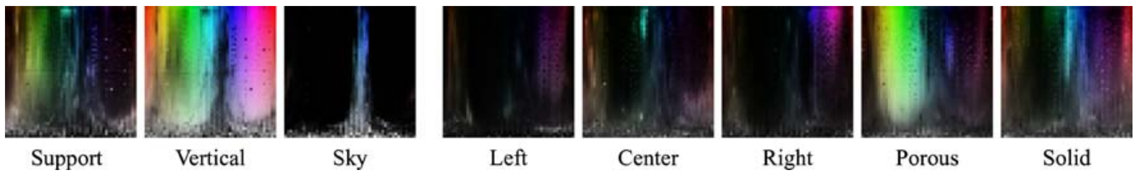


Figure 5. Likelihood of each geometric class given hue and saturation. Each image shows the given hue and saturation, with the label likelihood given by the intensity. Blue sky is easily distinguishable. More confusion exists between the support and vertical classes, though natural tones of brown, green, 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.

"greenness" of a segment to be easily extracted, while HSV allows perceptual color attributes such as hue and "grayness" to be measured.

4.3. 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 (2001). We generated the filters using publicly available code with the following parameters: 19×19 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).

4.4. Perspective

Knowledge of the vanishing line of a plane completely specifies its 3D orientation relative to the viewer (Hartley and Zisserman, 2004), 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.

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 (2002). 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 (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 (2002). 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.

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 a segment's center of mass with its center of mass of gradient magnitude.

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 the horizon estimate is often quite poor).

Note that, though little theory or experimentation supports the exact parameter values given in this subsection, our system seems to be robust to reasonable variations.

5. 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 Fig. 6).

5.1. 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

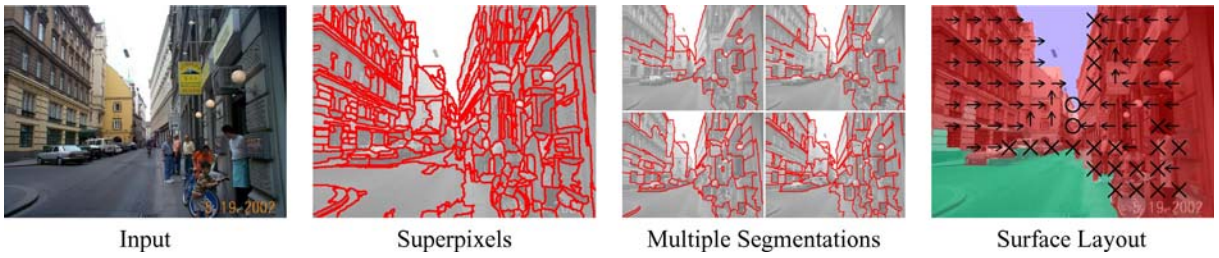


Figure 6. Surface layout estimation algorithm. From the input image, we create an oversegmentation into superpixels. We then group those superpixels into several different segmentations. The final surface layout combines estimates from all of the segmentations.

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 (Tao et al., 2001; Ren and Malik, 2003; Li et al., 2004). 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 (2004). 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 800×600 image). The superpixels, however, tend to be highly irregular in size and shape which could be disadvantageous for some applications. Alternative oversegmentation methods include the normalized cuts based method evaluated by Ren and Malik (2003) and the simple watershed algorithm, which was used effectively by Li et al. (2004). These methods produce more regular superpixels but require much more processing time or larger numbers of superpixels.

5.2. *Multiple Segmentations*

Our experiments (Section 8.2) 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 (e.g. Shi and Malik, 2000) 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 Fig. 8) 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 Fig. 7, 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. (2006) generate multiple segmentations using Normalized Cuts (Shi and Malik, 2000) while varying the size of the image and the number of segments. Other possibilities include varying the cues used for the segmentation (Rabinovich et al., 2006) or generating a hierarchical segmentation (Ahuja, 1996; Sharon et al., 2000; Arbelaez, 2006).

5.3. *Labeling*

Each segmentation provides a different view of the image. To arrive at a final conclusion, 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 in a probabilistic fashion. A segment is homogeneous if all contained superpixels have the same label. We estimate

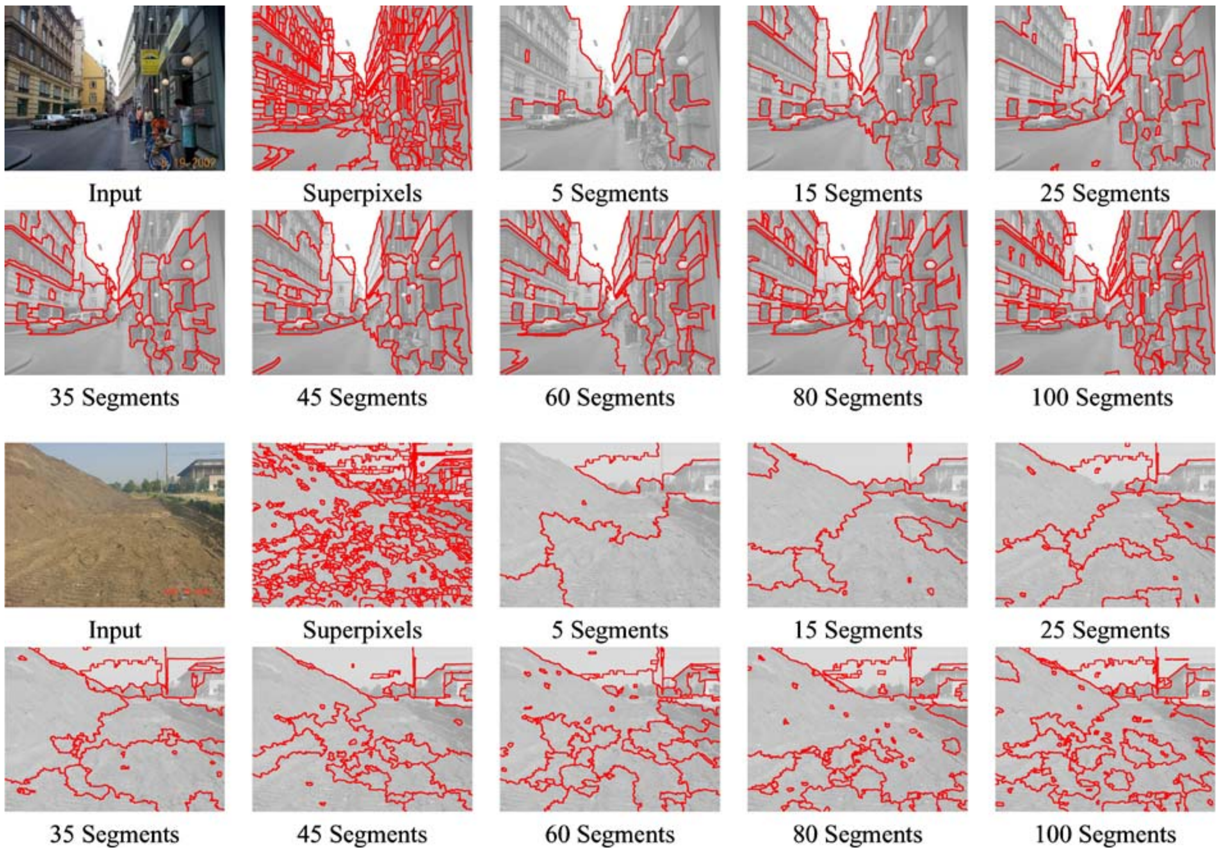


Figure 7. Examples of multiple segmentations.

```
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

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

Minimize energy: $\mathrm{E}(\mathbf{m}) = -\sum_{ij} \log \mathrm{p}_{ij}^{m_i=m_j}(1 - p_{ij})^{m_i \neq m_j}$
    Until local minimum is reached for $\mathrm{E}(\mathbf{m})$
      For each $m_i$: assign $m_i$ to maximize $\mathrm{E}(\mathbf{m})$

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

Figure 8. 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.

the homogeneity likelihood $P(s_j \mid \mathbf{I})$ based on all of the cues listed in Table 1 using boosted decision trees (in experiments, additional cues based on individual superpixel label likelihood and same-label likelihood estimates did not help). Homogeneity is very difficult to estimate from image data, 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 $P(\tilde{y}_j \mid s_j, \mathbf{I})$.

To get the label likelihood for the ith superpixel, we simply marginalize over the unique sampled segments $\{s_j\}$ that contain the superpixel:

$$P(y_i = k \mid \mathbf{I}) \propto \sum_{s_j \ni i} P(s_j \mid \mathbf{I}) P(\tilde{y}_j = k \mid s_j, \mathbf{I}) \qquad (1)$$

Here, y_i is the superpixel label and $\tilde{y}_j$ is the segment label. Since the sum is over segments containing the ith superpixel, rather than the joint space of segmentations, we believe that a small number of segmentations will provide a sufficient sample. Our experiments (Section 8.5) 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 9.1, 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.

6. Classifiers

In this paper, we use boosted decision trees for each classifier, using the logistic regression version of Adaboost Collins et al. (2002) and Friedman et al. (2000). 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 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 (Schapire and Singer, 1999) 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 Fig. 9. 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 $P(y_i = y_j \mid \mathbf{I})$ for the adjacent superpixels i and j and image data $\mathbf{I}$. The segment homogeneity classifier outputs $P(s_k \mid \mathbf{I})$ for the kth segment. We train separate classifiers to distinguish among the main classes and the subclasses of vertical. These are each learned in a one vs. all fashion.

```
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 9. Boosted decision tree algorithm using logistic regression version of Adaboost.

```
TRAINING OVERVIEW
1. For each training image:
    (a) Compute superpixels
    (b) Compute superpixel cues (Table 1)
2. Train same-label classifier (Section 6)
3. For each training image:
    (a) Produce multiple segmentations for varying $n_s$ (Section 5)
    (b) Label each segment according to ground truth
    (c) Compute cues in each segment (Table 1)
4. Train segment label classifier and homogeneity classifier (Section 6)
```

Figure 10. Outline of training procedure.

For instance, to distinguish 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".

7. Implementation

For training and testing, we gathered 300 outdoor images using Google image search for words such as "alley", "beach", "buildings", "city", "cliff", and "college". In collecting the images, we removed aerial photos and those with extreme viewpoints (i.e., we required that the horizon line be within the image). 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 300 $\times$ 240 to 1024 $\times$ 768, with varying aspects.

To provide ground truth, we first oversegmented the images into superpixels (described in Section 5.1). These superpixels, typically numbering about 500 per image, provide our atomic representation, and each was manually assigned a ground truth label by drawing polygons around regions and clicking individual superpixels. In all, about 150,000 superpixels were labeled in a time-consuming process. Labeling superpixels, instead of pixels, provides a very accurate, complete ground truth, but sometimes a superpixel will include more than one geometric class. On those rare occasions, we assign the most appropriate label. Table 2 shows the average image area of each main class and subclass in our data set.

Table 2. Average image area for each main class and subclass of vertical.

Support	Vertical	Sky
31.4%	48.0%	20.6%

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

Training and testing is performed with cross-validation. Our training algorithm is outlined in Fig. 10, and our inference algorithm in Fig. 11. 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 5.2). 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 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 the same ground truth label, the segment is assigned that same 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

```
SURFACE LAYOUT ESTIMATION
1. Image $\rightarrow$ superpixels via over-segmentation
2. Superpixels $\rightarrow$ multiple segmentations
   (a) For each superpixel: compute cues (Table 1)
   (b) For each pair of adjacent superpixels: compute same-label likelihood $P(y_i = y_j|\mathbf{I})$
   (c) Create multiple segmentations for varying $n_s$ (Section 5)
3. Multiple segmentations $\rightarrow$ superpixel labels
   (a) For each segment:
       i. Compute cues (Table 1)
       ii. For each possible label (main classes and subclasses): compute label likelihood $P(\tilde{y}_j|\mathbf{I}, s_j)$
       iii. Compute homogeneity likelihood $P(s_j|\mathbf{I})$
   (b) Compute label confidences for each superpixel: $P(y_i|\mathbf{I}) \propto \sum_{s_j \ni i} P(\tilde{y}_j|\mathbf{I}, s_j)P(s_j|\mathbf{I})$
```

Figure 11. Outline of surface layout estimation algorithm.

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.

Our current implementation differs from that of our previous work (Hoiem et al., 2005) in several ways. We replaced the texture filters, added texture histograms of the max response (T2), added explicit vanishing point cues (P8–P13), and restricted segmentations to continuous segments. The additional vanishing point cues improved subclass accuracy by about 4%. The change in texture cues accounts for most of the remaining improvement (roughly 2% and 5% for main and subclasses). The segmentation into continuous segments results in fewer scattered errors.

8. Experiments

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 comparison of varying levels of spatial support, the impact of different types of cues on accuracy, and choices of parameters for segmentation and classification.

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 8.1), the performance of our method is quite good, both quantitatively and qualitatively. Key ingredients to its success are the robust spatial support (8.2) provided by multiple segmentations and the inclusion of many types of cues (8.3). 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 (8.5) and classification parameters (8.6). Our method also easily extends to indoor images (8.7), achieving excellent accuracy despite a small training set.

8.1. Overall

In Table 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 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 Fig. 12, we plot the precision-recall curves for the main and subclasses.

In Figs. 16 and 17, we show qualitative results from our multiple segmentation method, and, in Fig. 18, we display results as confidence images for each of the surface labels. We demonstrate the generality of our geometric models in Fig. 20, where we label paintings, despite training on only real images.

8.2. Spatial Support

In Table 4, we report the average accuracy for increasing levels of spatial support, from pixels to multiple

Table 3. Confusion matrices (row-normalized) for multiple segmentation method.

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

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

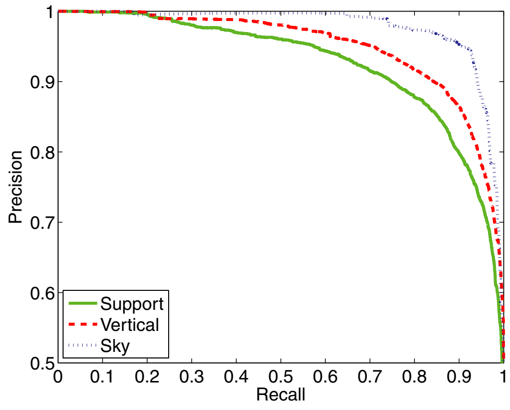
segmentations. When using only individual pixels as spatial support, based on color, filter responses, and position, the classification accuracies are 82.1% for the main classes and 44.3% for the subclasses. When using superpixels for spatial support, based on the simpler superpixel cues (L1, L6, C1–C4, T1–T2 in Table 1), the accuracies improve substantially to 86.0% and 52.9%, respectively. When using all of the cues, including perspective cues, however, accuracy improves only slightly to 86.2% and 53.5%. 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.

8.3. *Analysis of Cues*

We wish to evaluate the effectiveness of our four types of cues: location and shape, color, texture, and perspective (see Table 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



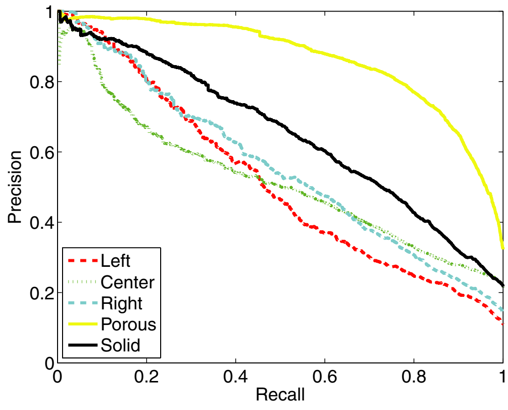


Figure 12. Precision-recall curves. Precision is the percent of declared labels (at some confidence threshold) that are true, and recall is the percent of true labels that are declared. The curve is generated by varying the confidence threshold.

Table 5. Average accuracy under different sets of cues.

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

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 5. In Fig. 19, 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 (Hoiem et al., 2005)) 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.

8.4. *Sufficiency of Data*

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 Fig. 13, providing evidence that much larger training sets would probably improve accuracy, especially for the subclasses.

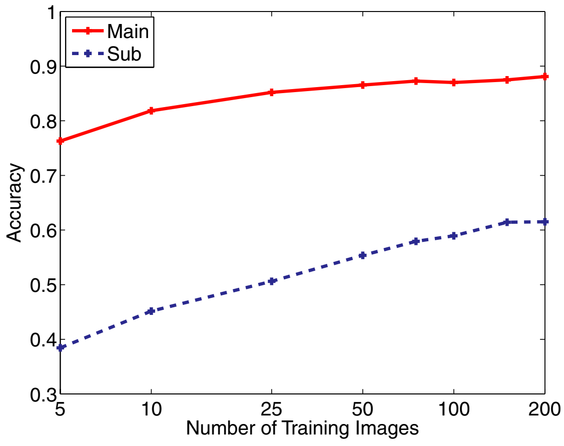


Figure 13. Accuracy while varying the number of training images. According to these trends, much larger training sets would probably result in significantly higher accuracy, especially for the subclasses.

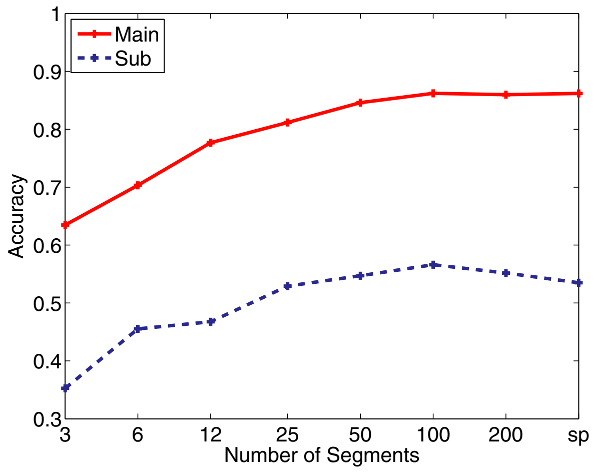
8.5. *Analysis of Segmentations*

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 Fig. 14(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 Fig. 14 (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 4, which shows that segmentation plays a more critical role in the subclassification.

8.6. *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



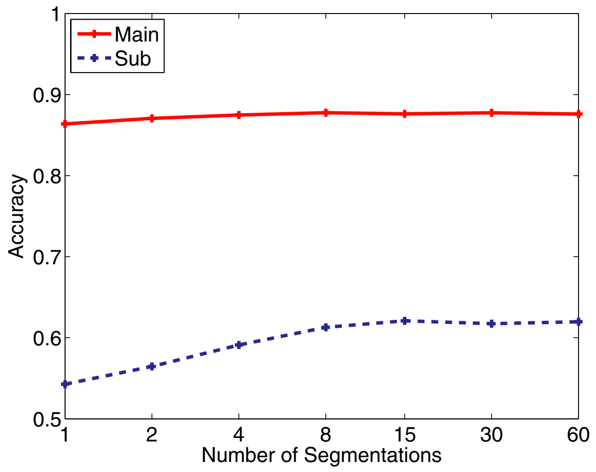
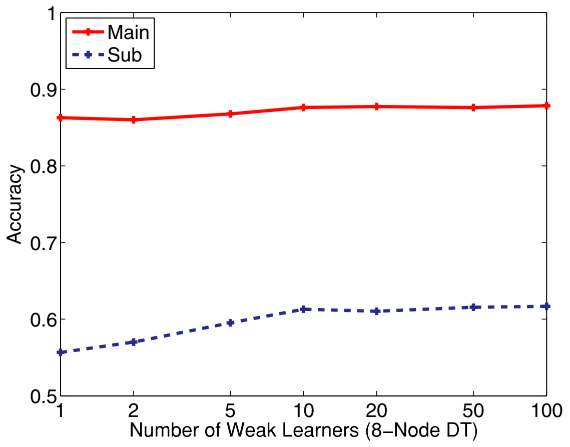


Figure 14. Analysis of segmentation parameters. On the left, classification is performed for single segmentations into varying numbers of segments ("sp" is the segmentation into superpixels). Peak accuracy is at 100 segments, with larger numbers of segments degrading the subclass accuracy slightly. On 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.



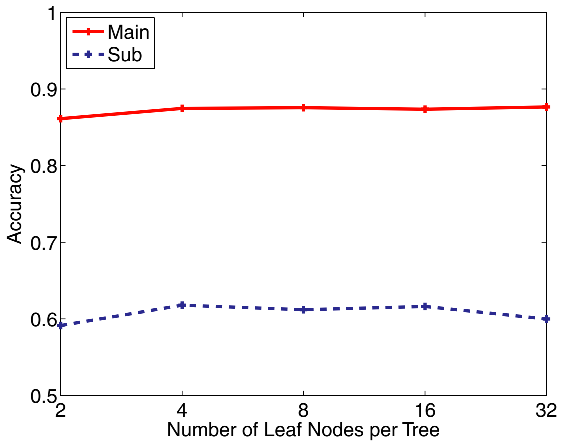


Figure 15. Analysis of classification parameters. The results are fairly robust to the classification parameters. When boosting eight-node decision trees, having ten or more weak learners gives the best results. When keeping the total number of decisions constant (160 nodes total), between trees of between 4 and 16 nodes produce the best results.

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 fixed the number of decisions so that there are 160 leaf nodes 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 Fig. 15 for the results.

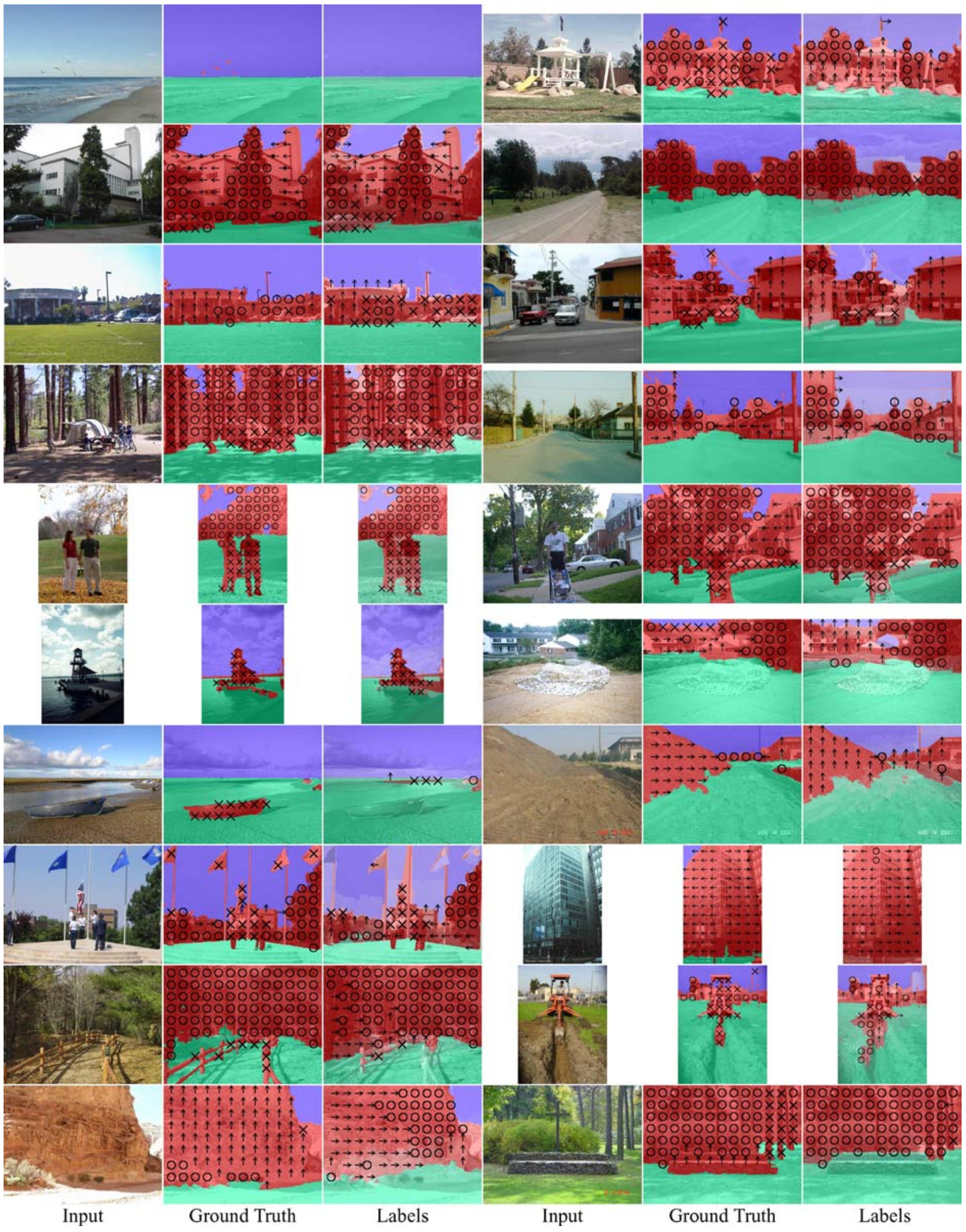


Figure 16. Results from multiple segmentations method. This figure displays an evenly-spaced sample of the best two-thirds of all of our results, sorted by main class accuracy from highest (upper-left) to lowest (lower-right). See Fig. 2 for explanation of colors and markings. Brighter colors indicate higher levels of confidence for the main class labels.

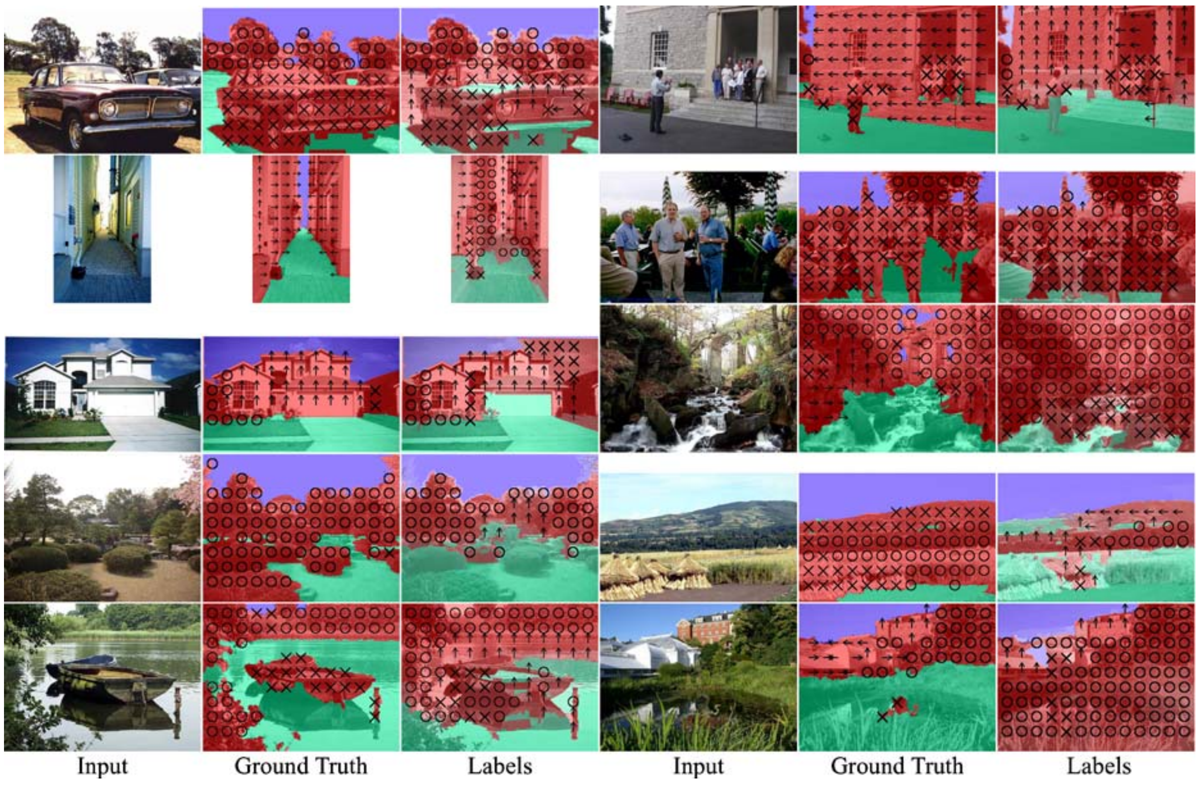


Figure 17. Continued results from multiple segmentations method. This figure displays an evenly-spaced sample of the worst third of all of our results, sorted by main class accuracy from highest (upper-left) to lowest (lower-right).

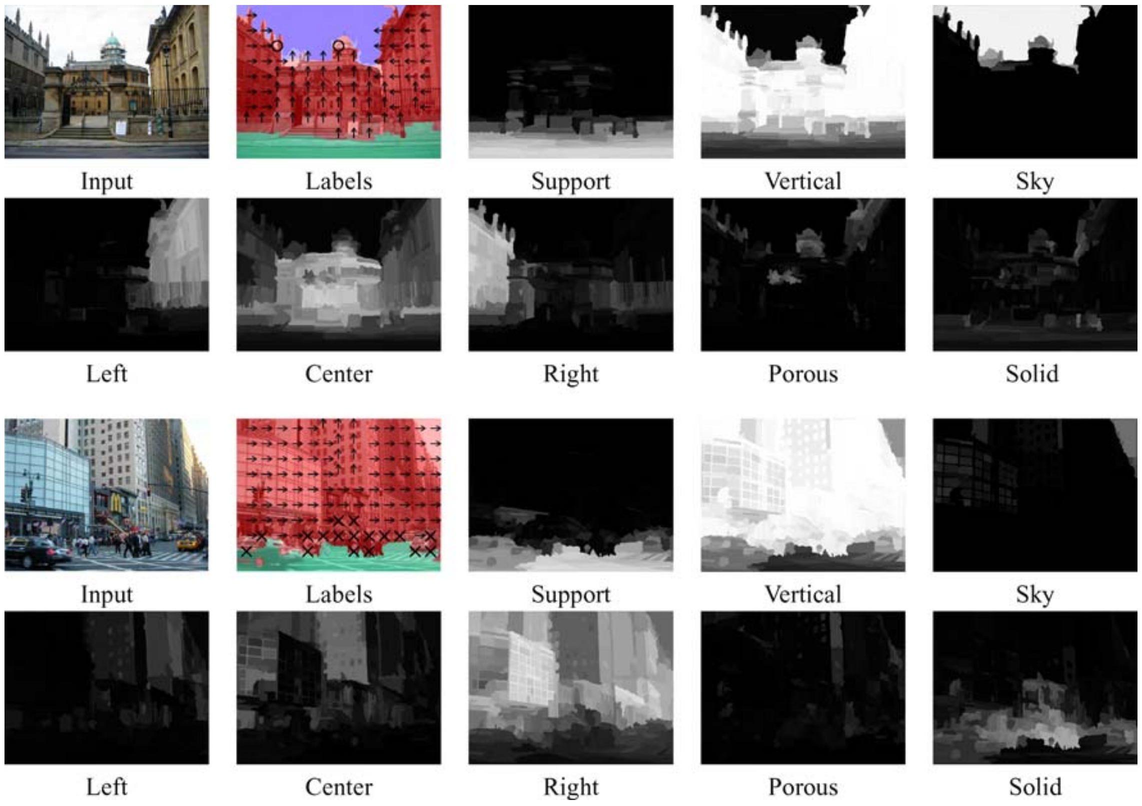


Figure 18. Results with confidence images for multiple segmentations method. The confidence image shows the estimated probabilities over the image for the given label.

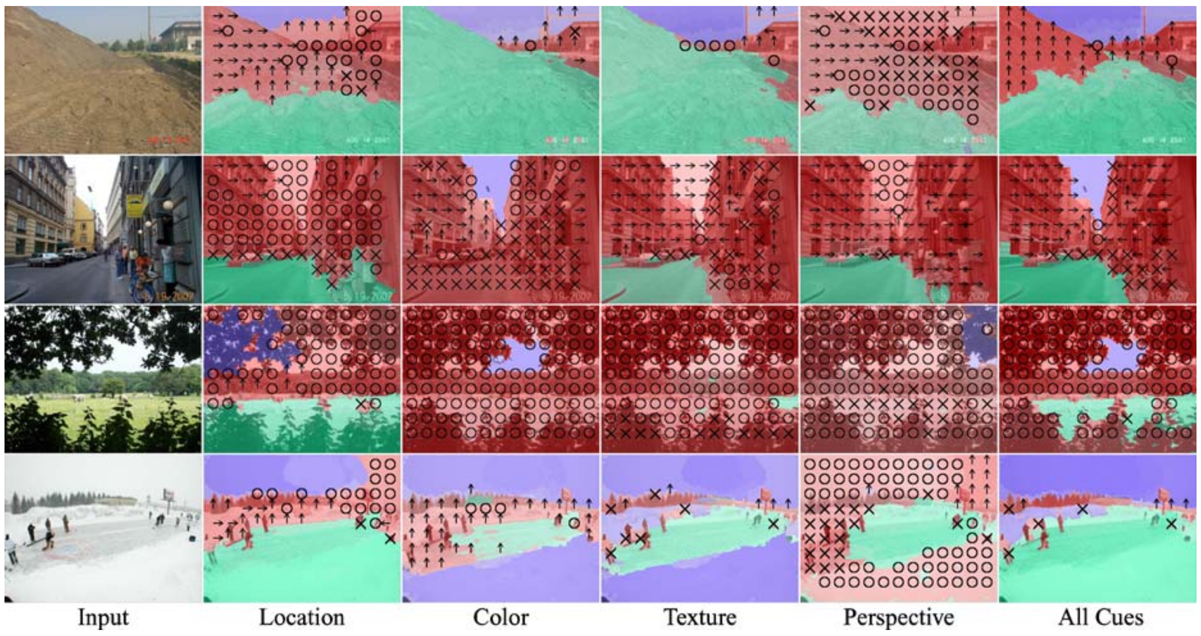


Figure 19. Results when performing classification based on each cue separately and using all cues. In each case, the same multiple segmentations are used (which are based on location, color, and texture), and those segments are analyzed with the given type of cues.

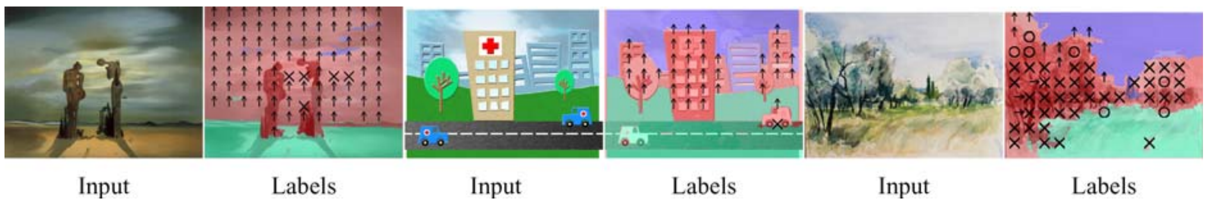


Figure 20. Results on paintings of outdoor scenes. Although the system is trained only on real images, it can often generalize to very different settings.

8.7. *Indoor Scenes*

To demonstrate that our approach can also be effectively applied to indoor images, we annotated ground truth geometry labels for the Stanford dataset of 92 indoor images used to test the indoor 3D reconstruction method of Delage et al. (2006). For simplicity, we used the same geometric classes as 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: the first measuring the accuracy when all classifiers are trained on outdoor images, and the second when 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 Fig. 21.

9. 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 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.

9.1. *Random Field Framework*

In image labeling problems, conditional random fields (Lafferty et al., 2001) (CRFs) are widely used to model

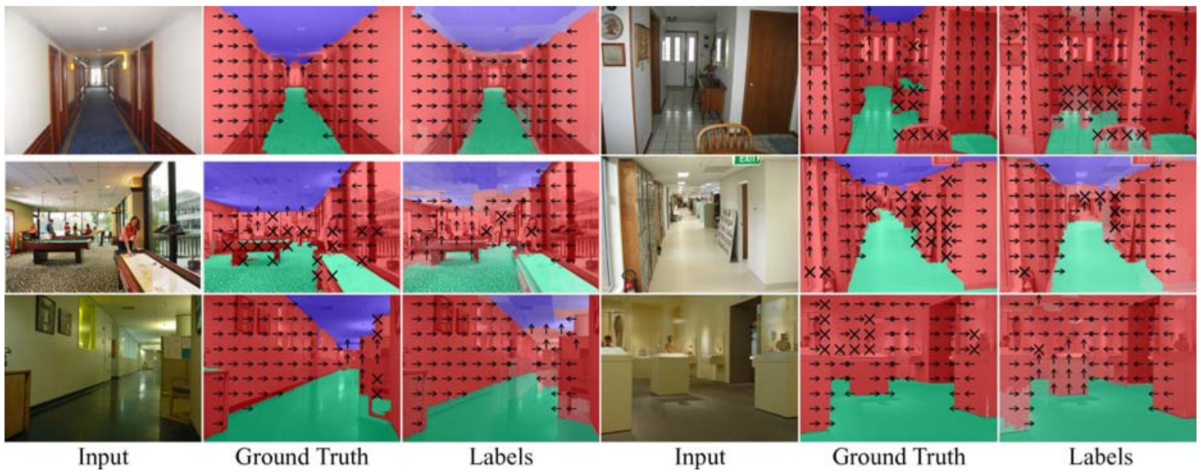


Figure 21. Results on indoor scenes.

the conditional distribution $\mathrm{P}(\mathbf{y} \mid \mathbf{I})$ 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:

$$\log \mathrm{P}(\mathbf{y} \mid \mathbf{I}) = \sum_i f_1(y_i, \mathbf{I}) + \sum_{ij} f_2(y_i, y_j, \mathbf{I}) - \log \mathrm{Z} \qquad (2)$$

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

We can rewrite this as an energy function:

$$\mathrm{E}(y_1..y_n) = \sum_i \mathrm{E}_i(y_i) + \sum_{ij} \mathrm{E}_{ij}(y_i, y_j) \qquad (3)$$

The first term penalizes lack of confidence in the superpixel label: $\mathrm{E}_i(y_i) = -\log \mathrm{P}(y_i \mid \mathbf{I})$. The second term penalizes discontinuity between neighboring labels. Denoting $\mathrm{P}(y_i = y_j \mid \mathbf{I})$ 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 (Boykov et al., 2001). The parameter β controls the relative strength of the pairwise interaction term.

We trained the superpixel label and pairwise likelihood functions as described for the multiple segmentation method. In Table 6, we show that slight gains in subclass accuracy are possible, though the main class accuracy is not significantly improved.

Table 6. Average accuracies for varying levels of β using a conditional random field.

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

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 used to improve the results from multiple segmentations.

9.2. 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 (2005).

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, using the algorithm given in Fig. 8. If split, 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 1. We also learn a classifier that

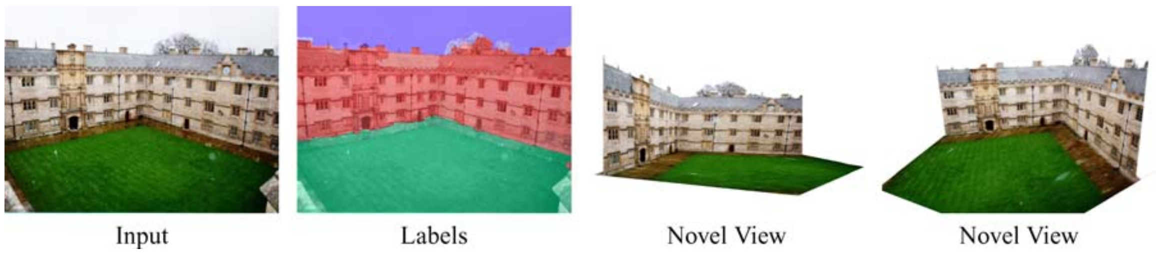


Figure 22. Example of automatic 3D reconstruction based on surface layout (from Hoiem et al. (2005)). Original image used by Liebowitz et al. (1999) and two novel views from the scaled 3D model generated by our system.

outputs the likelihood of the majority label (even if the segment contains multiple types of surfaces). The energy is the defined as

$$E(\mathbf{s}; \mathbf{I}) = -\log \sum_j a_j \max P(\tilde{y}_j \mid s_j, \mathbf{I}) r(\tilde{y}_j, s_j \mid \mathbf{I}) \quad (4)$$

where a_j is the (normalized) area of the segment s_j, $\tilde{y}_j$ is its label, and $r(\tilde{y}_j, s_j \mid \mathbf{I})$ is the regression tree estimate. Each move is accepted with probability $\exp(-\frac{\Delta E}{T_m})$ 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%.

The challenge of the simulated annealing approach is in correctly estimating the majority of the 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.

10. Applications

We believe that our surface layout representation has a wide variety of applications, including automatic single-view reconstruction, object detection, and vision-based navigation. In Hoiem et al. (2005), we show that our main surface labels and a horizon estimate are sufficient to reconstruct coarse, scaled 3D models of many outdoor scenes. By fitting the ground-vertical intersection in the image, we are able to "pop up" the vertical surfaces from the ground. Figure 22 shows the Merton College image from Liebowitz et al. (1999) and two novel views from a texture-mapped 3D model automatically generated by our system. See Hoiem et al. (2005) for additional results and details on how to construct these models.

Most objects, such as cars and people, are roughly vertical surfaces that are supported by the ground. Thus, an estimate of the surface layouts is very helpful for finding such objects. In Hoiem et al. (2006), we show that by simultaneously estimating surfaces, camera viewpoint, and object positions and sizes, we achieve much higher detection accuracy than with local detectors alone (and often recover the camera viewpoint very accurately). We show two examples of improvement in Fig. 23.

Lastly, we demonstrate an application to navigation. In Fig. 24, we show images taken from cameras mounted on a vehicle, the estimated surface layouts, and estimated contours of "safely navigable" terrain. In this simple demonstration, we consider support surfaces to be safe, and our contours are simply the lines for which the confidence in support is equal to 0.75 (green), 0.5 (blue), and

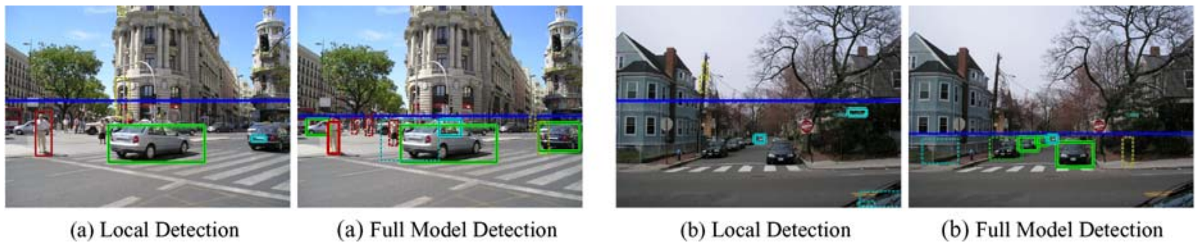


Figure 23. Example of object detection application (from Hoiem et al. (2006)). Detection with surface layout and viewpoint estimation performs much better than purely local detection. The blue line shows the horizon estimate (always 0.5 initially). The boxes show detection estimates (green=true car, cyan=false car, red=true ped, yellow=false ped), with the solid lines being high confidence detections and the dotted lines being lower confidence detections.

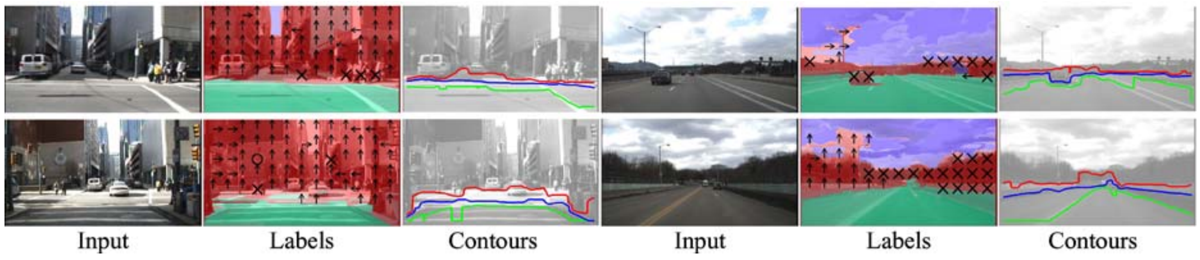


Figure 24. Example of navigation application. We process input images, taken from car-mounted cameras, and produce contours of navigable driving areas. Contours indicate boundaries of "support" surfaces at confidence values of 0.75 (green), 0.5 (blue), and 0.25 (red).

0.25 (red). A robot trying to decide where to explore could consider the area under the green line to be safe, between the green and the blue to be probably safe, between the blue and the red to be probably unsafe, and above the red to be unsafe. Thus, by using the confidences, rather than relying on the accuracy of the most likely labels, a robot can make a reasonable plan for exploration and traversal, as further demonstrated in Nabbe et al. (2006).

11. Discussion

By posing surface estimation as a recognition problem, we are able to recover the coarse surface layout in a wide variety of outdoor 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.

We see three important avenues for improvement. First, the surface layout estimation could benefit from additional image cues, more accurate segmentations, and models of label relationships. For instance, shadows and reflections, which now tend to confound our algorithm could be used as evidence to help it. Improved grouping or segmentation algorithms would provide reliable segmentations into fewer segments, allowing better use of complex cues and less error due to incorrect segmentations. Also, label relationships, such as such as relative position (the sky is above the ground) and boundaries (e.g. edge junctions), could be used to improve results further.

Second, 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. We have no knowledge of the occluding contours and, thus, cannot determine whether neighboring regions in the image are physically connected. With such knowledge, we could separate the person from the row of cars behind him and determine the rough layout of objects in cluttered scenes.

Finally, we need to use our information about the surfaces and space of the scene in conjunction with other types of scene information. Surfaces, boundaries, and objects all live and interact in the same visual world that reflects our highly complex yet structured physical world. Analysis of the individual constituents of the scene can only take us so far. To truly understand the scene, we need to model the interactions within the scene and the organization and semantics of the scene as a whole. This will certainly require advances in local analysis, but, more importantly, there is a great need for methods to reason about the various pieces of the scene in a coherent and intelligent fashion.

Acknowledgments

We would like to thank Rahul Sukthankar and Takeo Kanade for valuable discussions and references. We thank Jake Sprouse for providing images for the navigation application, Heegun Lee for annotating the indoor images, and our reviewers for helping to improve the clarity of the paper. This work is partially funded by NSF CAREER award IIS-0546547 and a Microsoft Research Fellowship to DH.

References

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

Arbelaez, P. 2006. Boundary extraction in natural images using ultrametric contour maps. In *Proc. CVPRW*.

Barbu, A. and Zhu, S.-C. 2005. Generalizing swendsen-wang to sampling arbitrary posterior probabilities. *PAMI*, 27(8):1239–1253.

Barrow, H. and Tenenbaum, J. 1978. Recovering intrinsic scene characteristics from images. In *Computer Vision Systems*.

Biederman, I. 1981. On the semantics of a glance at a scene. In Kubovy, M. and Pomerantz, J.R., (Eds), *Perceptual Organization*, chapter 8. Lawrence Erlbaum.

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

Brooks, R., Greiner, R., and Binford, T. 1979. Model-based three-dimensional interpretation of two-dimensional images. In *Proc. Int. Joint Conf. on Art. Intell.*

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

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

Delage, E., Lee, H., and Ng, A.Y. 2006. A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image. In *Proc. CVPR*.

en Guo, C., Zhu, S.-C., and Wu, Y.N. 2003. Towards a mathematical theory of primal sketch and sketchability. In *Proc. ICCV*.

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

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

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

Gibson, J. 1950. *The Perception of the Visual World*. Houghton Mifflin.

Guzman-Arenas, A. 1968. Computer recognition of three-dimensional objects in a visual scene. In *MIT AI-TR*.

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

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

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

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

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

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

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

Koenderink, J.J. 1998. Pictorial relief. *Phil. Trans. of the Roy. Soc.*, pp. 1071–1086.

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

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

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

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

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

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

Liebowitz, D., Criminisi, A., and Zisserman, A. 1999. Creating architectural models from images. In *Proc. EuroGraphics*, vol. 18.

Marr, D. 1982. *Vision*. Freeman, San Francisco.

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

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

Ohta, Y. 1985. *Knowledge-Based Interpretation of Outdoor Natural Color Scenes*. Pitman.

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

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

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

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

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

Roberts, L. 1965. Machine perception of 3-d solids, pp. 159–197.

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

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

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

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

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

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

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

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

Tao, H., Sawhney, H.S., and Kumar, R. 2001. A global matching framework for stereo computation. In *Proc. ICCV*, pp. 532–539.

Tenenbaum, J. and Barrow, H. 1977. Experiments in interpretation guided segmentation. 8(3):241–274.

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

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

Tu, Z. and Zhu, S.-C. 2002. Image segmentation by data-driven markov chain monte carlo. *PAMI*, pp. 657–673.

Warren, R.M. and Warren, R.P. 1968. *Helmholtz on Perception: Its Physiology and Development*. John Wiley & Sons.

Yakimovsky, Y. and Feldman, J.A. 1973. A semantics-based decision theory region analyzer. In *Proc. Int. Joint Conf. on Art. Intell.*, pp. 580–588.