Occlusion Boundaries: Low-Level Detection to High-Level Reasoning

Andrew Neil Stein

CMU-RI-TR-08-06

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

February 2008

Thesis Committee:
Martial Hebert, *Chair*
Alexei Efros
Rahul Sukthankar
David Fleet, *University of Toronto*

Contents

List of Figures

Please note that *many* figures in this document make extensive use of color.

List of Tables

Abstract

The boundaries of objects in an image are often considered a nuisance to be "handled" due to the occlusion they exhibit. Since most, if not all, computer vision techniques aggregate information spatially within a scene, information spanning these boundaries, and therefore from different physical surfaces, is invariably and erroneously considered together. In addition, these boundaries convey important perceptual information about 3D scene structure and shape. Consequently, their identification can benefit many different computer vision pursuits, from low-level processing techniques to high-level reasoning tasks.

While much focus in computer vision is placed on the processing of individual, static images, many applications actually offer video, or *sequences* of images, as input. The extra temporal dimension of the data allows the motion of the camera or the scene to be used in processing. In this thesis, we focus on the exploitation of subtle relative-motion cues present at occlusion boundaries. When combined with more standard appearance information, we demonstrate these cues' utility in detecting occlusion boundaries locally. We also present a novel, mid-level model for reasoning more globally about object boundaries and propagating such local information to extract improved, extended boundaries.

Building on these methods, we also demonstrate enhancement of two high-level vision tasks by incorporating boundary information. First we employ boundary fragments to suggest multiple "hints" of a scene segmentation and then use these suggestions collectively to achieve more consistent and parsimonious delineation of generic whole objects. Second, we augment a popular feature-based recognition technique for specific objects (the Scale Invariant Feature Transform) with boundary information in order to yield a method more robust to changes in background and scale.

This thesis thus contributes to research on occlusion at several levels, from low-level motion estimation and feature extraction; to mid-level reasoning, classification, and propagation; and finally to high-level segmentation and recognition. In addition, a new video dataset is presented to enable further research in this area.

Funding Sources

We gratefully acknowledge the following funding sources which helped make this work possible:

- National Defense Science and Engineering Graduate (NDSEG) Fellowship, administered by the American Society for Engineering Education (ASEE).
- National Science Foundation Graduate Fellowship.
- National Science Foundation Grant IIS-0713406.
- The Intelligent Robotics Development Program at the Korean Institute of Science and Technology (KIST), a 21^{st} Century Frontier Research and Development Program funded by the Korean Ministry of Commerce, Industry, and Energy.

Acknowledgements

First and foremost I want to thank my wife, Sarah, for her enduring patience, love, and support during the long, meandering process of defining and pursuing a thesis topic, not to mention the wildly unpredictable ups and downs of research and life in general. She has put up with more than would seem reasonable by any measure during our combined graduate school experience, and she has endured in spite of it all.

My immediate family — Mom, Dad, Renée, Eleanor, Gregg, and Brian — and many close friends also deserve recognition for their support and many hours of patient listening to excruciating details of my progress. In particular, my officemates Caroline Pantofaru and Tom Stepleton have certainly tolerated more than their fair share of discussion and worry over my work, despite the constant beckoning of their own research. I thank them for all the useful suggestions and support, and especially for the laughs; I hope I have been even half as much a help to them. My good friend, Brad Formby, has also been an enormous help to me over the years as a voice of reason, as an excellent sounding board for ideas and options, and as a thoughtful — and thought-provoking — source of support.

I also want to thank my committee, Alyosha Efros, Rahul Sukthankar, and David Fleet, and most of all Martial Hebert as my advisor, for their helpful suggestions and ideas, for their technical advice, for their proof-reading, and for keeping me focused on the problem at hand through it all.

A big thanks to Suzanne Lyons Muth for her role as the Robotics Institute's fairy godmother and for keeping things organized — and most certainly for all the afternoon chocolate.

Finally, I would be remiss if I did not acknowledge my neurotic cats, Zoe and Linus, for substantial comic relief and (sometimes unwanted) distraction over the years, irrespective of prevailing stress levels. Despite my best efforts, however, their Matlab skills still leave a lot to be desired.

Chapter 1

Introduction

Consider the scene depicted in Figure 1.1, taken from the LabelMe database [147]. There are many overlapping objects and surfaces in this scene. Indeed, almost *everything* in the scene is occluded by and/or occludes another object or surface! A user has begun to label a few foreground objects thus far (indicated by the blue and red dots), but the process of painstakingly labeling each and every inter-related boundary is daunting.

In natural images, however, this type of complexity is common. Objects are generally not conveniently laid out in well-separated poses or in front of uniform backgrounds. As a

Figure 1.1: Example scene exhibiting extensive occlusion from [147]. Almost every object or surface is occluding and/or occluded by another object or surface. Any computer vision method which spatially aggregates information in this scene will almost certainly simultaneously consider data from two different objects.

consequence, occlusion and dis-occlusion at objects' boundaries frustrate many processing techniques frequently used in computer vision. Due to imaging noise and the inherent lack of information (and resulting ambiguity) when considering an individual pixel, most processing techniques in our field aggregate information spatially in images. This aggregation may be the result of simple smoothing (*e.g.*, Gaussian blurring for the purpose of down-sampling and multi-scale processing), the consideration of discrete patches of pixels (*e.g.*, for feature-based object recognition or face/pedestrian detection), or the use of graphical models connecting neighborhoods of pixels (*e.g.*, Markov, Conditional, or Discriminative Random Fields [63, 92, 93]).

Figure 1.2: In cluttered scenes, any technique that aggregates information spatially, such as a patch-based method, will erroneously combine information from physically different objects/surfaces, leading to poor results.

Each of these methods implicitly assumes that all the nearby or connected pixels "belong together" (*e.g.* are from the same object, motion layer, *etc.*). But this assumption is violated at object boundaries, where information from two different physical surfaces is smoothed/transmitted across the boundary or collected within a single patch (as shown in Figure 1.2), and thus subsequent results will be muddied or completely incorrect. For this reason, pixels near boundaries are often treated as outliers to be handled by robust methods, or multiple/adaptive-windowing techniques are employed [61, 73, 84]. These sorts of approaches are inherently focused on reasoning based on information contained within the regions enclosed by the boundaries. By contrast, the work of this thesis will focus on boundaries themselves and will attempt to detect and reason about them directly. Note that these two sources of information, regions and their boundaries, are complementary and are each useful in their own way [56, 70, 163]; we are not advocating the exclusive use of one over the other.

1.1 High-level Reasoning with Occlusion Boundaries

The boundaries present in a scene are not only a nuisance for many processing techniques, they are also a valuable source of perceptual information important for understanding a scene's overall structure and content [20]. Since occlusion boundaries exist at locations in an image where one physical surface is closer to the camera than another, they correspond

to the physical extent of objects and structures in a scene, providing strong 3D shape cues without explicit 3D reconstruction. In this thesis, we will focus on increasing performance in two high-level tasks using detected occlusion boundaries: object segmentation and object recognition.

1.1.1 Object Segmentation

There has been impressive progress in the last few years in *recognizing* specific objects in images such as cars, bicycles, people, *etc.*, by using, for example, methods based on appearance features [15, 19, 28, 49, 87, 105, 106, 120, 124] or based on shape features [18, 31, 121, 153]. Progress is also being made in the related, but far more difficult problem of recognizing *categories* of objects [46, 47, 82, 136]. But the problem of *detecting* generic, never-before-seen objects — *i.e.* without a given library of knowns — remains a difficult challenge. For example, how may we determine a telephone sitting on our desk is an object separate from its surroundings, without already knowing what a telephone is? Or as Adelson and Bergen put it [2], how do we distinguish the "things" from the "stuff?" Imagine a system as capable as our own of understanding the structure of a visual scene *without* necessarily having in its memory an instance of every specific object present, and *without* explicit 3D reconstruction or geometry.

This goal, depicted in Figure 1.3, is variously known as object segmentation, *pop-out*, or figure-ground labeling, and we believe that detected occlusion boundaries, which themselves delineate the physical extents of objects, can provide a strong cue for tackling it. Thus, in Chapter 5, we will use occlusion boundaries to provide segmentation "hints" which we combine to compute a powerful affinity measure suitable for existing clustering and segmentation algorithms.

1.1.2 Object Recognition

Consider the methodology employed by feature-based object recognition relying on appearance, *e.g.* the popular Scale Invariant Feature Transform (SIFT) approach due to Lowe [106]. There are generally two stages: feature detection and feature description. Image information is aggregated during both. During detection, filtering processes such as Gaussian smoothing, which are used to create scale-space Difference-of-Gaussian pyramids, will smear information across boundaries. Then, patches of image data which may cross object boundaries are used to create descriptor vectors for matching. In particular, for scale-invariant methods, larger and larger neighborhoods are considered as the scale of detection/ description increases, resulting in many unusable large-scale features which contain information from (multiple) objects and background. Equivalently, as an object appears smaller and smaller within a scene, we must rely more heavily on its larger scale

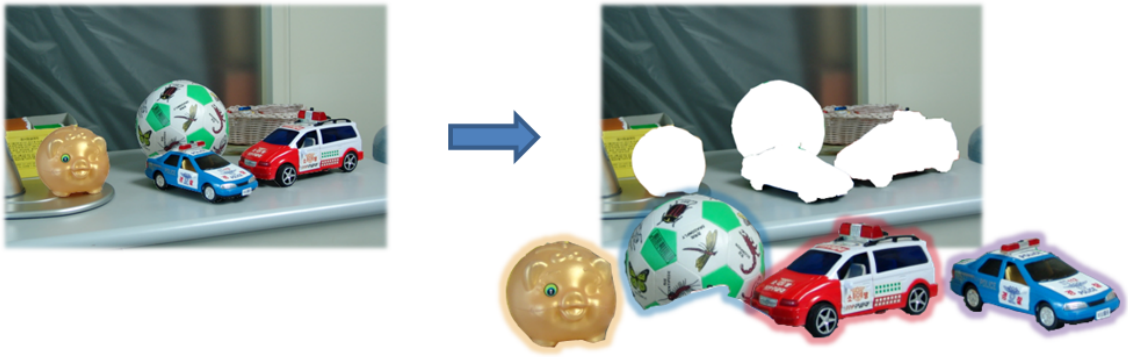


Figure 1.3: An example of object "pop-out". Without prior knowledge of these four specific objects (*e.g.* in a database of example images), we would like still to be able to differentiate them from the background. Knowledge of their physical extents, as delineated by their boundaries, is a strong cue for enabling this task.

features (relative to observed size of the object) for matching. Knowledge of the location of object boundaries in a scene could be used to combat these problems and improve recognition in cluttered scenes [165], as will be discussed in Chapter 6.

1.2 Locally Detecting Occlusion Boundaries

Given the motivations above for locating occlusion boundaries in a scene, we turn now to methods of detecting them, which is the focus of Chapters 2-4. Usually, occlusion boundaries are also visible as edges in images. In this work, then, we will explicitly distinguish between the detection of typical *edges*, which may result solely from changes in appearance, and occlusion *boundaries* which additionally correspond to 3D scene structure. In some situations, the foreground and background surfaces may share indistinguishable appearance, producing a camouflaged boundary. We do not address such situations explicitly in this work, but we note that they are rather rare in general. Furthermore, when such camouflage arises, it is often a *local* phenomenon. More global reasoning about boundaries in the scene can help alleviate this type of difficulty; this is the subject of Chapter 4.

In our work, we will consider occlusion boundaries to be a subset of the edges in a scene. Our goal, then, will be to identify those edges in an image which also correspond to occlusion boundaries. We will employ tools from machine learning in order to learn classifiers for this task from labeled data. This will require extracting and reasoning about the various cues present at boundaries, particularly those cues that distinguish them from edges.

Given the overlap between standard *edges* and the *boundaries* we seek, typical appearance cues such as gradients of brightness, color, and texture are clearly important for

our task. Unfortunately, non-boundary edges, such as simple surface markings or lighting effects, can also give rise to these cues. Thus, while the use of sophisticated edge detectors that can handle texture and color [114, 116, 148, 186] may offer improved results over a simple Canny edge detector [30], they still (correctly) respond strongly to edges which do not correspond to any physical occlusion. Note that Berkeley's learned *Pb* detector [114], which is becoming increasingly popular, has been trained to detect *semantic* edges (though the authors refer to them as "boundaries" in that work) from a database of human-labeled images and not necessarily boundaries which explicitly correspond to occlusion.

Many other cues have been proposed with respect to this problem and to the related problem of figure-ground determination as described above, most notably those based on Gestalt principles such as continuation, parallelism, proximity, and closure [135, 145, 164]. Though largely a product of human studies, many of these cues have received significant attention in computer vision, *e.g.* [4, 71, 109, 140, 194].

Most of these cues can be exploited in a single image, but ours is a temporal experience, and increasingly, image sequences and video are becoming commonplace. Indeed, while some computer vision applications, such as image retrieval, necessarily limit the system to the consideration of a single image, using only one instantaneous snapshot of the world as input can unnecessarily over-complicate some problems. For vision applications operating in the physical world, it is quite reasonable to assume a temporal sequence of images is available. Why force a mobile robot, for example, to attempt to understand its surroundings from disconnected still snapshots? It has the ability to move itself or to manipulate its environment and observe the result as a continuous, connected event. In such a system, the additional temporal dimension provided by the image sequence yields an extra source of information that ought not be ignored: *motion*.

Thus another potentially powerful cue for our task, and one which is well-known in psychophysics, is that of motion — specifically the relative motion discrepancies at the depth discontinuities which occur at occlusion boundaries. Most often studied in controlled laboratory experiments in order to elucidate or model human perception [16, 33, 85, 117, 173, 191], such motion cues have received comparatively little attention by computer vision researchers for the purposes of boundary detection [20, 51, 126] or object segmentation and pop-out, *e.g.* [4, 142].

We will therefore employ motion observed in a short video clip to help distinguish between edges and boundaries in a single reference image from that clip. This motion may be caused by camera movement, which induces parallax at depth discontinuities, or it may be a result of dynamic objects in the scene (or a combination of the two). Our approach handles either situation equivalently and is thus more general than motion *detection* work that relies on a static camera for background subtraction, *e.g.* [142, 179]. We will discuss the various motion cues available at occlusion boundaries in Chapter 2 and discuss their

use in locally detecting those boundaries in Chapter 3. Note that despite our use of video clips, we are still only concerned with the discovery of boundaries in a single "reference" frame of the clip, namely the middle frame. The remaining frames serve only as additional (temporal) information, allowing the computation of motion cues.

Focusing explicitly on motion at the edges in a scene will allow us to avoid dense motion estimation and reasoning via full-blown optical flow followed by region-growing or clustering approaches. In addition, we can detect *local* occlusion directly using a bottom-up approach which is complementary to top-down approaches that rely on higher-level reasoning. As discussed in Section 1.4, such methods often impose the restrictive assumption that the scene consists of a set of distinct layers moving separately. Our approach will not require this assumption. One challenge of a purely local, bottom-up approach, however, is estimating and using motion information at exactly those locations (occlusion boundaries) at which it is arguably the most difficult to obtain.

We recognize that depth cues are not strictly limited to temporal data (*e.g.* in the form of motion): there exist monocular, static cues such as shading, shadows, familiar size, perspective, and atmospheric effects, as well as proprioceptive/occulomoter cues such as focus and convergence (*i.e.*, binocular stereo). But it is currently difficult if not impossible to extract and employ all such cues in a practical computer vision system. Many of these cues require high-level, *a priori* information — often precisely the same information we seek to extract from using those cues in the first place (one of the many "chicken and egg" difficulties in computer vision). Low-level proprioceptive data is also generally not available. This is not to say that these cues are useless. In some domains, they may in fact be necessary, and continued research in these areas is certainly warranted. But their extraction is beyond the scope of this work. *Motion cues*, however, are practical and readily available in video.

1.3 Global Boundary Reasoning

Thus far, we have only discussed utilizing *local* motion and appearance cues for occlusion boundary detection. Purely local strategies are generally limited and noisy. Moreover, occlusion may not even be visible locally, as is the case along a section of boundary seen in front of a uniform background (see Figure 1.4). Thus, a natural next step is to use our local cues in a more global reasoning approach. In Chapter 4 we present such an approach, which builds on the local methods described in Chapters 2-3.

There we combine many local cues into a single classifier, following recent approaches to edge detection/classification in a single image [41, 90, 114], with the differences that we use motion cues in addition to appearance cues, and we focus on physical occlusion boundaries. We also utilize the region boundaries of an image's (over-)segmentation as

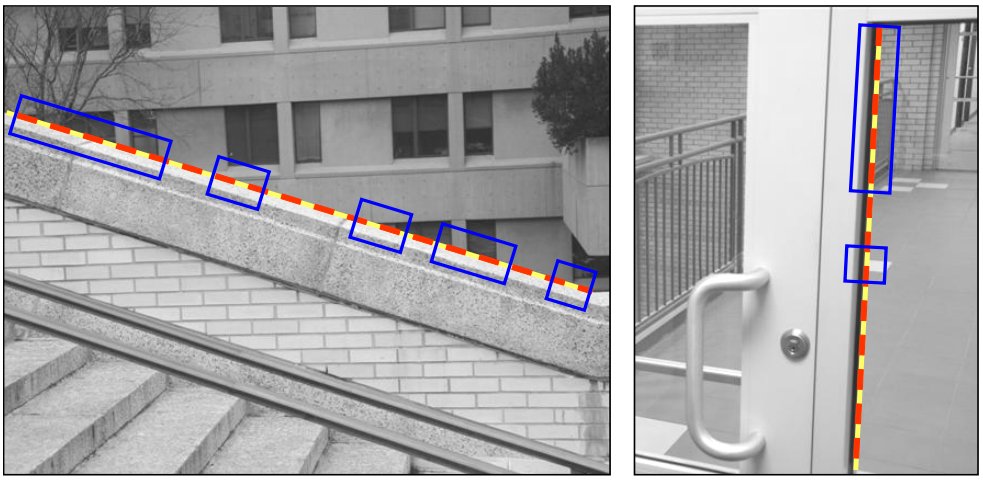


Figure 1.4: The need for propagation of local occlusion information. For the red/yellow dashed boundaries in the scene, a purely local occlusion detection method will likely only be able to detect occlusion within the blue boxes. We therefore need to propagate the higher-confidence occlusion observations to the in-between sections of the boundary, for which labeling may be less confident due to the lack of motion observed against the nearly-uniform background.

initial candidates to be labeled as occluding or non-occluding. This idea is largely due to Ren [140], who used such an approach to induce a figure/ground labeling of the regions themselves. An additional advantage of this approach lies in the improved spatial support for extracting cues by relying on data-driven regions of support and more extended boundary *fragments* instead of individual pixels. Finally, we also define a novel model for inferring a globally consistent labeling of the boundary fragments by constructing a factor graph capable of propagating measurements between local structures. This process is illustrated in Figure 1.5.

1.4 Additional Related Work

In this section we will briefly discuss work closely related to (motion-based) occlusion boundary detection and not necessarily explicitly covered elsewhere. Note, however, that further discussion of related work is distributed throughout the remainder of this thesis.

Prior attempts to use motion cues to extract object contours (or object segmentations implying the contours) can be divided roughly into two groups: those that segment regions directly from the motion input, and those that detect the contours more directly via some local computation on the motion data.

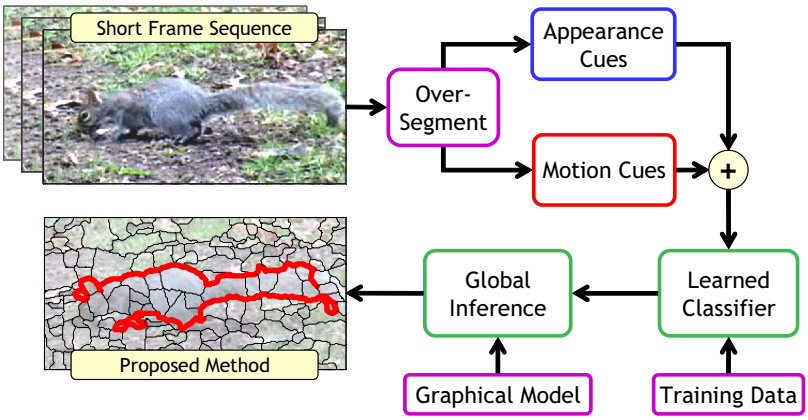


Figure 1.5: In Chapter 4, we develop a framework for object/occlusion boundary detection combining appearance cues, subtle instantaneous motion cues, learned classifiers, and a global inference technique.

1.4.1 Segmentation-Based Techniques

The first category includes approaches that attempt to infer segmentation or scene structure directly from reasoning about large-scale occlusions observed through dynamic object motion [27, 128] or the use of multiple, calibrated cameras for obtaining silhouettes [65].

Also in this category is layered motion segmentation [38, 181], in which regions are segmented from an input image sequence based on the consistency of motion within each region, *e.g.* [78, 79, 80, 86, 91, 128, 158, 162, 184, 188]. Most of these techniques use a parametric motion model for each layer, and employ various techniques, such as Expectation Maximization, for estimating those models and for assigning pixels to the correct layer or model. Typical models are restricted to near-planar, rigidly-moving regions (though an exception can be found in [184]). In addition, many approaches assume a known, fixed number of layers in the scene or do not scale well as that number increases. We argue that attempting to explain the scene generatively in terms of a specific number of motion-consistent connected regions may not be necessary, and instead we propose to detect a large fraction of the objects' boundaries by estimating local motion cues and using them in a discriminative statistical classifier combined with a mechanism to enforce global consistency. Quite recently, a method for *binary* segmentation of video was presented which combats some of the difficulties of layered motion segmentation methods by combining clustered motion features (akin to the textons popularized by recognition research) with a boosted tree-based classifier [190].

Usually, the erratic results near boundaries are treated as outliers to an underlying smooth process by using, for example, the methods described above. The subsequent delineation of precise motion boundaries, if performed at all, is generally of secondary

importance. A notable exception, however, is found in [70], where vertical and horizontal between-pixel motion boundaries plus their interactions with nearby dense optical flow vectors are considered in an MRF framework. Similarly, dual graphs on the pixels and the edges between them (sometimes referred to as "cracks") are considered in a normalized cuts framework in [195]. Stereo or structure from motion techniques also have trouble near occlusion boundaries and usually focus on the interiors of regions while handling data near occlusions as complex special cases [61, 73, 84]. Our work, on the other hand, is not focused on the precise, dense motion estimates themselves, nor on full 3D scene reconstruction; we first seek only to *identify* boundary locations that correspond to visible occlusion and then to employ those boundaries in higher-level reasoning.

Browstow and Essa [27] segment a static scene into planar relative depth layers by observing the occlusions introduced by an object moving through that scene. The use of edge detection and contour completion help provide good estimates of the layers' (occluding) boundaries. The approach, which is limited to a stationary camera, is aimed primarily at compositing video and may also require manual intervention.

Because videos often include dynamic objects observed by stationary cameras, particularly in surveillance applications, motion detection and background subtraction are closely related topics [142, 179]. These techniques are limited to finding the boundaries of moving objects, while we are interested in extracting boundaries of static objects as well. In fact, our approach is completely agnostic as to the whether the camera is moving, the scene is dynamic, or both.

1.4.2 Contour-Focused Techniques

In the second category, techniques have been developed based on the observation that occluding contours can be defined as extremal boundaries, where the viewing ray is tangent to the object's surface. This led to the development of algorithms that rely on an explicit geometric model of the motion of occluding contours [95, 150, 153, 178]. These approaches are appealing because they rely on well-defined, mathematically correct, geometric models. However, one drawback is their sensitivity to deviations of the actual data from the model.

An alternative is to use an implicit model, either learned from local motion cues estimated from training data or based on some fixed model of the distribution of motion cues in the vicinity of occluding boundaries [20, 51, 76, 126, 167, 168]. Our work falls in this general category in that we do not attempt to precisely *model* the motion of occlusion boundaries directly. Instead we rely on the statistical discrimination of *relative* local motion cues at those boundaries.

Smith *et al.* [162] track edge fragments and use Expectation Maximization to then segment the scene into regions with consistent motion. Their approach is one of the few

that tracks edges/boundaries directly (along with [20, 51, 126] discussed below), but it still assumes a strictly layered scene structure (and thus was also referenced in the previous section). The method seems to work best on two-layer sequences as computation increases exponentially with the number of layers. Also, the initial step in [162] of linking edge pixels into chains which are part of a single surface is non-trivial, but quite crucial since it is a hard decision imposed on the remainder of the system. One theme throughout our work here is to follow the conventional wisdom of maintaining "soft" estimates for as long as possible, thereby retaining as much information for as long as possible in processing. Thus, as discussed in Chapter 4, we choose to avoid this type of brittle edge chaining procedure.

Black, Fleet, and Nestares also attempt to estimate local evidence of occlusion by analyzing motion in the vicinity of boundaries [20, 51, 126]. They build a generative, parametric appearance model of local occlusion within sampled circular regions in terms of foreground/backgroud motion and a linear occlusion boundary offset from the region's center. They then estimate the posterior probability of this model using particle filtering. Sampling the large parameter space for each region requires thousands of particles and significant computational expense, a common problem with particle filtering in high-dimensional spaces.

1.4.3 Other Approaches

For *active* detection of occlusion boundaries, it is possible to reason about the various shadows produced by taking multiple images illuminated by offset flashes [48, 138]. Furthermore, [37] uses these occlusion boundaries to reconstruct entire surfaces, instead of using traditional visual hull techniques [64, 115]. Our occlusion boundary techniques may facilitate similar applications in a purely passive vision system.

Also related to occlusion detection is the classical problem of T-junction detection [62, 69, 89, 151, 193]. This problem was recently addressed in a discriminative framework by Apostoloff and Fitzgibbon [4]. They utilize spatio-temporal data (as do we), as pioneered by Baker and Bolles [10, 22], who introduced Epipolar-Plane Image (EPI) analysis for reasoning about 3D scene structure from gradients in spatio-temporal slices of video. By detecting T-junctions in those same spatio-temporal slices rather than in single images (though not in spatio-temporal *volumes* as we will describe in Chapter 2), they demonstrates sparse T-junction detection and limited extraction of occlusion boundaries. Furthermore, as will be discussed in Chapter 5, the same authors later demonstrated the utility of their T-junction detector for binary video segmentation [6]. For higher level reasoning, note that T-junction detection may be a complementary source of information to the motion boundary detection presented here.

Finally, and somewhat related to the notion of extremal boundaries discussed above,

there exist many techniques for constructing *visual hulls* from multiple views or silhouettes of an object, *e.g.* [65, 64, 115]. These methods generally rely on known camera geometry and/or widely-spaced views to reason about visibility in 3D space and "carve" objects out of their surroundings. Usually, the resulting boundaries are quite rough and thus our boundary-focused approaches could be considered complementary.

1.5 Our Dataset

We first need appropriate data for testing our methods as well as ground truth labelings of that data in order to train those techniques relying on learned classifiers. In addition, using such a dataset with a significant number and variety of scenes offers a more complete, quantitative analysis as compared to typical anecdotal examples often provided in research using motion data. While segmentation datasets exist, most notably the Berkeley Segmentation Data Set (BSDS) [113], they are not appropriate for our task for two reasons. First, our use of a motion cue requires that we have at least one additional image of each scene in order to observe the effects of camera/scene motion. The BSDS consists only of single isolated images, as do all other segmentation datasets of which we are aware. Second, the BSDS edge labels provided by the human subjects do not necessarily correspond to physical occlusion boundaries: any edge which a subject found *semantically* salient may be marked.

For these reasons, we have created — and made publicly available — a new dataset which addresses both of these issues: 30 short image *sequences*, approximately 8-20 frames in length, are provided to allow motion estimation, and only the occlusion boundaries are labeled as ground truth in the reference (*i.e.* middle) frame of each sequence.[1] The scenes in this dataset are used throughout the experiments described in this thesis. We emphasize that all results for boundary detection, object segmentation, *etc.*, are reported for the reference frame only; the additional frames are used only to provide extra motion cues.

Some example scenes are depicted in Figure 1.6 along with their ground truth occlusion/object boundary labels. The dataset is quite challenging, with a variety of indoor and outdoor scene types, significant noise and compression artifacts, unconstrained handheld camera motions, and some moving objects. Though probably not appropriate for outdoor scenes, the multi-flash imaging approach described in [138] could prove useful for automating the labeling process for future additions to the database.

All sequences were collected with the same video camera at 15 frames per second (thus they are approximately one to two seconds in duration). For those scenes in which the camera is moving, that motion is generally horizontal (panning) with an effective

[1] Admittedly, some subjectivity in labeling is unavoidable, but we believe the boundaries we seek are defined more clearly than typical edges.

Figure 1.6: Ground truth occlusion boundaries labeled for 12 of the scenes from our dataset. Each example is the reference (middle) frame of a short sequence, usually 8-20 frames.

total baseline on the order of ten centimeters. The objects of interest (not considered as "background") are roughly one to five meters from the camera, with between-object distances ranging from a few centimeters to a few meters. The amount of motion within the image plane is generally 2-20 pixels per frame. After stabilization (see Section 3.1.1), this drops to half a pixel per frame on average.

1.6 Contributions and Overview

This thesis contributes to research on occlusion boundaries at several levels, from low-level processing to high-level reasoning:

- We describe a novel patch-based, spatio-temporal edge detector which extends the benefits of state-of-the-art spatial edge detectors to video, offering the ability to estimate not only orientation but also normal edge speed [169]. See Section 2.2.
- We analyze the relationship of various local motion estimates to the discovery of occlusion and demonstrate that the use of these motion cues improves detection performance over techniques relying on appearance cues alone [166, 168]. See Chapters 2 and 3.

- We introduce a novel factor graph model with learned potentials (developed in conjunction with Hoiem *et al.* [76, 167]), combine it with over-segmentation and boundary hypothesis techniques, and demonstrate mid-level, global reasoning about boundaries in an image using a variety of motion and appearance cues. See Chapter 4.

- We incorporate our boundary detections into a framework targeting whole-object segmentation for unsupervised object discovery [170]. We employ current image matting techniques, driven by our boundaries, to produce a set of segmentation "hints" which can be combined to produce an affinity measure suitable for existing clustering/segmentation methods. See Chapter 5.

- We suggest low-level modifications to a popular feature-based recognition technique, Lowe's SIFT [106], allowing inclusion of mid-level boundary knowledge to produce high-level recognition which is more robust to variation in the background. We call this the Background and Scale Invariant Feature Transform or BSIFT [165]. See Chapter 6.

- As an additional contribution of this work, we have made our dataset of video clips and corresponding ground truth boundary labels available online for use by other researchers:
  `http://www.cs.cmu.edu/~stein/occlusion_data/`

An overall schematic of the work presented here is provided in Figure 1.7. The following chapters describe each of the processes in more detail, from low-level motion estimation and feature extraction, to mid-level reasoning, classification, and propagation, and finally to high-level segmentation and recognition. In the conclusion (Chapter 7), we will also present possible future extensions of this work.

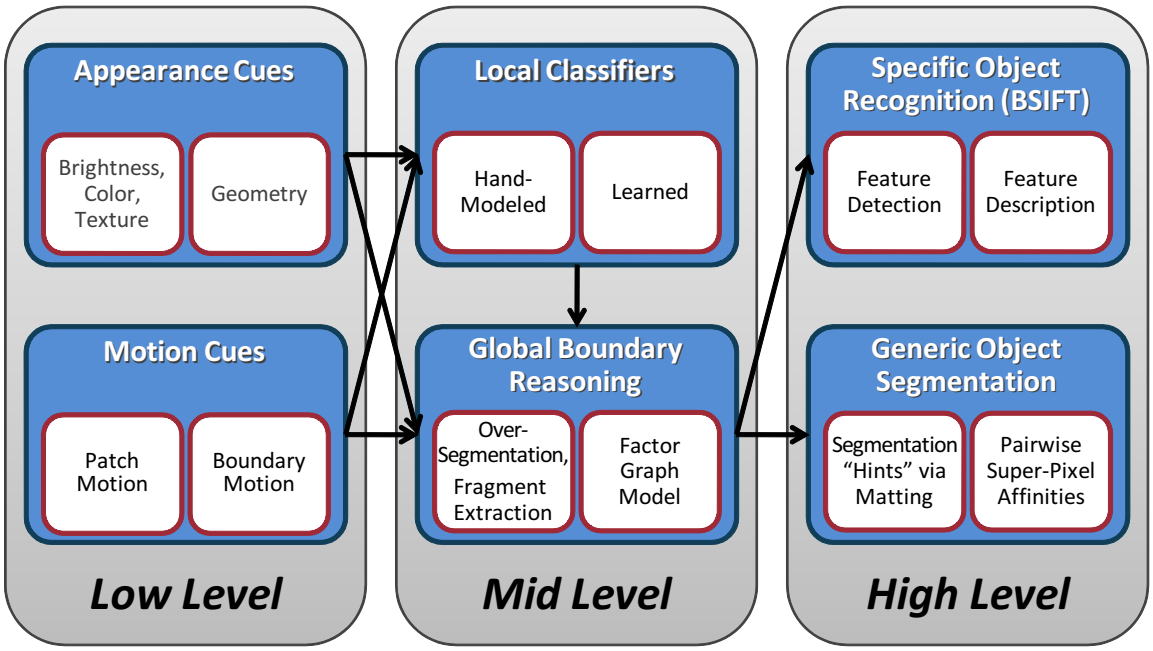


Figure 1.7: A schematic of the overall scope of this thesis, including low-, mid-, and high-level processing and reasoning.

Chapter 2

Local Motion Cues at Occlusions

Assume for a moment that we have detected a moving edge in an image. If that edge corresponds to simple texture on the surface of a single object, we would expect the patches on either side to exhibit motion consistent with the edge and with each other. If the edge's observation is the result of an occlusion boundary, we would expect, in general, the patch on the background side of that occlusion to move *in*consistently with the foreground and/or the edge itself. Note that boundaries in front of a uniform background or at which the background actually does move with the foreground, are not *locally* observable via this type of motion reasoning.

Thus there exist two types of motion inconsistencies at occlusion boundaries: differing observed motion on either side of the boundary and differing motion between the background and the boundary itself (which is assumed to be "attached" to the foreground object [70]). Therefore, in exploiting motion cues for the detection of occlusion boundaries, we are interested in two types of motion estimates: patch motion and boundary motion. The estimation of these motions is the subject of this chapter. The subsequent use of these motion cues for occlusion detection is covered in Chapters 3 and 4.

2.1 Patch Motion Estimation

A motion estimate for a patch of intensity data may be computed from the patch's spatio-temporal derivatives [107, 156, 175]. This idea is based on the brightness constancy assumption and forms the fundamental building block for many optical flow, tracking, and registration methods. Such estimation of image motion is a classical problem in computer vision (see [53] for a recent tutorial on optical flow). In general, the goal is to determine a motion vector (u, v) for each pixel or patch which indicates its displacement from one frame to the next. Here, we will consider several consecutive frames of video and compute *multi-frame* motion estimates. As compared to using only two frames, we find that using

multiple frames produces substantially more robust estimates that are more discriminative for our classification task (see Chapter 3).

For our purposes, since we are interested only in motion estimates near edges (rather than a dense flow field), we will choose patches of data P_L and P_R on either side of each detected edge pixel, as shown in Figure 2.1. In addition, because we have an estimate of each edge pixel's orientation, θ, we can align those windows to the edge in order to prevent the collection of information across a potential occlusion boundary. This technique is related to adaptive/multiple-window techniques, *e.g.* in stereo vision [73, 84] and in occlusion reasoning [166, 168].

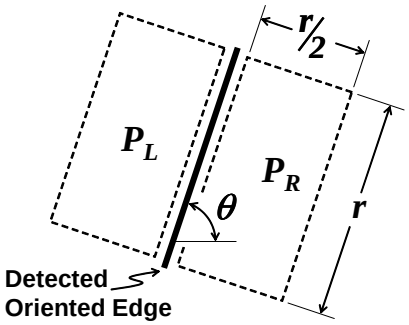


Figure 2.1: Patches for motion estimation aligned to an oriented edge.

Spatio-*temporal* alignment to moving edges can also be performed as in [168]. In this case, patches are aligned not only to the spatial orientation of the detected edge, but also skewed to match its estimated motion path through time (see Section 2.2).While this additional alignment is intuitively helpful in reducing the potential corruption of foreground spatio-temporal derivatives with background data (and vice versa), we have found that it is not strictly necessary. Careful implementation of the multi-frame optical flow estimation procedure, as described below, seems to be sufficient — and is not additionally dependent on accurate initial boundary motion estimates.

Given a set of frames $\{I^{(n)}\}_{n=-N}^{N}$ our goal is to find the translational motion, with components u and v, which best matches a patch of pixels P in the central reference image, $I^{(0)}$, with its corresponding patch in each of the other images, $\{I^{(n)}\}_{n\neq 0}$:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \arg\min \sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} w(x,y) \Big(\underbrace{I^{(n)}(x,y) - I^{(0)}(x-nu, y-nv)}_{r} \Big)^2. \quad (2.1)$$

Note that this implicitly assumes constant translation for the duration of the set of frames, which is most reasonable over brief time periods.[1] We employ Gaussian-shaped weighting functions, $w(x,y)$ and $h(n)$ (centered at the middle of the patch in the reference frame and with associated bandwidths σ_h and σ_w), to decrease the contribution spatially and temporally of pixels distant from the center of the reference patch.

Aggregation of patches of data near occlusion boundaries is problematic and addressing this problem specifically for optical flow estimation is the subject of extensive research,

[1] First performing a global, translational stabilization of each frame in the clip to the reference frame can also help make the data better adhere to this constant-translation assumption — see Section 3.1.1 and Figure 3.4 — in addition to removing large motions and focusing subsequent processing on the more subtle relative motions that are most important for our task, as discussed in Chapter 3.

including multiple motion estimation, robust estimators, line processes, and parametric models [20, 53]. Recently, impressive results computing dense flow fields in spite of significant occlusion boundaries by using a variational approach and bilateral filtering were demonstrated in [187].

For our work, we iteratively estimate u and v using a multi-frame, Lucas-Kanade style differential approach to find the minimum of (2.1). Iteration is employed because the use of finite-sized patches may prevent us from finding the full translation vector (u, v) in one application of least squares. Thus, to solve for the update, (u', v') to the current estimate (u_k, v_k) at iteration k, we replace the residual (or error) term, r, in our objective function (2.1) by

$$r = I^{(n)}(x + nu_k, y + nv_k) - I^{(0)}(x - nu', y - nv'). \tag{2.2}$$

Here, the position of the patch in frame n has been adjusted by the previous translation estimate, which can be initialized to zero for the first iteration, when $k = 0$. According to the classical brightness constancy assumption and a first-order approximation we can approximate the second term as

$$I^{(0)}(x - nu', y - nv') \approx I^{(0)}(x, y) - nu' I_x^{(0)}(x, y) - nv' I_y^{(0)}(x, y), \tag{2.3}$$

where $I_x^{(0)}(x, y)$ and $I_y^{(0)}(x, y)$ represent the spatial derivatives of the reference frame, estimated by finite central differences. Substituting into (2.2) yields

$$r = \underbrace{I^{(n)}(x + nu_k, y + nv_k) - I^{(0)}(x, y)}_{I_t(u_k, v_k, n)} + nu' I_x^{(0)}(x, y) + nv' I_y^{(0)}(x, y)). \tag{2.4}$$

Finally, the objective function at each iteration becomes

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \arg\min \sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} w(x, y) \left(I_t(u_k, v_k, n) + nu' I_x^{(0)}(x, y) + nv' I_y^{(0)}(x, y)) \right)^2. \tag{2.5}$$

The corresponding linear least squares formulation is as follows (where $I_x = I_x^{(0)}$ and $I_y = I_y^{(0)}$ for clarity):

$$\underbrace{\begin{bmatrix} n_1 I_{x_1} & n_1 I_{y_1} \\ n_2 I_{x_2} & n_2 I_{y_2} \\ \vdots & \vdots \\ n_M I_{x_M} & n_M I_{y_M} \end{bmatrix}_{M\times 2}}_{A} \begin{bmatrix} u' \\ v' \end{bmatrix} = - \underbrace{\begin{bmatrix} I_{t_1}(u_k, v_k, n) \\ I_{t_2}(u_k, v_k, n) \\ \vdots \\ I_{t_M}(u_k, v_k, n) \end{bmatrix}_{M\times 1}}_{b} \tag{2.6}$$

$$A^T A \begin{bmatrix} u' \\ v' \end{bmatrix} = -A^T b \tag{2.7}$$

$$\underbrace{\begin{bmatrix} \sum n^2 I_x^2 & \sum n^2 I_x I_y \\ \sum n^2 I_y I_x & \sum n^2 I_y^2 \end{bmatrix}}_{G} \begin{bmatrix} u' \\ v' \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t(u_k, v_k, n) \\ \sum I_y I_t(u_k, v_k, n) \end{bmatrix}, \tag{2.8}$$

where the sums are taken over all M pixels within the patch, across all frames. (For reduced clutter, we have omitted the weights, $w(x, y)$ and $h(n)$, in this formulation.) The next translation estimate, (u_{k+1}, v_{k+1}), is computed from the previous estimate, combined with the current update:

$$(u_{k+1}, v_{k+1}) = (u_k, v_k) + (u', v') \tag{2.9}$$

As is well known, motion estimates near occlusion boundaries are prone to error [70], but the estimates near such boundaries are a crucial component of our approach, rather than outliers to be ignored or filtered out. Therefore, in addition to using the spatial and temporal weighting functions $w(x, y)$ and $h(n)$, we use an M-estimator on the residual term, r, in (2.1) — or (2.5) — rather than simply squaring it. This allows us to handle outliers in the temporal difference (or error) term more robustly by preventing them from having undue influence on the solution. The form of the M-estimator used in our work is

$$\rho(z) = \frac{z^2}{z^2 + \sigma^2}. \tag{2.10}$$

The resulting objective function,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \arg\min \sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} w(x, y)\rho(r), \tag{2.11}$$

is of course non-linear. In practice, however, we can continue to find a solution via a linear set of equations, as in (2.8), by employing the classical technique of Iteratively Re-weighted Least Squares (IRLS) [17, 39, 77, 102], as follows. We found a minimum of the original objective function in (2.1) by setting its derivative equal to zero, yielding our linear set of equations corresponding to (2.8),

$$\sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} 2\ w(x, y)\ \frac{\partial r}{\partial \mathbf{u}} r = 0, \tag{2.12}$$

where $\mathbf{u} = [\ u \quad v\ ]^T$. We can do the same for our new robust objective function, (2.11):

$$\sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} w(x, y) \frac{\partial \rho}{\partial r} \frac{\partial r}{\partial \mathbf{u}} = 0. \tag{2.13}$$

Trivially multiplying by $\frac{2r}{2r}$ (*i.e.* simply one) yields

$$\sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} w(x, y) \underbrace{\left(\frac{1}{2r} \frac{\partial \rho}{\partial r}\right)}_{\gamma(x,y,n)} \left(\frac{\partial r}{\partial \mathbf{u}} 2r\right) = 0. \tag{2.14}$$

From (2.10), we see that

$$\gamma(x, y, n) = \frac{1}{2r} \frac{2r\sigma^2}{\left(\sigma^2 + r^2\right)^2} = \frac{\sigma^2}{\left(\sigma^2 + r^2\right)^2}. \tag{2.15}$$

So we are simply left with the same linear set of equations as in (2.12), but with a new weighting term given by $\gamma(x, y, n)$:

$$\sum_{n=-N}^{N} h(n) \sum_{(x,y)\in P} 2\ w(x, y)\ \gamma(x, y, n) \frac{\partial r}{\partial \mathbf{u}} r = 0. \tag{2.16}$$

Thus we can compute the update (u', v') from (u_k, v_k) according to (2.8) as before. But we then *refine* that update within an additional loop, by iteratively adjusting the weights $\gamma(x, y, n)$ and re-solving (2.8), where the scale parameter, σ, of the m-estimator is determined at each iteration according to a classical result from the field of robust statistics [144, 198]:

$$\sigma = 1.4826\ \text{Median}(r). \tag{2.17}$$

Furthermore, we initially consider only $I^{(0)}$ and its two immediate neighbors. We then gradually increase the temporal window, initializing with the previous translation estimate, until finally considering all frames from $-N$ to N. Since the true corresponding patches in frames temporally distant from the reference frame could be quite far spatially from their initialized locations, especially when there is significant motion, this process prevents frames at extremes of the temporal window from pulling the solution to poor local minima of (2.1). Note that the same effect could also be achieved by gradually increasing the bandwidth of $h(n)$. This approach alleviates the need to initially align the patches spatio-temporally to the moving edge, as discussed above.

Finally, we place a prior on small motions, since the relative motions we seek are quite subtle (see Section 3.3). In practice, this amounts to adding a small value, inversely proportional to the expected variance of the motion components, to the diagonal of G in (2.8):

$$\begin{bmatrix} \sum n^2 I_x^2 + \frac{1}{2\sigma_u^2} & \sum n^2 I_x I_y \\ \sum n^2 I_y I_x & \sum n^2 I_y^2 + \frac{1}{2\sigma_v^2} \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t(u_k, v_k, n) \\ \sum I_y I_t(u_k, v_k, n) \end{bmatrix}, \tag{2.18}$$

where we use $\sigma_u = \sigma_v = 1$ in this work.

We can now estimate the motions of the patches on either side of each edge, $\mathbf{u}_L = [\ u_L \quad v_L\ ]^T$ and $\mathbf{u}_R = [\ u_R \quad v_R\ ]^T$, using the IRLS approach outlined above. We will discuss how these patch motions can be used in local occlusion detection in Chapter 3.

2.2 Boundary Motion

A second type of motion cue available related to occlusions is the motion of the boundaries themselves. One approach would be to attack this problem from a frame-to-frame matching or tracking standpoint [162]. Edges could be detected in one frame and then matched to detections in the next frame by solving a correspondence problem or using tracking approaches like those used in feature tracking [175]. Unfortunately, tracking (which typically relies on patches of local data) is notoriously difficult at edges and even more so at occlusion boundaries. Tracking at any edge suffers from the so-called aperture problem, which limits us to seeing only motion normal to the edge when considering only local information. More global reasoning could help resolve resulting ambiguities, but not without substantial computational machinery. If we try to group edges into more easily trackable chains [162] — a hard, often brittle problem in its own right — we can suffer from an early commitment problem.

At occlusion boundaries, the problem is even worse. By definition, the data on either side of the boundary belongs to different surfaces. Since these two surfaces can move independently, they can confound any tracking or matching approach which assumes all the local data moves contiguously. Thus it would be dangerous to employ local patches of appearance data to estimate motion as described above in order to estimate the boundary's motion by simple tracking. But without any appearance data, simple line segments are not distinctive enough to be matched reliably without again resorting to a more global approach such as [98]. Instead, we need boundary motion estimates derived independently from the motion of neighboring patches.

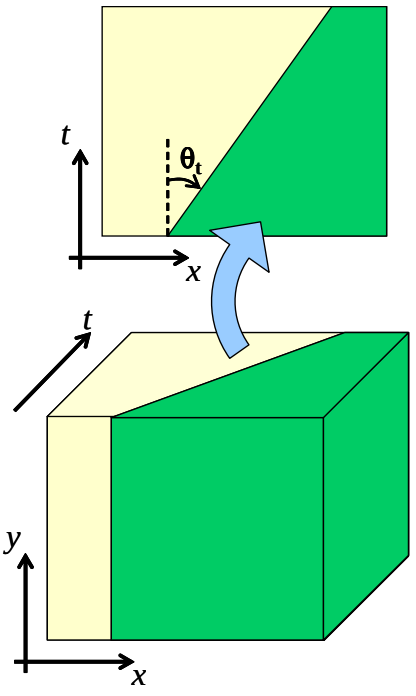


Figure 2.2: A moving edge sweeps out an oriented path in a temporal slice of the video volume. The speed of the edge is given by $tan(\theta_t)$.

On the other hand, consider treating the video data as a spatio-temporal volume, rather than processing individual frames separately. A moving edge traces out an oriented path along the temporal dimension of the data [1, 23, 68, 182], as depicted for a simple vertical edge in Figure 2.2. The tangent of the angle of this path, $tan(\theta_t)$, corresponds to its speed. So applying an oriented edge detector to a *temporal* slice of data would detect edges not at some spatial orientation in the image,

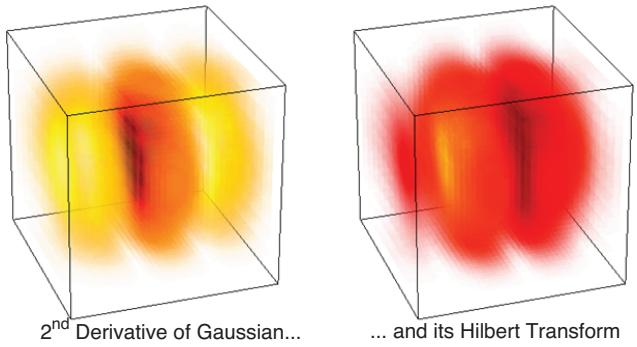


Figure 2.3: A quadrature pair of spatio-temporal filters.

but at speeds corresponding to the orientation of the detector. Combining detectors for spatial orientation and motion, it is possible to design a 3D spatio-temporal detector which simultaneously detects edges at a given spatial orientation *and* a given speed of motion normal to the edge. The spatial orientation and speed of such a detector will correspond to the orientation of a spatio-temporal *plane* in the video volume, rather than an oriented line in an single frame or temporal slice. Note also that this reasoning only requires a consistently different *appearance* on either side of the edge as it moves, but it does not make any assumptions about the consistency of *motion* on either side of that edge (which still may or may not be an occlusion boundary).

Having cast the problem of motion estimation at boundaries into one of spatio-temporal edge detection, we can rely on extensions of existing edge-detection techniques, of which there are two main types: filtering-based and patch-based, with the former being the most common.

2.2.1 Filtering-based Edge Detection

At their most basic level, filtering approaches model an edge as a step function of image intensity. Therefore, edges can be detected by looking for positions in the image with high derivative (or zero second derivative), *e.g.* using the response of the image to a derivative-of-Gaussian filter. Other more complicated filters have been designed which can handle somewhat more complex models of edges as well [12, 132]. By using steerable filters [57], it is also possible to look for peak response with respect to spatial orientation. Furthermore, all of these approaches can be made computationally efficient by exploiting the use of separable filters.

For video data, 3D spatio-temporal filters, still based on Gaussian derivatives, were designed in the classical work of [1, 68]. These filters have not only a spatial orientation but

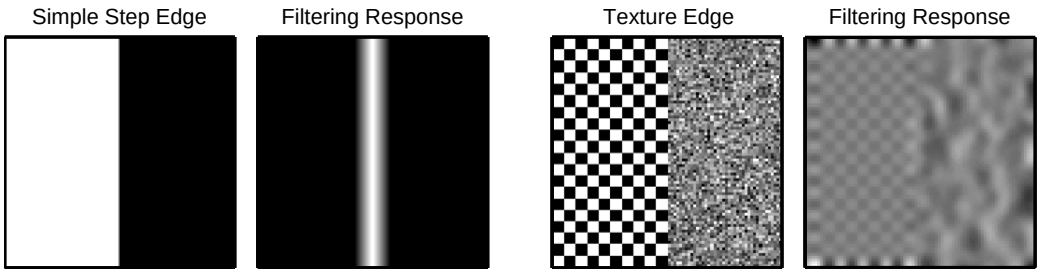


Figure 2.4: While filtering-based methods respond well at simple step edges (left), they fail at more complex texture edges (right). Here the mean intensity is exactly the same on either side of the edge, so linear filtering completely fails to respond.

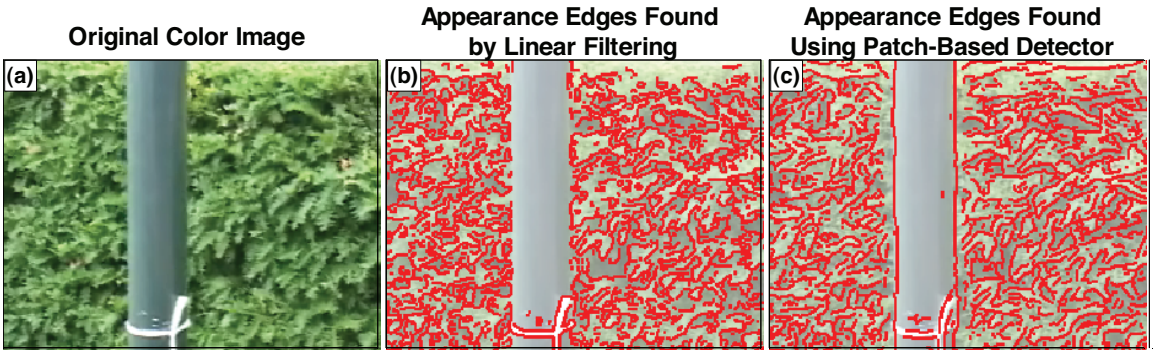


Figure 2.5: A comparison of traditional filtering-based edge detection (b) to a patch-based detector (c). Only the patch-based detector fires consistently on the edges which lie on occlusion boundaries of the pole.

also a temporal one. An example quadrature pair of spatio-temporal filters can be found in Figure 2.3. Thus, the combined *spatio-temporal* orientation of the maximum response of such a filter indicates an edge's orientation and speed simultaneously. Their potential for efficient implementation via separability was recently reported [40].

Unfortunately, filtering performance is often poor at edges which do not conform to the particular intensity profiles for which the filters were designed. An extreme case of this problem is illustrated synthetically in Figure 2.4. In practice, this is a problem at boundaries bordering textured or cluttered regions, as seen in Figure 2.5(b), where a filtering-based method fails to consistently detect edges along the sides of the pole. Note that these edges are also the ones providing the most evidence of occlusion since the texture and clutter — *i.e.*, the leaves of the bush behind the pole — can offer the indicative motion information. Thus we need an edge detector which will perform well in such cases.

2.2.2 Patch-based Edge Detection

Instead of using classical filtering, one can also take a more general, non-model-based view of edges. In this case, the profile of the edge is not assumed to have any particular form. Locally, an edge is instead defined to be a line segment which divides a patch of data into halves which contain significantly different *distributions* of some property (*e.g.* brightness, color, texture, *etc.*). By comparing histograms computed in each half of a circular patch, such approaches have been shown to produce good results even on textured/cluttered data [114, 116, 148, 186]. For example, consider the result in Figure 2.5(c), especially comparing the performance along the pole's boundaries versus the result in (b), obtained using filtering.

Naturally, we would like this sort of improved performance for a *spatio-temporal* edge detector as well, especially since the occlusion boundaries in which we are interested are often observed at locations where one or both sides of the boundary are textured or cluttered. First, we will describe the patch-based approach to edge detection in single images in more detail before extending the method to video in the next section.

Consider a circular patch of pixels extracted from an image. The radius of this patch is related to the scale of the edges we wish to detect. To evaluate the possibility that an edge passes through the center of this patch at a particular orientation, we split the patch in half along a line at that orientation. We estimate the distribution of the data — be it raw intensity, color, textons, *etc.*— on either side of the proposed line by binning each half of the patch into two histograms. The more these two histograms differ from one another, the more likely an edge exists in this patch, at this orientation. By rotating the dividing line about the patch center, we can evaluate edge strength for various orientations.

There exist several possible methods of measuring the difference between two distributions. The simplest is to compute Euclidean distance between them by evaluating bin-wise count differences. Quantization effects can make this approach a poor choice. Other approaches include the Mallow's or Earth Mover's Distance (EMD) [101, 111] as used by [148] or dynamic time warping (DTW) as used by [116], which each take the distance between bins into account. While DTW can be made significantly more efficient than EMD [116], both come at an unacceptable computational cost, especially once we transition to video.

Here, we follow the approach suggested by [114] and use the χ^2-Distance,

$$\chi^2(g,h) = \frac{1}{2}\sum_{i=1}^{N_{bins}} \frac{(g_i - h_i)^2}{g_i + h_i} \tag{2.19}$$

between binned kernel density estimates g and h. This normalizes the difference between the bins by their total count. Using a small Gaussian kernel when binning the data helps nearby bins interact when differencing, thus avoiding quantization effects and approximating EMD

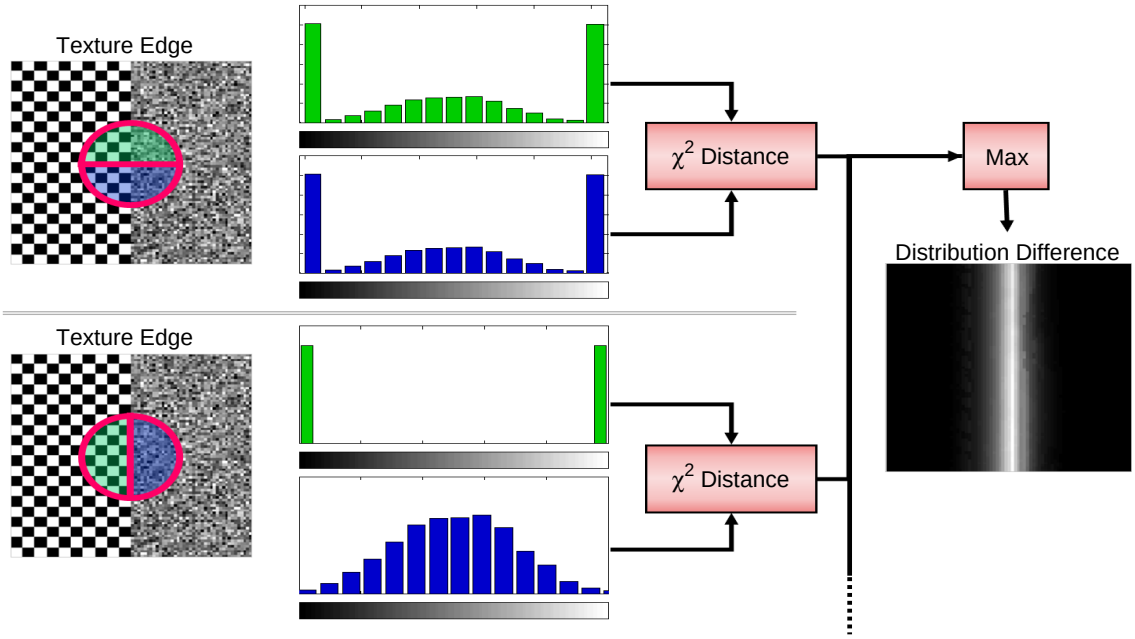


Figure 2.6: Schematic describing the detection of of an oriented texture edge by comparing non-parametric distributions of intensity with a patch-based detector. Note that a detector based on linear filtering fails to find this same edge, since it effectively compares only mean intensity (see Figure 2.4).

to some degree. Note that instead of using a kernel each time we add a pixel to a histogram, we can keep simple integer counts and smooth the result with that kernel to achieve approximately the same result. Thus, we can replace the numerator and denominator of (2.19) by smoothed versions of the histogram difference and sum, $(k_\sigma * (g-h))^2$ and $k_\sigma * (g+h)$, respectively. Here, k_σ is a Gaussian smoothing kernel with standard deviation σ. The various parameters required, including number of bins, kernel bandwidth, *etc.*, can be learned from training data [114].

A diagram describing the operation of an oriented, patch-based edge detector is provided in Figure 2.6. Here, the texture edge is successfully detected since this patch-based approach employs histograms to retain higher-order, local statistical information at the edge. By contrast, simple linear filtering approaches, such as the example in Figure 2.4, only use the mean of the local data distributions, thereby throwing away the higher-order statistics which actually indicate the presence of an edge in more complex cases.

Implemented naïvely, there is substantial redundancy in binning data from each half of a circular patch into two histograms as we vary orientation. As shown in Figure 2.7 (a)-(b), a substantial portion of each half covers the same data when we rotate the dividing line from one orientation to the next. Furthermore, the slice which should be added to one half is the same slice which must be removed from the other half, and vice versa. Therefore

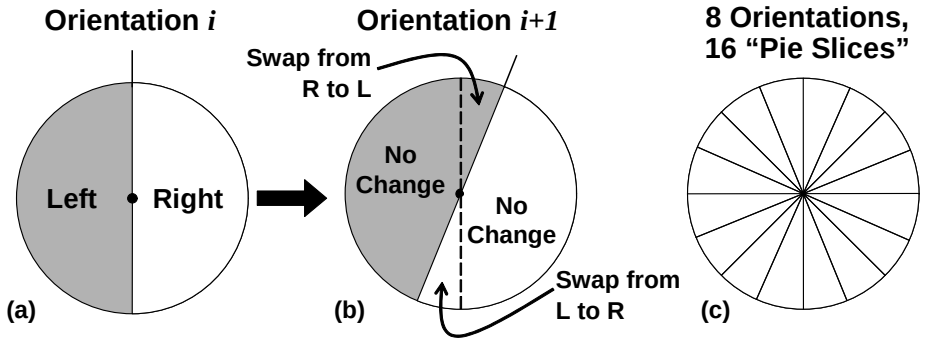


Figure 2.7: Circular patch used for patch-based edge detection. (a) The left and right halves of the patch at one orientation. (b) Only a fraction of the pixels in the patch will change halves as we rotate to the next orientation. (c) "Pie slices" for an 8-orientation detector.

we can first divide the patch into a "pie" of slices, as in (c), and bin the data in each slice individually. Then we can sum the histograms of appropriate slices for each half, swapping slices between halves as we rotate the dividing line [114].

2.2.3 Extension to Video

The use of 2D patches in images has become common practice for object recognition tasks ([3, 46, 106, 176] to name a few), but recently several researchers have begun to explore the utility of spatio-temporal, *volumetric* patches extracted from video data, mostly in the context of event/action detection and recognition [88, 94, 156]. In this section we describe the use of spatio-temporal patches for the low-level vision task of simultaneously detecting edges *and* their motion. Our approach extends to the spatio-temporal domain ideas from existing spatial (2D) patch-based edge detectors described in the previous section, which offer improved performance in textured/cluttered regions versus classical filtering techniques. Since occlusion is most likely visible in such regions of an image, we need an edge detection method which works well in such cases.

Patch-Based, Speed-Only Detector

Assume we first detect edges spatially using the circular patch-based method described above. To then determine the speed of the detected (oriented) edges, we can use a *cylindrical* detector, analogous to the spatial, circular one, but rotated into the temporal dimension and aligned to the edge's orientation. By dividing the cylinder into halves with a plane, we can detect the speed of the edge's motion in the direction normal to its orientation. (As is always the case in local edge reasoning, we suffer from the aperture problem, limiting us to normal speed estimates as opposed to 2D velocity vectors.) In Figure 2.8 we see a

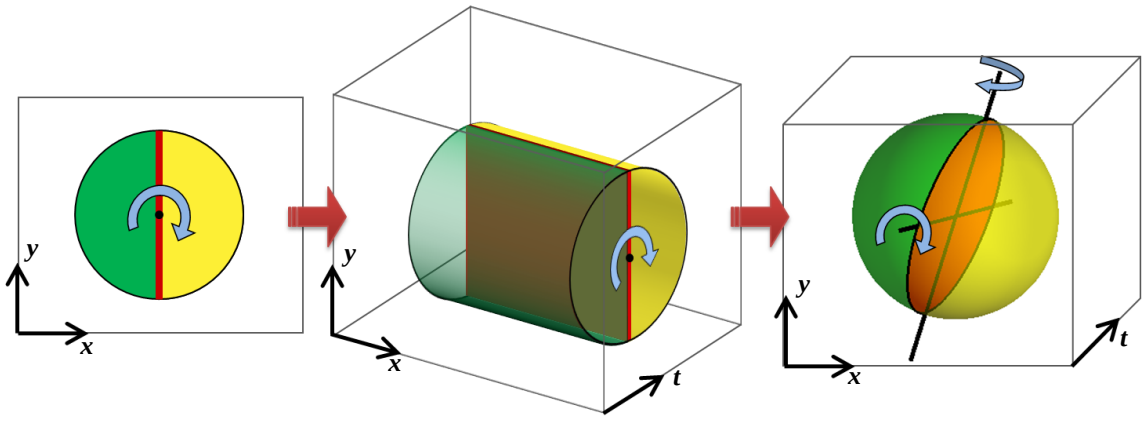


Figure 2.8: Three patch-based edge detectors. From left to right, we have a simple oriented edge detector for a single frame, a cylindrical edge-speed detector for video data, and a spherical detector capable of detecting spatial orientation and normal speed simultaneously.

standard patch-based, oriented spatial edge detector for use in a single frame to the left and the analogous cylindrical, edge *speed* detector in the middle.

Simultaneous Speed-Orientation Detector

In the previous approach, we detected orientation first with a spatial, circular detector, and then used a temporal, cylindrical detector to estimate speed. To detect the two simultaneously, we can use a *spherical* spatio-temporal patch of pixels, as shown at the right of Figure 2.8.[2] We will split the patch into two hemispheres using an oriented plane. The intersection of this plane with the image plane is a line which corresponds to the proposed edge's spatial orientation, as in existing 2D patch-based oriented detectors. The projection of the dividing plane's normal onto a second plane which is normal to the orientation line corresponds to the motion of the proposed edge in the direction normal to its orientation. This is illustrated in Figure 2.9. Thus we have two degrees of freedom to specify the dividing plane for the sphere: the spatial orientation of the edge followed by the normal motion of the edge with respect to that orientation.

This method samples spatial edge orientations, θ_s, directly. The normal motion speed can be computed from dividing plane's temporal rotation angle about this orientation axis according to $tan(\theta_t)$. (The same is also true for the cylindrical detector above.)

We note that there are two special cases of dividing planes. First, planes which are orthogonal to the image (xy-plane) represent a non-moving edge at some spatial orientation.

[2]The radius of our patch in the temporal dimension is not restricted to be the same as that used for the spatial dimensions (nor are the units of those dimensions even comparable). Thus, we may in fact have an ellipsoidal patch of data rather than sphere. But we will consider a spherical volume for simplicity in the discussion, without loss of generality.

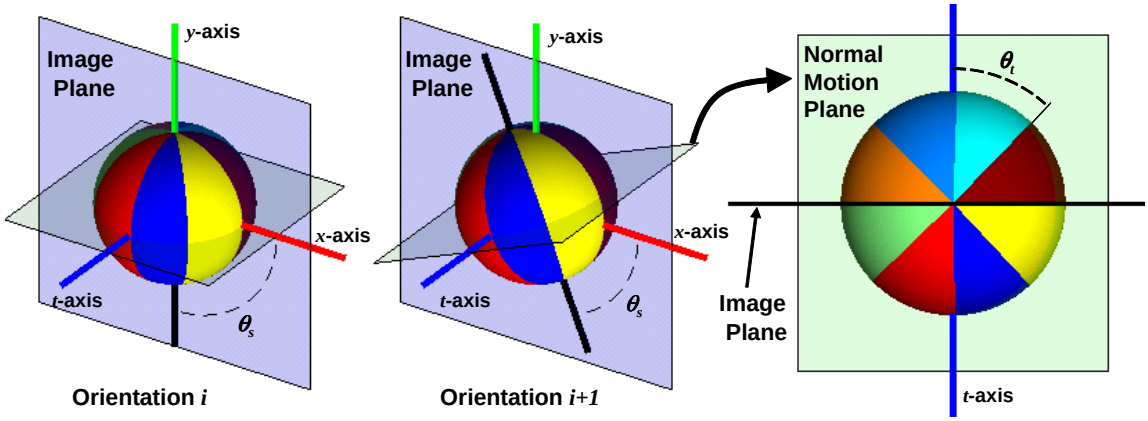


Figure 2.9: Our volumetric patch can be divided into histogram sections and oriented to produce a detector for a given set of spatial orientations and normal motions. See text for details.

Such planes are a direct extension of simple oriented-edge detection to video and are essentially using the extra temporal information as additional support for determining oriented strength while combating imaging noise. Second, planes which are parallel to the image plane correspond to an edge moving "infinitely" fast through the local spatio-temporal volume. These planes have completely ambiguous spatial orientation.

Once again, computing the histograms of data in each hemisphere for each plane orientation would involve significant redundancy and inefficiency (especially now that we are working with volumetric data). Unfortunately, directly extending the "pie-slice" idea from the circular or cylindrical cases to this new spherical detector is non-trivial. It is not a simple matter to divide the sphere into a regular set of sub-volumes which can easily be combined to form the set of hemispheres we require. It may be possible to use an icosahedron to uniformly sample normals from the unit sphere for our dividing planes [13] (rather than specifying a set of spatial orientations and normal motions). Then a tessellation of the sphere's volume could potentially supply the solid angle sub-volumes analogous to our 2D pie slices. These are themselves difficult problems, though their solutions could certainly be pre-computed.

We have instead chosen a simpler approach. We choose a vertical edge as a canonical, initial orientation. For zero motion, we divide our sphere in half along the y-t plane to detect this edge. Rotating this dividing plane by θ_t about the y-axis will sample various (horizontal) motions of this vertically-oriented edge. Here we can use a direct extension of the 2D slices into 3D: we divide the spherical patch into sections like those of an orange, oriented along this vertical edge, as in the left diagram of Figure 2.9. Combinations and swaps of these sections as were performed with the pie slices in the 2D case can be used to

construct any hemisphere, allowing us to sample motions *for this spatial orientation*.

For the next spatial orientation, we rotate this set of volume sections by θ_s about the t-axis, as in the middle of Figure 2.9. This results in a new set of sections that can be used to determine the motion strength *for this new spatial orientation*. We can pre-compute which voxels change section membership due to the spatial rotation by θ_s, allowing us to efficiently update each section from one spatial orientation to the next. As long as we always evaluate the set of orientations in the same order, as is naturally the case in practice, we can save substantial redundant binning — particularly as the radius of the sphere increases.

2.2.4 Experiments

In this section, we compare our results for determining instantaneous edge orientation and motion using histograms built over a spatio-temporal patch to results using a classical filtering approach.[3] For the filtering results we use a steerable, oriented edge filter which can detect composite intensity profiles [57, 132], but extended to a 3D spatio-temporal version [1, 68]. Its form is the sum of squared responses of a quadrature pair, where the even filter is the second derivative of the Gaussian (with the derivative taken across the putative moving edge), and the odd filter is its Hilbert transform, as shown in Figure 2.3. The filter kernel is set to the same size as the spatio-temporal patch used for our method, and the standard deviations are chosen to provide $\pm 3\sigma$ of support along each axis.

We also suppress non-local maxima of edge strength within the image (with respect to the edge normal) and then fit parabolas to edge strengths along the edge normal to determine the edge locations with sub-pixel precision [132]. Finally, we use a hysteresis threshold to remove noisy detections. We scale the edge strengths such that the maximum is one and use the same thresholds for either approach.

In the experiments that follow, we let $n_s = n_t = 8$. This corresponds to spatial orientations linearly sampled every 22.5° (as with other oriented edge detectors, such as [114]). Because the edge speed corresponds to $tan(\theta_t)$, the speeds are sampled non-linearly between 0 and ± 2.4 pixels/frame (and also including the "infinite" motion case, as discussed earlier). Interpolation is used both to achieve sub-pixel edge localization and sub-sample orientation/speed estimates. The temporal radius for the examples we consider here is three to five frames. Note that the use of interpolation allows us to recover speeds greater than 2.4 as well.

Synthetic and Semi-Synthetic Data: Due to the difficulty of obtaining ground truth

[3] A similar comparison can be achieved by simply taking the mean of the histograms in place of linear filtering. Here, however, we show results compared to the more popular filtering approach (which also effectively just takes the mean of the data) in order to emphasize the shortcomings of this commonly-used technique in our domain.

motion information for arbitrary video data, we will first compare results on synthetic data where ground truth is known. Note that there are three quantities being estimated for each edge: strength, orientation, and normal speed. Here, having "ground truth" means that we know the exact motion in the image and thus the normal speed at the edges. The concept of edge strength is qualitative and thus we do not have a means (nor need) for comparing to any ground truth. For orientation, we know ground truth of simple horizontal and vertical edges of course, but pixel-wise ground truth orientation for arbitrary curves is more difficult to label — though fairly easy to evaluate qualitatively.

The first sequence, whose middle frame is shown on the left side of Figure 2.10, consists of eight frames depicting a textured square translating up and to the right in front of a differently-textured background. The square moves exactly 2 pixels along each axis between every pair of frames. Note that this speed was intentionally chosen *not* to be in the set of sampled speeds. In Figure 2.11, edge strength maps are shown (after non-local maxima suppression and hysteresis thresholding) for filtering (top) and our method (bottom). Clearly, only our approach can find the square's boundaries against the textured background. In the center we show corresponding maps of absolute normal speed, where we see that only those edges from the square are found to be moving. On the right we provide a close-up view of the corner of the square displaying oriented edges and normal motion vectors overlaid on a lightened version of the original data. (The color of the edges corresponds to their strength: magenta is high, cyan is low.)

Figure 2.10: Representative frames for the synthetic and semi-synthetic sequences.

We see that both approaches respond to some of the horizontal and vertical structure of the background texture, but the filtering approach totally misses the edges of the square itself, which are arguably more useful in this sequence. Both methods correctly label the background edges as not moving and the orientations are consistent. Our method, however, also has a strong response at the square's edges and estimates their motion and orientation quite well. Note that this is due to the higher-order statistical information captured by the histograms; there is no explicit preprocessing to handle the texture (*e.g.* using textons). For the square's edges, our method reports a mean normal speed of 1.8 (versus the correct value of 2.0) with a standard deviation of 0.2. The only comparison we can make for the filtering approach is for the few texture edges it found on the interior of the square. Since

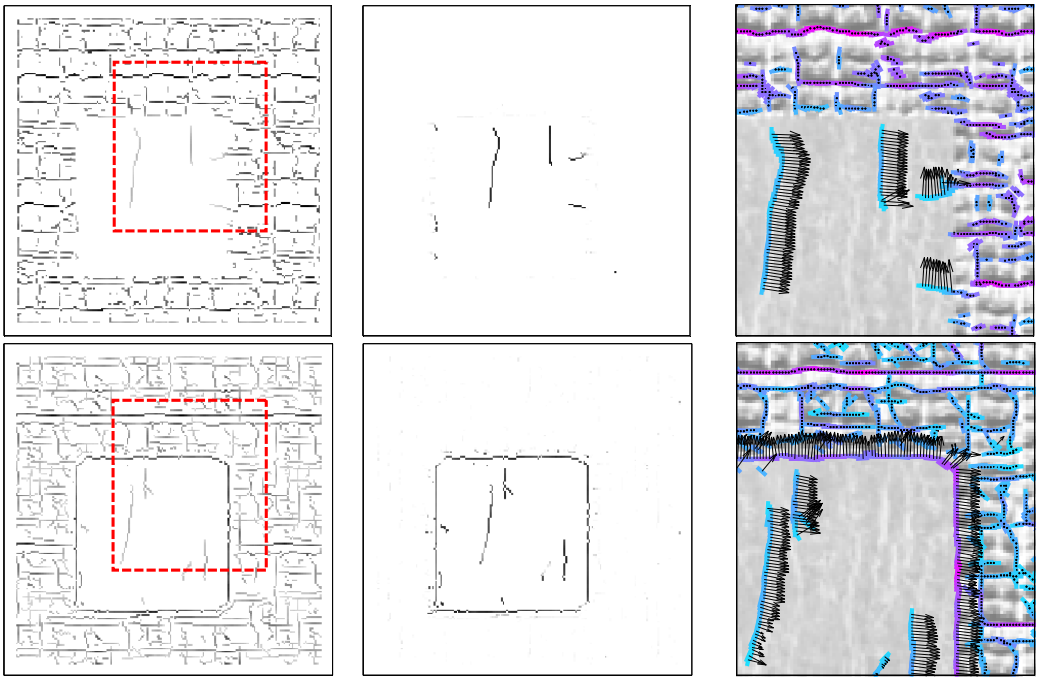


Figure 2.11: Results for the synthetic data. For standard filtering (top) and the proposed method (bottom), the columns depict non-local maxima suppressed edge strength (left), speed of edge motion normal to the edge (center), and a closeup view of oriented edges, color-coded by edge strength, with normal motion vectors for the region of the image indicated by the red dashed rectangle (right). Only our approach finds and estimates correct normal motion for the square's boundaries.

they are approximately vertical, their normal speed should also be near 2.0. The filtering approach reports a mean speed of 2.1 with standard deviation 0.16 for those edges.

The second sequence, a frame of which is provided on the right side of Figure 2.10, is a real image (obtained from [114]) which we synthetically translated to the right one pixel per frame for six frames, *e.g.* somewhat simulating camera motion. Its results, are shown in Figure 2.12. For the face and hair, both approaches do a good job detecting edges, but our approach better estimates the motion, particularly on the bridge of the nose. (Recall that we are only estimating *normal* motion of the edges, thus the motion vectors should *not* all simply point to the right nor be the same length. For example, a horizontally-oriented edge should have zero normal motion since the sequence moves entirely tangential to that edge.) We also see that the polka-dots of the woman's dress prevent the filtering approach from performing well at all; edges of her body and arms are missed almost completely. For our histogram result, notice the strong edges and consistent motion along the dress and arms.

If we assume the orientation of the edge detections are correct — and they qualitatively appear to be fairly accurate — we do know what the normal speed at each edge should be

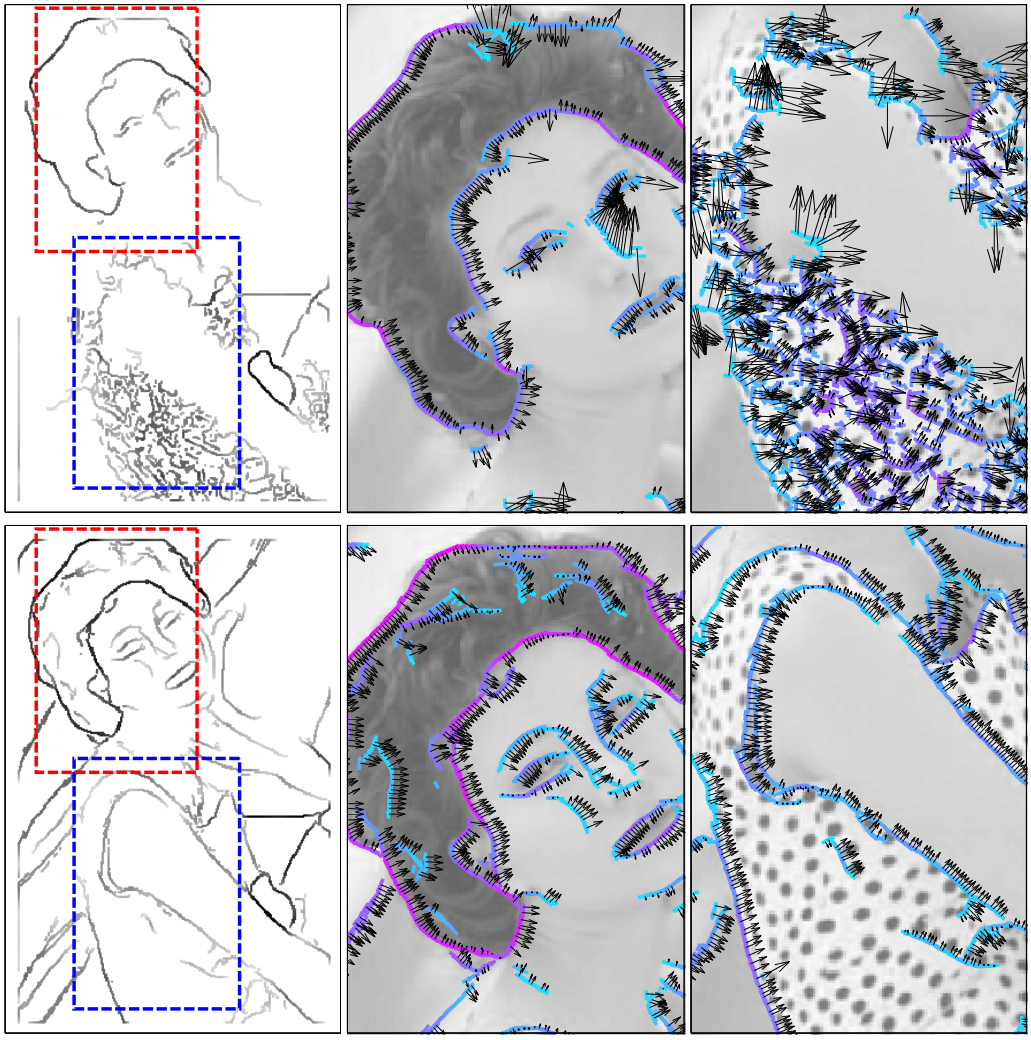


Figure 2.12: Results for the semi-synthetic data. The columns depict the edge strength and two close-up views of color-coded oriented edges and their normal motion vectors, overlaid on the data. The filtering approach (top) makes motion errors on the face and cannot handle the edges between the polka-dotted dress and smooth regions. Using our method (bottom) finds those edges, as well as ones from the face, and correctly estimates their normal motion, while ignoring spurious texture edges from the dots.

Figure 2.13: Representative frames for the real image sequences.

given the known image translation. Considering only the area around the woman's hair, for which strong edges are detected by each method, our mean absolute error in normal speed is 0.10 pixels/frame, with a standard deviation of 0.08. For the same region, the filtering approach has an average error of 0.23 with a standard deviation of 0.41, making its estimation of normal speed less precise and less accurate than using our proposed patch-based method. More importantly, however, the patch-based approach can detect (and thus meaningfully estimate the motion of) many more useful edges in the scene, *i.e.* those around the arms and edges of the dress. Note that the speed error statistics for *all* edges in the scene found using our method do not differ significantly from those reported for the region around the hair. The same cannot be said for the filtering result: the errors are clearly much worse on the woman's dress and body.

Real Data: Results for a real video sequence are provided in Figure 2.14. This sequence depicts a hand translating from right to left over a keyboard, as seen by a handheld, *nearly*-stationary camera (see Figure 2.13 for a representative frame). We expect to see relatively large normal motion for those edges perpendicular to the hand's motion and small motion vectors for the keyboard and background, due only to slight camera movements. For the simple, high contrast edges, either method performs fairly well. But the histograms used by our method are better able to distinguish the boundaries of the fingers seen against the clutter of the keys. This type of complex edge created by an approximately uniform foreground against cluttered background is not modeled well by the filtering approach. Also note that clean motion boundaries of this type could be quite useful for a layered motion segmentation since feature-based approaches will likely fail on such a uniform foreground object.

Note that improving the filtering approach's results is not simply a matter of choosing a better threshold on edge strength since approximately the same strength is reported for the boundaries of the fingers as the clutter. In fact, we consistently found in all our experiments

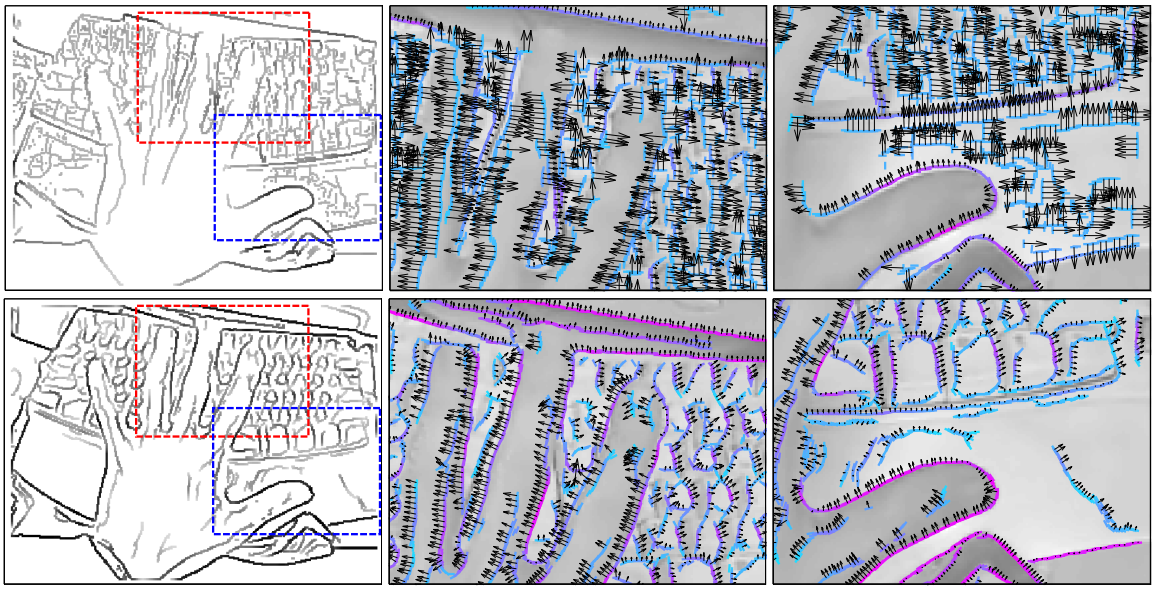


Figure 2.14: Results for the real sequence from the "hand" sequence of Figure 2.13. Our approach (bottom) finds stronger, more distinguishable edges for the fingers in front of the clutter of the keys, and its normal speed estimates are more consistent with the motion observed in the sequence, particularly for the background edges.

that our histogram-based method was much less sensitive to choice of threshold than the filtering technique. It also appears that using filtering over-estimates the speed of the hand slightly. For the edges detected on the keys, the orientations and speeds are completely wrong, whereas our approach's results are much more reasonable, in spite of the clutter. Finally, significantly better results were also not possible simply by selecting a better patch size (*i.e.* scale) for the filtering. Performance in some regions improved at the expense of others, depending on the chosen scale. Once again, the histogram method also appeared to be more stable with respect to selection of patch size, though a multi-scale detector could also offer improvement.

The second real sequence is shown on the right of Figure 2.13. It depicts a concrete bench in front of ivy clutter. In this sequence, the camera translates approximately in the direction of the long axis of the bench, while rotating slightly around the bench's center. The results, shown in Figure 2.15, again demonstrate our method's ability to detect edges bordered by a cluttered texture. For example, we detect a strong, consistent edge along the top side of the bench, while filtering performs very poorly there. We also see fewer bogus motion estimates by our method, *e.g.* around the markings on the surface of the bench and in the clutter. For the long edges of the bench, our method correctly estimates little to no normal motion (recall the camera translates approximately *along* these edges). At the close end of the bench, the correct normal motion is also shown. Even in the clutter, the motion

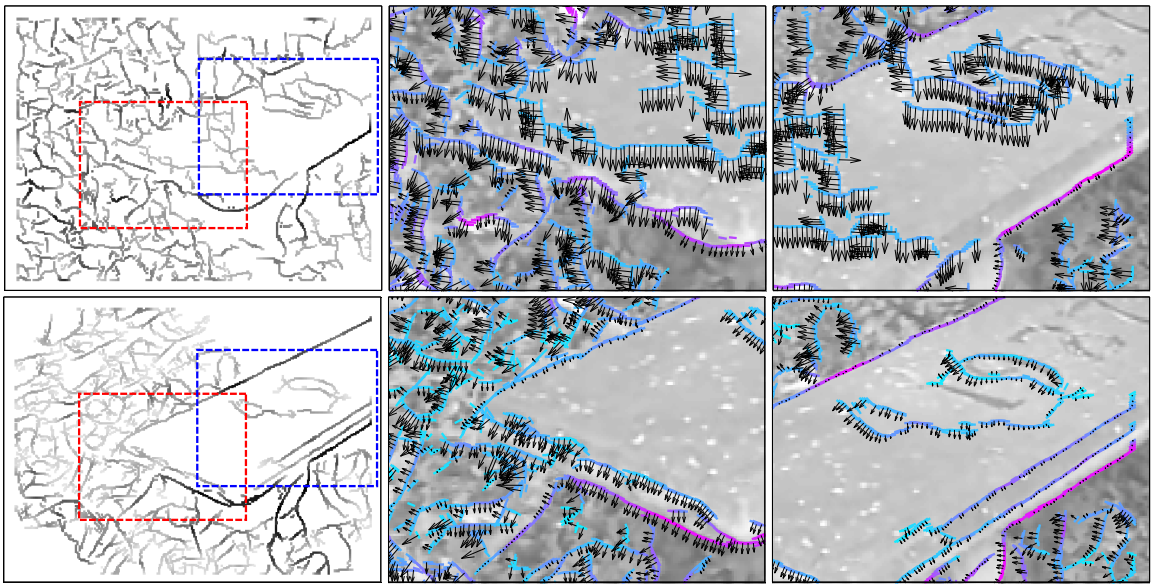


Figure 2.15: Results for the real sequence from the "bench" sequence in Figure 2.13. Our approach (bottom) finds the boundaries of the bench much more consistently despite the clutter of the ivy, and the normal speed vectors are consistent with the camera motion. The filtering approach proposes bogus motion estimates on the surface of the bench and in the background clutter.

estimates are consistent with the actual camera motion. Some of the motion estimates by the filtering approach do appear to be reasonable, for example those along the underside of the bench, where a strong edge is resolved. But in general, the result using our method is far superior.

2.3 Discussion

The relatively poor performance of energy-based linear filtering for the boundary-motion experiments in the previous section is consistent with the findings of Barron *et al.* [14]. Their results also suggest that phase-based approaches to optical flow [52] may perform better, though they do not focus explicitly on determining the normal motion of detected edges but rather the extraction of dense flow fields. An exhaustive comparison and characterization of optical flow techniques (either for estimation of patch motion as in Section 2.1 or boundary motion as in Section 2.2) is beyond the scope of this thesis. For more in-depth analysis, work such as [14] or [53] may be consulted. Instead, we have presented here a novel extension of current spatial edge detection techniques to the domain of boundary-motion estimation, and we have explained the particular method for multi-frame patch motion estimation we use. In the next two chapters, we will use each of these motion estimates in reasoning about occlusion boundaries, which is the primary focus of this document.

Chapter 3

Local Occlusion Detection

In the previous chapter we described the local motion cues which arise due to the depth discontinuity present at occlusion boundaries and provided methods for extracting those cues. In particular, we expect to see some form of motion *in*consistency in the vicinity of an occlusion boundary. For example, the foreground and background patches may move differently from each other, reflecting the parallax due to the relative distance to the two surfaces or the independent trajectories of the objects to which they belong. In other cases we may witness an inconsistency between the motion of the boundary itself and the background.

In some situations, the inconsistency can be fairly complex. In the case of a rotating cylindrical object, such as a soda can (as in [20]), the foreground side of the boundary apparently "disappears" as the can rotates (rather than moving according to some simple motion model), while the boundary and the background remain stationary.[1] In this case, our translational motion model from Section 2.1 will not accurately capture the true motion of the foreground patch. Still, the inconsistency between the foreground and background motion estimates — even if they are not individually correct — offers evidence of occlusion here. Thus we emphasize the subtle point that accurate motion estimation is not our goal. Instead we seek only to identify *relative motion inconsistencies*. In other words, relative motion information can provide a strong discriminative cue for occlusion even if the individual motion estimates themselves do not adhere to a generative model capable of completely explaining the observed image sequence.

In Section 3.1, we will explore the detection of occlusion boundaries based on such local relative motion cues.

Most occlusion boundaries are also visible as *appearance* edges, however, exhibiting changes in color, brightness, texture, *etc.*, in addition to relative motion cues. But many

[1] For in-depth analysis on the motion behavior of boundaries as objects rotate or viewpoint changes, see [95, 150, 153, 178].

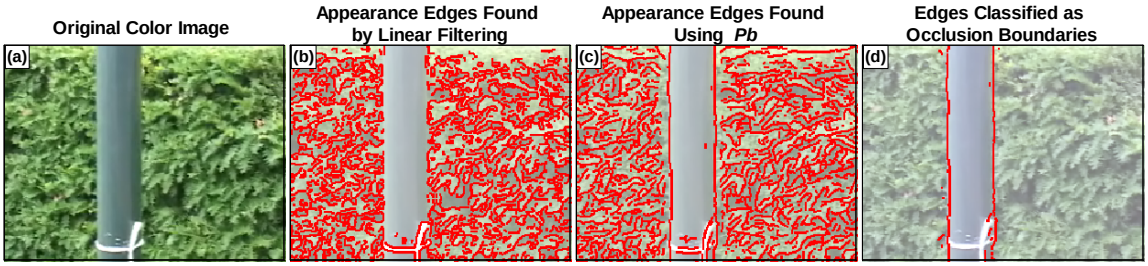


Figure 3.1: Only the patch-based edge detector (c) fires consistently on the vertical occlusion boundaries of the pole, giving subsequent classification (d) a chance of succeeding.

appearance edges arise solely due to surface markings or illumination effects and not occlusion boundaries. In addition, accurate local motion estimates may be difficult to obtain in many cases. Therefore neither motion nor appearance alone is likely to be sufficient for the detection of occlusion boundaries. In Section 3.2, we will combine multiple appearance cues, captured by state-of-the-art edge detectors, with local motion cues to show that *together* these distinct sources of information produce superior detection results to using either cue alone.

In particular, our goal is to determine the subset of initially-detected appearance edges that also correspond to occlusion boundaries, thereby framing our problem as one of classification. Thus we will first find "all" appearance edges with non-trivial response to a standard edge detection approach. We will then employ local motion cues (in Section 3.1) or local motion *and* appearance cues (Section 3.2) to classify those initial detections as occlusion boundaries or not. In Figure 3.1, we see again in (a) the same scene from the previous chapter, comparing the edges detected by filtering (b) to those found using a patch-based approach (c). In addition, we see an example of the type of classification we seek in (d), which was obtained by classifying the edges in (c) as occlusion boundaries. Such a result would be impossible to achieve using the filtering responses in (b), where edges were not consistently detected along the vertical boundaries of the pole.

3.1 Detection from Motion Only

Before moving to combined motion-plus-appearance classification in Section 3.2, we will first explore methods of detecting occlusion boundaries based solely on local motion information, *i.e.* using local spatio-temporal derivatives. For each edge identified in a scene, we would like to define a motion-based score which will be near zero for edges at which no occlusion is visible and one for edges where there is clear evidence of occlusion [168]. From two patches, P_L and P_R, aligned to either side of the edge, we estimate motion vectors $\mathbf{u}_L = [\ u_L \quad v_L\ ]^T$ and $\mathbf{u}_R = [\ u_R \quad v_R\ ]^T$, as described in Chapter 2, and compute their difference

$\mathbf{u}_d = \mathbf{u}_L - \mathbf{u}_R$.

While one can always find a solution to the least squares formulation for motion estimation in Equation (2.8), our "confidence" in the resulting estimated vector, (u, v), depends on the presence of strong spatial gradients within the patch. Loosely speaking, in a patch taken from a nearly-uniform region of the image, the uncertainty on the estimated motion will be much higher than the uncertainty for a patch containing strong gradient information (see Figure 3.2).

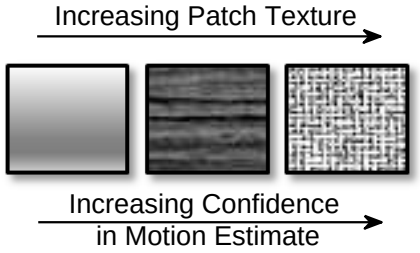


Figure 3.2: As horizontal and vertical texture within a patch increases, so too does our confidence in the estimated motion.

It is important that we take this uncertainty into account when comparing motion estimates from different patches [14, 161], especially since it is common in practice that one side of an occlusion boundary is nearly uniform, particularly for indoor scenes. In solving for (u, v) according to (2.8), the Hessian matrix G,

$$G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_y I_x & \sum I_y^2 \end{bmatrix}, \tag{3.1}$$

defines the shape of the solution surface around the particular local minimum of our least squares formulation in (2.1). Considered in a probabilistic sense, this local minimum also corresponds to the Maximum Likelihood solution for (u, v) under an assumption that the data likelihood, $\Pr(I_t|u, v)$, can be approximated by a Gaussian. When a prior is added, as in (2.18), we find the Maximum a Posteriori (MAP) solution. In either case, G specifies the covariance of that Gaussian distribution. Thus we can use this matrix to capture the uncertainty in our solution: when G indicates a flat solution surface around our estimated minimum or, equivalently, a Gaussian with large covariance, then we are less certain about the precise location of that minimum than if the surface or Gaussian is strongly peaked. It is worth nothing that the Gaussian assumption does not always hold [161]. In our experience, though, it is a good enough approximation to be of practical utility in a large number of cases. In addition, resorting to local sampling techniques in order to obtain better estimates of the true underlying distribution of $\Pr(I_t|u, v)$ can be computationally prohibitive [20].

Since we have two G matrices, one from each side of the patch, we will consider two motion estimates, $\mathbf{u}_L$ and $\mathbf{u}_R$, to be consistent if their difference is small according to *either* side's Gaussian model. Thus, we will take the maximum of the two consistency scores to accomplish this logical OR operation, as will be seen below. A matrix of spatio-temporal derivatives was also used by [156] to evaluate motion inconsistencies directly, as will be discussed — and to which we will compare — in Section 3.1.1.

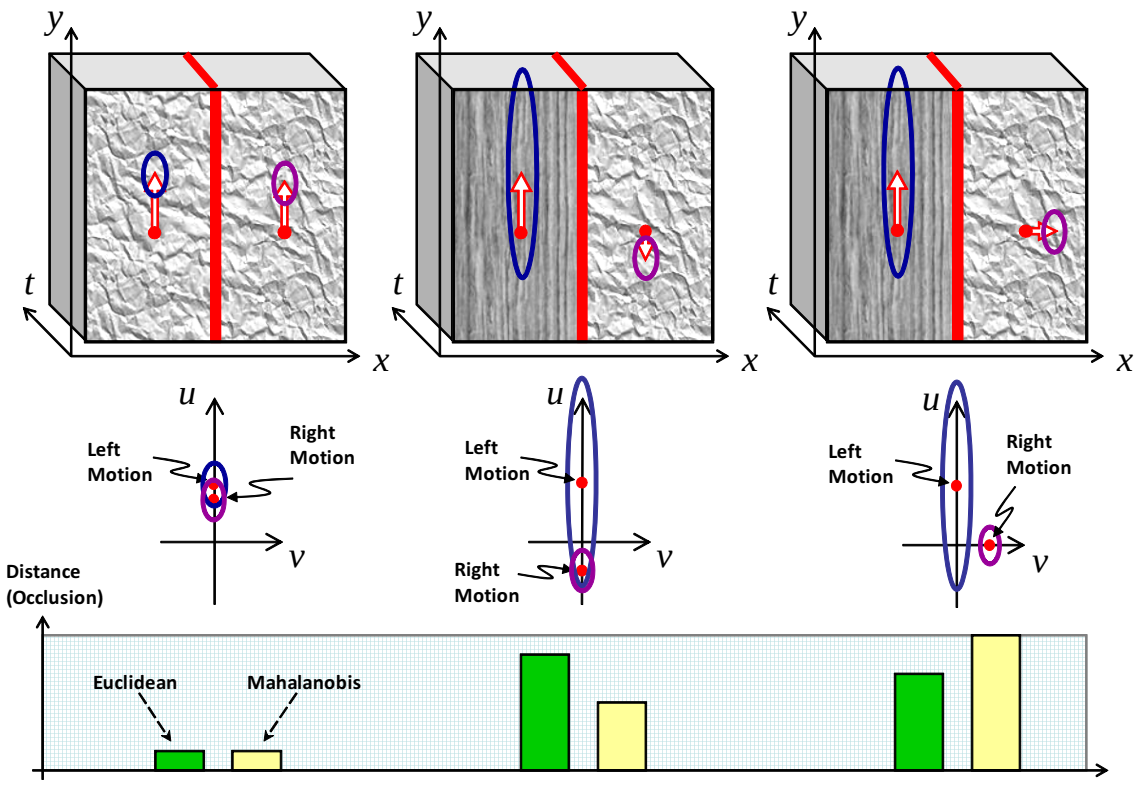


Figure 3.3: In the leftmost example, we are equally confident in the motion estimates on either side of the vertical edge because of the strong horizontal and vertical texture in each patch. In the middle and rightmost examples, however, we are less confident in the vertical motion on the left side of the edge. Taking these covariances into account via a Mahalanobis distance thus influences the occlusion scoring.

To make this discussion more concrete, consider the patch of data taken around a vertically-oriented edge shown in the left half of Figure 3.3. Assume we have used Eq. (2.8) to estimate the indicated motions on either side of the edge. In the left example, we are fairly confident in both motion estimated, as indicated by the covariance ellipses (which correspond to the Gaussian distribution indicated by G). Thus the distance between the left and right motions in (u, v) space (corresponding to the degree of occlusion) is similar whether or not we take the uncertainty into account.

For the middle example, however, we cannot be very sure of the estimated vertical motion on the left side of the edge due to the lack of significant horizontal texture there. Without taking the covariance into account, the difference in motion appears to be significant, which would indicate occlusion. But using G to normalize the distance computation (*i.e.* computing a Mahalanobis distance) allows us to be more "forgiving" along the vertical direction since we are less sure about the left side's motion in that

direction. Thus, this edge will receive a lower (albeit non-zero) occlusion score, as desired.

In the rightmost example, we see the same data but with different estimated motion for the right patch. In this case, we are still fairly confident that there is no horizontal motion in the left half, but we are also confident that the right half did undergo some horizontal translation. In this case, the two points' motions plotted in (u, v) space are roughly the same (simple Euclidean) distance apart as they were for the previous example, so without taking covariances into account, either example would have approximately the same degree of motion inconsistency (and thus the same occlusion score). But because the variance on the horizontal component of our motion estimate in the left patch is small, we will trust the disagreement in horizontal motions between the two sides and give the correct high occlusion score for this example. Note that this is true despite the fact that the difference in the horizontal components is actually *smaller* than the difference in the vertical components. These examples demonstrate that the covariance estimates help us utilize the most informative motion component when determining occlusion.

We can then compute the consistency of the left and right motion vectors according to each patch's spatial information, encoded in G_R and G_L. As discussed, we will keep the maximum of the the two scores. Finally, to convert to a measure of *in*consistency, and thus a measure of occlusion, we will subtract the result from one. The following equation captures this process:

$$score = 1 - \max\left\{\exp\left(-\frac{\mathbf{u}_d^T G_L \mathbf{u}_d}{2}\right), \exp\left(-\frac{\mathbf{u}_d^T G_R \mathbf{u}_d}{2}\right)\right\} \tag{3.2}$$

We can use this equation to score each edge as also being an occlusion boundary. Results using this approach are provided in the next section.

3.1.1 Motion-Only Experiments

Given one of our video clips, we can first find edge pixels and then estimate whether they exhibit occlusion within the image sequence. It is important to note that this approach is purely local and can therefore only reason about occlusion at an edge if visible occlusion actually occurs at that location within the observation's timespan. We may observe edges which lie in front of a uniform background for the duration of the video clip. These edges will not exhibit any motion cues of occlusion locally, and thus they will not be detected as occlusion boundaries by this method. The mid-level reasoning necessary to handle such situations is addressed in Chapter 4.

We initially pre-process the data to remove dominant motion. This prevents large observed motion due to camera movement from overshadowing the relative motion between surfaces (*i.e.* the parallax), which provides the evidence of occlusion and is usually quite subtle. In addition, stabilizing the motion makes the assumption of locally constant velocity

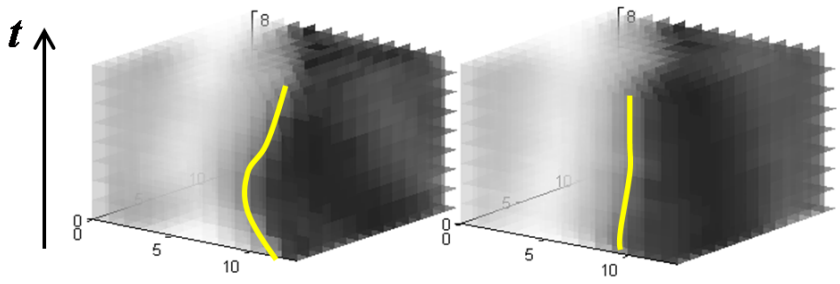


Figure 3.4: The same spatio-temporal patch of data around an occlusion boundary, shown before (left) and after (right) stabilization. The yellow line indicates the boundary's path through time, where the reference frame is at the center of the patch.

more realistic as more complex motions from camera shake, for example, are effectively filtered out. This step simply consists of a full-frame, translational registration of each frame in the sequence to the reference frame. See for example Figure 3.4, where we compare a spatio-temporal patch of data around an occlusion boundary before and after stabilization.

First we compute oriented edge strength at each spatial location using a patch-based edge detector as described in the Section 2.2.2. Non-local maxima of edge strength are suppressed [132], and the result is hysteresis thresholded. Left and right patches of data are then extracted on either side of each edge, aligned to its spatial orientation. The translational motion vectors are estimated for each patch as described. Finally, the measure of occlusion given by (3.2) is computed.

Shechtman and Irani have recently proposed an efficient method for computing motion (in)consistency from two spatio-temporal intensity patches *without* explicitly estimating motion itself [156]. Their work, however, was in the context of behavior recognition. We compare their rank-based score to our explicit motion comparison score in the results that follow. Each score is computed using the same initial edge detections and extracted patches.

For each occlusion scoring method (*i.e.* Equation 3.2 and the rank-based score from [156]), we vary the threshold on the score and count true and false positives. We can then generate and compare plots of pixelwise precision versus recall for each example using both scoring methods. (For these results, we dilate the ground truth boundary labels by one pixel when comparing to the score images in order to mitigate difficulties in precisely localizing boundaries when labeling images by hand.) Here *recall* refers to the number of pixels labeled as occlusion boundaries out of the total number of occlusion boundary pixels in the ground truth image that also received non-zero edge strength: we wish to compare the occlusion scoring methods rather than evaluating the edge detector. The *precision* is the number of those pixels that were in fact occlusion boundaries.

Figure 3.5 shows the middle frame of the same 8-frame synthetic sequence from

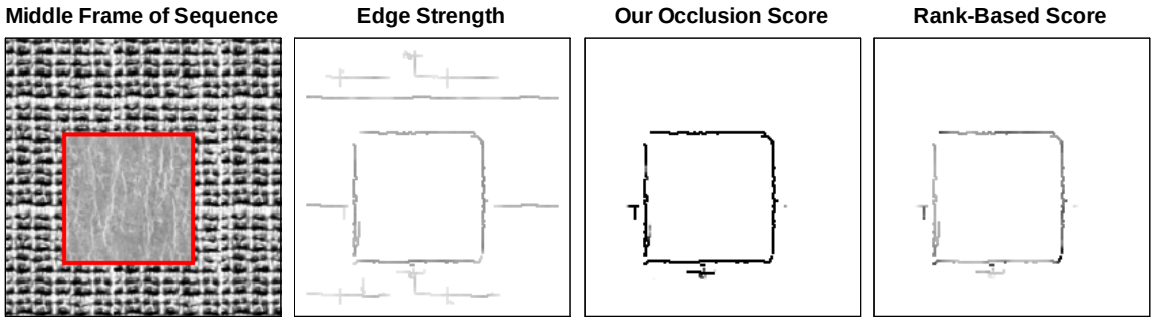


Figure 3.5: Results for a 6-frame synthetic sequence depicting a square moving 2 pixels up and 2 pixels to the right in each frame. Ground truth occlusion boundaries are displayed in red. For edge strengths and scores in this figure and in Figure 3.6, darker color corresponds to larger values.

Section 2.2.4, depicting a square translating up and right in front of a textured background. The hand-labeled ground truth occlusion boundaries are overlaid in red, and the edge strength is also provided. Using our method, we see that the boundaries of the square — and *not* the background's texture edges — correctly receive a high occlusion score. Using the rank-based score provides visually similar discrimination between edges and occlusion boundaries. When we compare the scores' precision-recall plots in Figure 3.7 (a), we see that ours is slightly superior in the high-recall regime.

The results for three real sequences from a handheld video camera, each 8-10 frames in length, are provided in Figure 3.6. Each shows qualitatively that we are generally able to distinguish occlusion boundaries from texture edges using the described approach. Furthermore, we see in Figure 3.7 (b)-(d) significantly better performance using our scoring approach, regardless of choice of threshold.

3.2 Combining Appearance and Motion

Though the initial results above for occlusion boundary detection using motion only are promising, we found that performance was not uniform across a wide variety of scenes. While motion is a strong cue for occlusion, we are still ignoring potentially useful information in the appearance cues present at boundaries as well. Furthermore, the hand-designed "score" function is not easily-adaptable to incorporate other cues directly. So instead of continuing with this type of scoring function, we will now turn to a more principled approach based on *learning* a classifier for each edge pixel, given labeled training data. This classifier should label each edge pixel as occlusion boundary or not, using a combined set of appearance and motion cues as input.

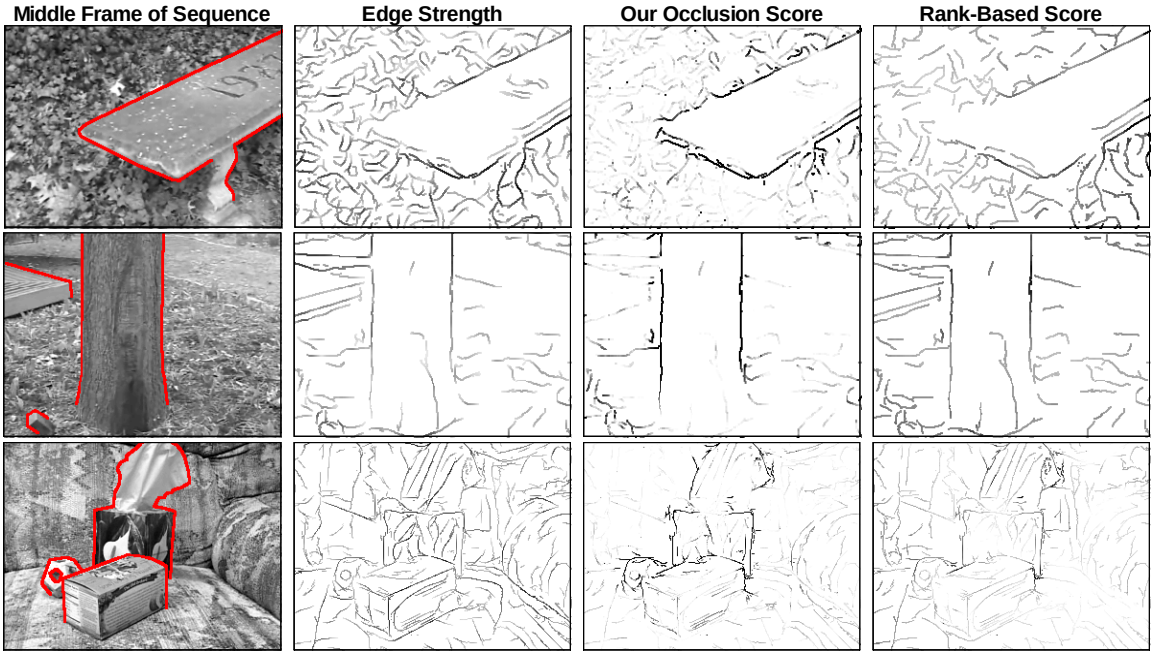


Figure 3.6: For handheld video sequences observing a bench in front of ivy (top) and a tree trunk (middle), and a set of objects on a textured couch (bottom), we see a representative frame of data with ground truth occlusion boundaries labeled in red, the detected edge strengths, and the two different occlusion scores, from left to right respectively.

3.2.1 Motion Cues

For our motion cue, we will continue to use the difference in estimated translational motion between the left and right patches, $\mathbf{u}_d = \mathbf{u}_L - \mathbf{u}_R$. In the results below we will simply use the Euclidean norm of the $\mathbf{u}_d$ vector to capture the relative motion between the surfaces on either side of a potential occlusion boundary. This distance, d, will serve as one of the cues to the classifier desribed in Section 3.2.3 below.

In these experiments with a learned local classifier, this Euclidean metric proved to be just as useful as a Mahalanobis distance or the score function in (3.2). This is likely due to the difficulty of obtaining in practice good estimates of the necessary covariance information on the motion components, without resorting to expensive sampling techniques [20]. Using the Hessian, G, from (2.8) directly as described above is a start, but appears not to be sufficient. We will, however, return to the idea of incorporating motion uncertainty into our classifiers in Chapter 4.

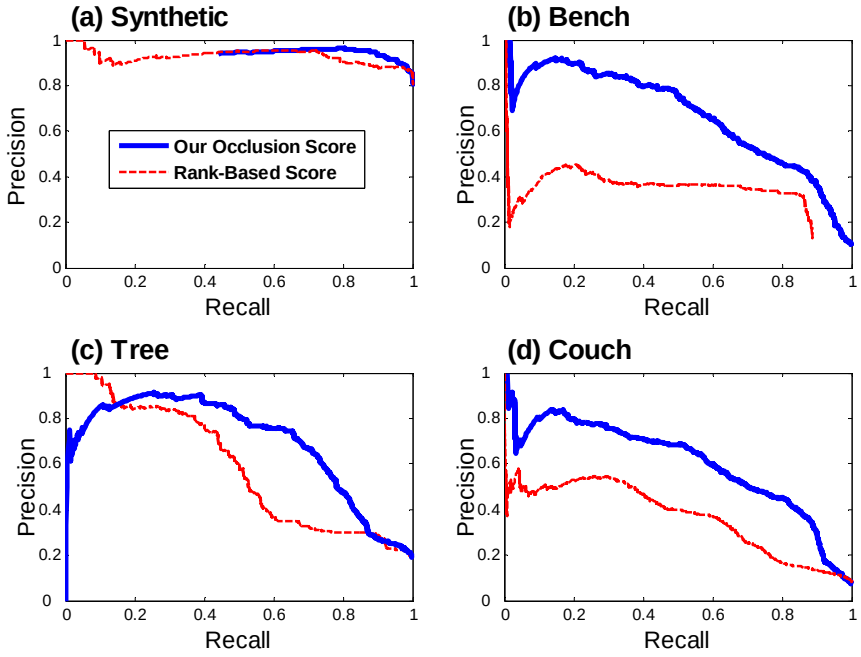


Figure 3.7: Precision vs. Recall plots for each example, showing significant performance gain when using our motion-estimation score over the rank-based motion inconsistency measure from [156].

3.2.2 Appearance Cues

As presented above, edge detectors generally assign a "strength" to each pixel, which captures the degree to which an edge exists there, based on the contribution of various perceptual cues such as local contrast in brightness, color, or texture. These are exactly the sort of appearance cues we would like to incorporate into our classifier.

There has been significant progress recently in combining these sorts of cues, resulting in improved edge detection. As described in the previous chapter, this progress has been achieved in part by moving beyond classical model-based views of edge detection, which rely on linear filtering [30, 132], and instead using non-parametric comparisons of statistical distributions of cues within local patch of data [90, 114, 116, 148, 186]. Furthermore, *multiple* cues can be employed by applying this type of patch-based detector to each cue and then using data-driven, machine learning methods to combine the various detector responses into a single edge strength [90, 114].

Thus, we have chosen to use the popular Berkeley "*Pb*" detector for our experiments [114], which offers a publicly available implementation and already incorporates three appearance cues: brightness contrast, L*a*b* color contrast, and texture contrast (using comparisons of texton distributions [81, 199]). As an added benefit, the *Pb* detector's

default parameters were learned on a large set of human-segmented data [113], allowing us to avoid tedious parameter tuning. The resulting edge strength, e, captures a variety of appearance cues in a single measurement and will serve as the second cue for our classifier below.

3.2.3 Learned Classifier

Our goal is to label edges suggested by the patch-based Pb detector as occlusion boundaries or not. We do so by using the posterior probability of the existence of an occlusion boundary given our features, $\Pr(B|f)$, where f may represent the motion difference d, the edge strength e, or both $\{d, e\}$. Given the substantial, scene-dependent variation in the fraction of appearance edges that are also occlusion boundaries, we assume a uniform prior on $\Pr(B)$ and use Bayes' Rule to estimate this posterior (note that we found estimating a prior from the training data to be unhelpful):

$$\Pr(B|f) = \frac{p(f|B)}{p(f|B) + p(f|\neg B)}. \qquad (3.3)$$

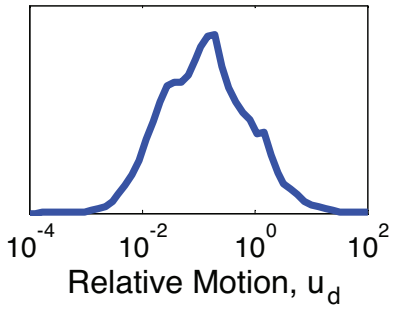


Figure 3.8: The empirical distribution of relative motions $\mathbf{u}_d$ at occlusion boundaries reveals the subtlety of the relative motion cue we employ in our classifier.

Given training data, we can sample our edge strength and motion difference features to estimate the necessary data likelihoods, $p(f|B)$ and $p(f|\neg B)$, as described in the next section. Thresholding this ratio yields the classifier used for our experiments. In the future, it may be possible to achieve better performance by learning adaptive priors for a given image sequence.

3.3 Experiments

For our experiments, we first extract our edge strength feature by applying the Berkeley Pb code to the reference frame of each sequence, using all default parameters (*i.e.* those learned from the BSDS training data [113]). Then, as described in Section 3.1, we align small patches ($r = 12$ pixels) on either side of each edge according to the edges' detected orientations (see Figure 2.1). Using (2.8), we estimate the translational motion of each patch separately and compute the Euclidean distance between the two estimates. We use a temporal window radius of $N = 3$ frames and spatial and temporal weighting function bandwidths of $\sigma_h = N$ and $\sigma_w = r$, respectively. As shown by the distribution in Figure 3.8, most relative motions $\mathbf{u}_d$ at occlusion boundaries are quite small, with a mean of only 0.14

pixels/frame. This supports our claim that the motion cue available for our task is quite subtle.

3.3.1 Training

We randomly select half of our dataset to use for training. We first determine the correct label for all detected edge pixels in an image by matching them to occlusion boundary pixels from the ground truth data. Because of localization inaccuracies (in labeling and detection), we use an approach similar in spirit to the one outlined in Appendix B of [114], which seeks to find a one-to-one correspondence between detected edge pixels and nearby hand-labeled boundary pixels. A given training set consists of 15 scenes, yielding a total of approximately $80,000$ individual examples of edge pixels for training. Unfortunately, these examples are taken from contiguous edges and therefore the patches used in generating their appearance and motion cues overlap significantly. Thus they are highly dependent samples, making it inappropriate to use them all for training.

To alleviate this problem somewhat, we consider only a random subset of the edges available in the training set. This subset is selected such that no two samples which come from the same image could have utilized overlapping patches of data in estimating motion or computing Pb. Thus, for these experiments, we sample edges that are at least $r = 12$ pixels apart. This is depicted in Figure 3.9. The resulting subset contains approximately 6000 examples, which we use for the training described below. (For testing in Section 3.3.2, we classify *all* edges detected in a given image.)

Using the edge strength and motion features for all (sub-sampled) edge pixels corresponding to ground truth occlusion boundaries, we construct kernel density estimates of each cue likelihood independently, $p(e|B)$ and $p(d|B)$, as well as their joint likelihood, $p(e, d|B)$. Similarly, we use any detected edges that are *not* occlusion boundaries as negative examples to learn $p(e|\neg B)$, $p(d|\neg B)$, and $p(e, d|\neg B)$. We use a Gaussian kernel with $\sigma = 1$

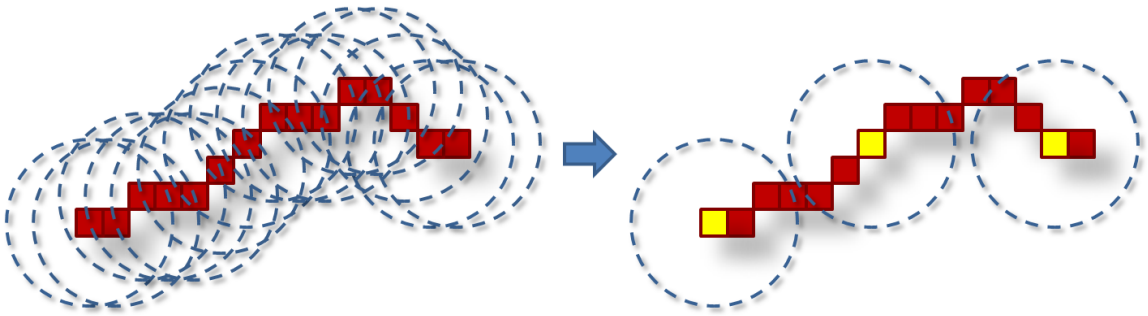


Figure 3.9: Since the patches of data for cue extraction overlap for adjacent edge pixels along a contour (left), we sub-sample when training our classifier (right).

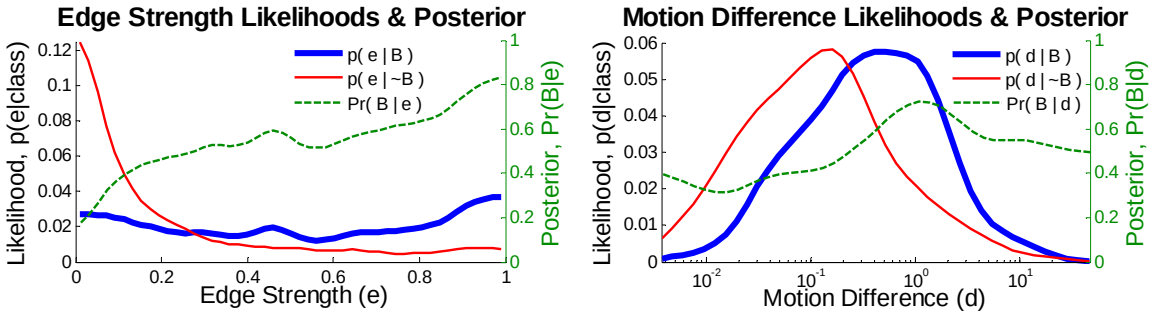


Figure 3.10: Independent distributions and ratio scores for our two cues.

bin, and $\pm 3\sigma$ support. For each cue, we use 50 bins (and thus the joint likelihood estimate contains 50×50 bins). In our experience, using a kernel does offer improved results, despite the fairly coarse binning, particularly in terms of generalization from training to test data. To emphasize the importance of distinguishing the very small motion differences (see Figure 3.8), the bins used for estimating the likelihood of the motion-difference cue are logarithmically spaced between 10^{-3} and 10^{2} (where very large motion is indicative of noise or lack of texture). The bins for edge strength are linearly spaced between 0 and 1.

The resulting independent cue likelihoods are shown in Figure 3.10. As evidenced by the separation of the distributions for each class, these cues do contain some distinct information for our classification task. The distributions also make intuitive sense: higher edge strength and larger motion differences more commonly correspond to occlusion boundaries. It is worth noting that the motion difference cue is fairly weak (*i.e.* the distributions overlap significantly). While improved motion estimation techniques may help, this suggests that the use of optical flow alone for finding occlusion boundaries, as is common practice in segmentation schemes based on motion, could produce poor results on natural scenes which lack texture at many true occlusion boundaries.

The estimated joint likelihoods are shown in Figure 3.11. We have estimated the full two-dimensional joint distributions $p(e, d|B)$ and $p(e, d|\neg B)$ as well as approximate joint distributions $p(e|B)p(d|B)$ and $p(e|\neg B)p(d|\neg B)$, which assume our two cues are conditionally independent. Given the visually similar estimates, it would appear safe to make such an independence assumption and approximate the joint in this manner. We will test our classifier with both versions below.

Next we compute the posterior probability according to (3.3). For the separate cues, the result is overlaid on the likelihoods in Figure 3.10. For the combined cues, the posterior estimates are found in the rightmost pair of Figure 3.11. Rather than fitting an arbitrary model to the posterior, we have chosen to use the estimates as non-parametric lookup tables.

Finally, we evaluate the learned classifier on the training data itself. After estimating

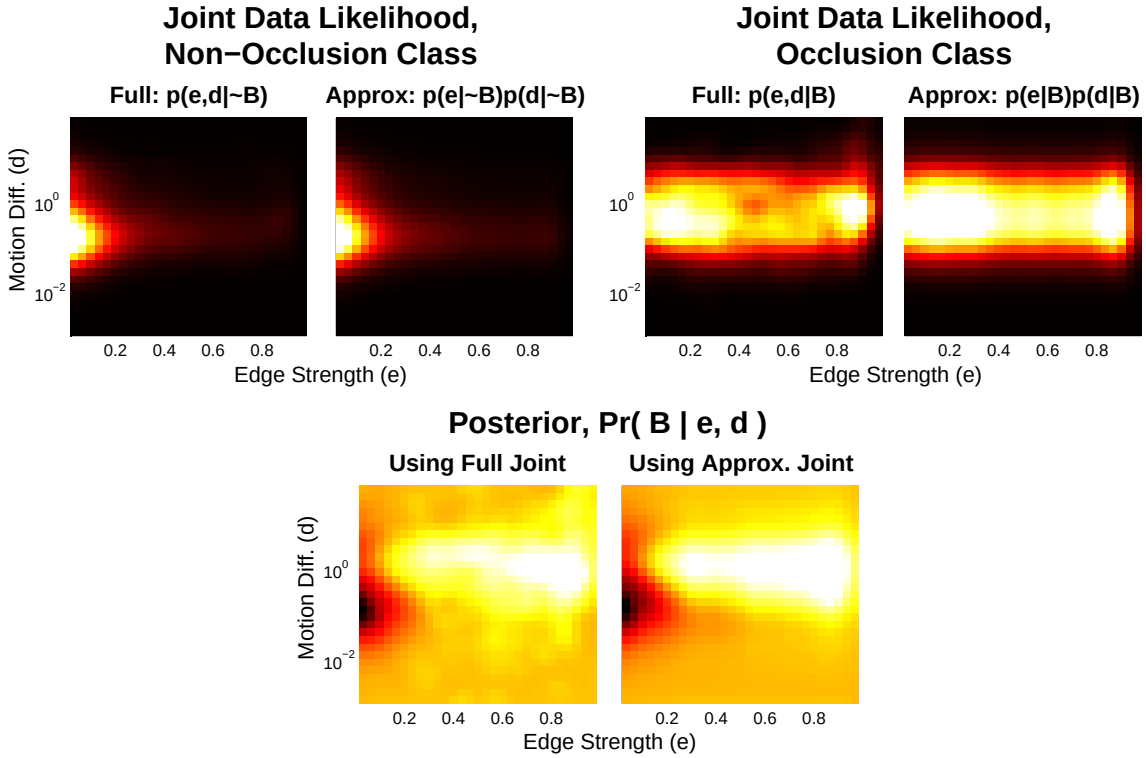


Figure 3.11: Learned joint likelihood distributions and ratio scores for our two cues. Left and right of each pair shows the result using the full and approximated joint, respectively.

$\Pr(B|f)$ at each edge pixel, we generate precision-versus-recall curves by varying the threshold on that posterior estimate and recording the number of correctly labeled pixels. As seen in the left plot of Figure 3.12, each cue separately provides some information, but the two together perform better, with the full joint providing the best result. The precision levels of these curves also capture a notion of the difficulty of our task and dataset.

We can repeat the entire training process with a different randomly-selected set of sequences for training. Doing so allows us to compute the error bars on the precision recall curves shown in Figure 3.12. These error bars represent plus/minus one standard deviation ($\hat{\sigma}$) for $n = 50$ trials. Thus they indicate the typical distribution of the curves for various divisions of the data. The confidence intervals based on standard errors ($\hat{\sigma}/\sqrt{n}$) are very tight and visually imperceptible from the mean (and thus are not shown). This indicates a statistically significant difference between the mean curves in the plots.

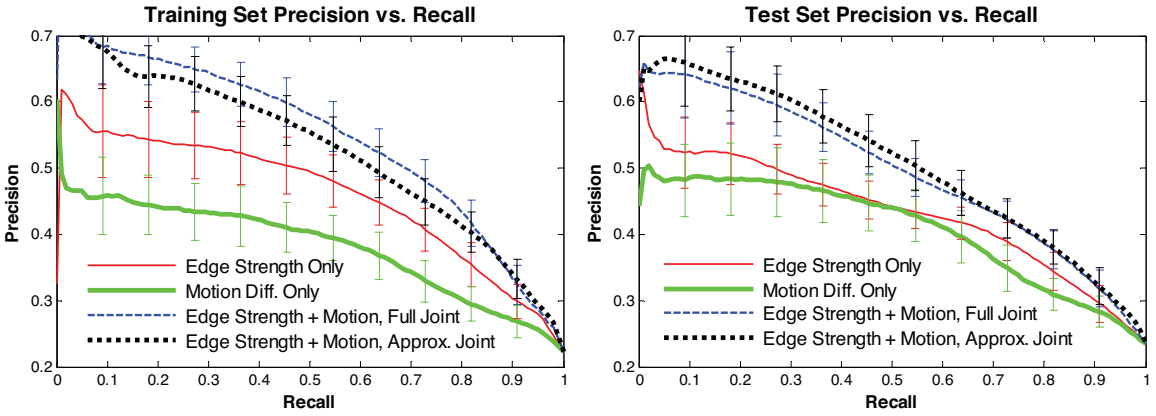


Figure 3.12: Precision vs. recall curves for the training and test sets (left and right, respectively), using various combinations of cues. Error bars indicate plus/minus one standard deviation of the curves for 50 randomly selected divisions of the dataset.

3.3.2 Testing

For testing, we use the remainder of the dataset, extracting motion and edge strength cues as before. This includes the other half of the scenes, with approximately $80,000$ edge pixels to be classified. Again, we classify each edge by thresholding the estimated posterior. We can vary this threshold to produce the precision-recall curves shown in the right plot of Figure 3.12. Here we see confirmation that the learned classifier can generalize to novel scenes. We see similar performance between the full and approximated joint distributions, with marginal improvement using the approximation. This may indicate that the full joint estimate is slightly overfitting the training data. Once again, by repeating the experiment with different test sets, we can also generate error bars.

Aggregated quantitative results as provided in Figure 3.12 give a general sense of performance, but here we also provide a few anecdotal examples from the dataset to exhibit more concretely the information sometimes hidden in such cumulative comparisons. Figure 3.13 shows a scene with ground truth overlaid. To illustrate the improvement when using both cues together, we have selected the threshold for each classifier that results in 60% recall, as indicated on the precision-recall plot. For the indicated window of the original scene, the right four boxes compare the ground truth labeling and the classification results using the cues individually and together. As shown, the best result (with significantly higher precision) is achieved using both cues. For example, combined cues yield improved detection with fewer false positives on the left side of the leg as compared to the result using individual cues alone. Similarly, the examples in Figure 3.14 demonstrate classification improvement using combined cues.

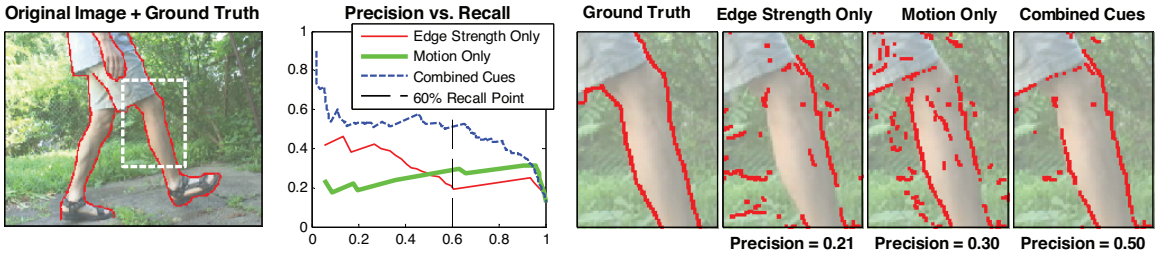


Figure 3.13: Example classification result at a chosen operating point of 60% recall. Combining appearance and motion cues produces superior precision than either cue alone. Note in the combined result the increased detection on the left of the leg as compared to using edge strength alone, and the decreased spurious detections as compared to using motion alone.

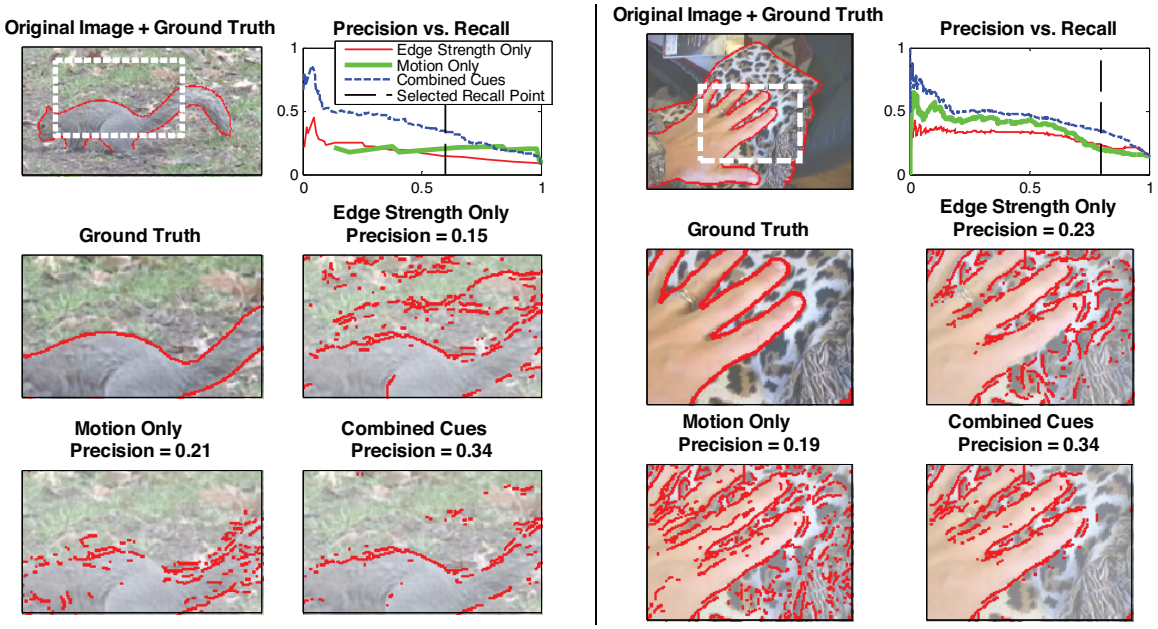


Figure 3.14: Two additional example classification results. Combining appearance and motion cues produces superior precision at the selected recall operating point than using either cue alone. Note the decreased false positives when using the combined cues.

3.4 Motion-Based Edge Detection

Because the performance of any local edge detector is limited, some edges will always be missed. By restricting ourselves to the classification of only the appearance edges which *are* detected, we therefore inherit those limitations when trying to find the subset of occlusion boundaries. As detectors improve (*e.g.*, in detecting very weak edges), so too will our approach. For our dataset, however, the edge detector fires with non-zero strength on 83.5% of the ground truth boundaries, indicating that our technique is viable in practice. Still,

a complementary approach including the detection motion-based edges *directly* is worth investigation.

Using the recent, state-of-the-art optical flow technique by Brox *et al*. [29], we can first compute dense estimates of motion vectors (u, v) throughout the image. To find edges in these motion fields, we again employ a compass-style detector in order to compare distributions of *motion* at each location, rather than distributions of various appearance cues as in Section 2.2.2. By comparing the degree to which these motion distributions differ at various orientations, via the χ^2-distance, this captures information analogous to our motion difference cue used thus far. An example is provided in Figure 3.15.

Using this "motion edge strength" cue, we can perform the same local occlusion detection experiments as before and examine the resulting precision versus recall. In the top row of Figure 3.16, we see the same example scenes from Figures 3.13 and 3.14. Each shows the classification and precision given motion-based edge strength, thresholded to achieve the same recall operating points from the previous figures. In the bottom row, the overall precision versus recall for occlusion detection is presented using these motion-based edges, as compared to our multi-frame motion-difference cue from before (*i.e*. the same curve from the right graph in Figure 3.12).

While there exist some examples, such as the bench in Figure 3.15, for which computing edge strength directly from the optical flow data provides good results, the results in Figure 3.16 indicate that our motion-difference cue is superior for our task. This is partially due to the poor localization of the motion-based edges (probably owing to the coarse-to-fine strategy employed by [29]), *e.g*. around the right foot in the first example in Figure 3.16. If one is willing to sacrifice localization, however, detections further from the ground truth boundaries can be considered to be correct, *i.e*. true positives for computing precision and recall. In this case, we found that the precision performance using motion-based edge detection improves in the low-recall regime (and surpasses our motion-difference cue), but remains lower in the high-recall regime. In addition, this and many other highly-accurate optical-flow methods are quite computationally expensive, and for our purposes, can waste time computing dense flow estimates away from the boundaries in which we are most interested.

3.5 Discussion

The sort of approach described in the previous section may in some cases allow for the detection of occlusion boundaries visible *only* due to motion, such as those often studied in psychophysics [1, 33, 67], but in our experience these cases are relatively rare in practice and may not justify the associated increase in computational cost for dense motion estimation. Furthermore, as discussed earlier, failure to detect weak *texture-less* edges is a much more

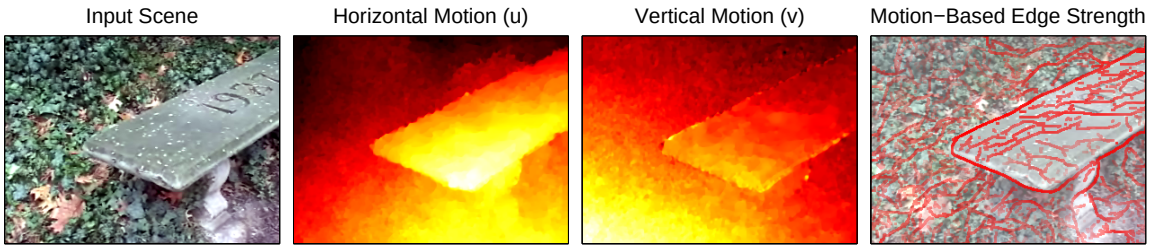


Figure 3.15: Motion-based edge detection using a compass-style edge detector directly on dense optical flow computed according to [29].

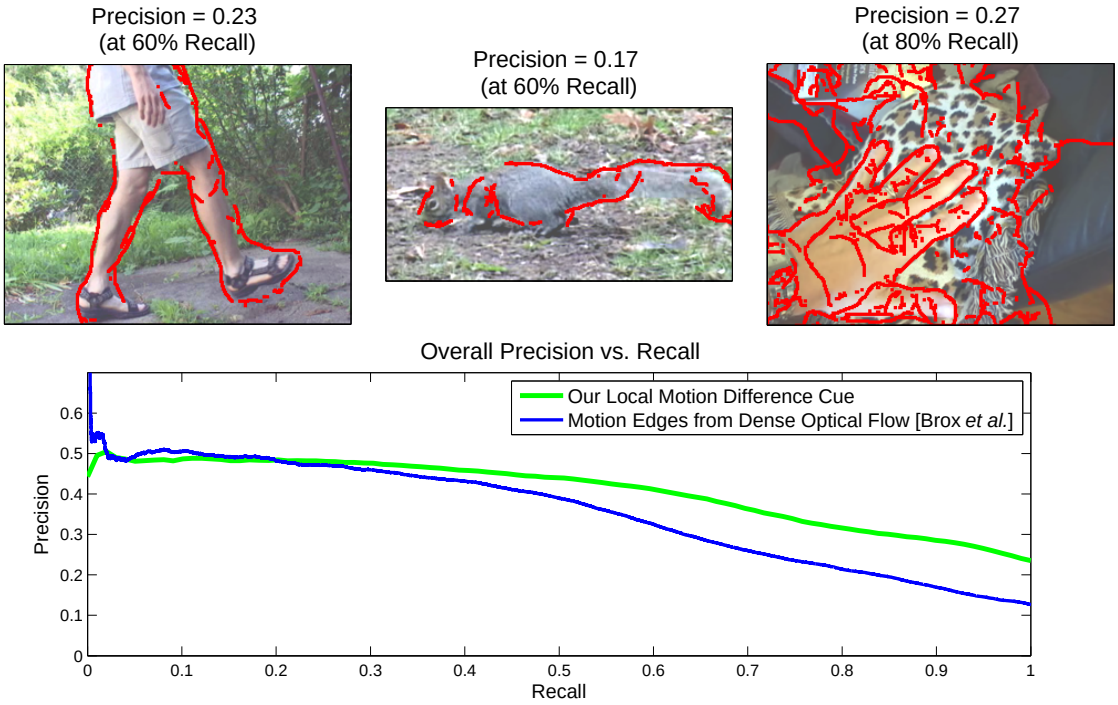


Figure 3.16: Results for occlusion detection using dense optical flow estimates for motion-based edge detection. The detection results in the top three scenes can be compared to the motion-difference results in Figures 3.13 and 3.14, which were thresholded at equivalent recall points. The overall precision versus recall plot also suggests that our motion-difference cue is superior for our task of differentiating occlusion boundaries from other edges.

frequent phenomenon. We note that motion cues at these same edges would almost certainly *not* be informative anyway since they rely on the presence of some texture for accurate estimation. In such cases, we believe better local edge detection (via incorporation of motion information, *etc.*) is likely not the answer. Instead, mid-level reasoning, such as the approach described in Chapter 4, is more important.

Local estimates of any kind, including the Pb detector and our motion difference feature,

are inherently noisy and ambiguous. They are most useful when incorporated into more global reasoning, *e.g.* using a graphical model. Rather than blindly using local estimates for mid- and high-level tasks, however, we believe it is important, if not crucial, to evaluate the utility of these low-level cues themselves (separately and in combination). Having verified here the benefit of using motion in this chapter, we are now ready to describe in the following chapter methods of globally reasoning about object/occlusion boundaries which build on the combined local cues described here and in Chapter 2. In addition, the learned classifier up to this point is still fairly simplistic. It does not incorporate any notion of uncertainty of the motion estimates, nor does it reason about the *boundary* motion as discussed in Chapter 2. These and other improvements will be covered in the next chapter.

Chapter 4

Global Boundary Reasoning

In the preceding chapters, we described methods for detecting occlusion boundaries based on local motion and appearance cues, and we demonstrated that the consideration of subtle relative motion cues provides improved performance. Such low-level, local processing techniques, however, cannot be expected to solve the problem entirely. Several challenges remain:

- **Spatial Support.** Thus far we have considered identifying and classifying single-pixel edges based on rectangular patches of a set size. Since boundaries are themselves extended contours in the image, we would instead prefer to consider edge elements with more informative extent than a single pixel. In addition, the regions of support for the patches that we extract on either side of a potential boundary ought to be data-driven rather than set to an (arbitrary) spatial shape and/or size.

- **Boundary Motion.** Though we introduced the motion of the boundaries themselves when reasoning about occlusions in Chapter 2, we subsequently ignored this cue, lacking a natural strategy for integrating it into the classifiers and scores described in Chapter 3.

- **Motion Uncertainty.** We motivated the consideration of motion uncertainty in Chapter 3 and incorporated it to some extent in a hand-designed "inconsistency" score function (at least for *patch* motion estimates). But it was not clear how to employ effectively this type of confidence information in our initial learned classifier, which was based on a simple likelihood ratio. Intuitively, this confidence information should be taken into account in reasoning about occlusion boundaries since motion estimates can be unreliable (and therefore misleading for classification) in many situations.

- **More Cues.** Related to the previous two points, it is not straightforward to introduce potentially large numbers of additional cues, including uncertainty and boundary

motion, into our simple likelihood-ratio classifier from Chapter 3. In particular, it becomes infeasible to populate and employ non-parametric lookup tables in higher dimensions. Nor do the methods used thus far address the problem of feature *selection*.

- **Propagation of Local Information.** Locally in an image, occlusion may in fact not be visible at all. We need a higher-level method of propagating occlusion information more globally between local, potential boundary elements.

In this chapter we will present a more global processing approach, relying on mid-level reasoning, which combats the problems above by building on the analyses in previous chapters. First we will describe the extraction of improved boundary elements and associated regions of support on either side. Then we will describe a more powerful inference model which utilizes learned classifiers capable of considering a larger set of cues, including uncertainty and boundary motion. Using this model, we will offer a more powerful approach to tackling the mid-level problem of boundary identification.

4.1 Initial Segments and Fragments

We will begin by over-segmenting the image in order to generate candidate boundary elements, or *fragments*, and associated patches for cue extraction, or *segments* (also known as "super-pixels" [141]). In particular, the boundaries of these segments will serve as our set of hypothesized boundary fragments, and the segments on either side of each fragment will provide data-driven regions of support for our motion and appearance cues.

Another option could be to start from (pixel-level) edge detections and then perform an edge chaining procedure, as in [104, 162] for example. But edge chaining is inherently brittle in natural cluttered scenes, and perhaps more importantly, the over-segmentation approach offers two distinct advantages for our model. First, by construction, fragments meet at intersections of regions, and thus closed contours are immediately available. This produces a natural graph structure suitable for global inference without the need to impose artificially such structure later (*e.g.* with Constrained Delaunay Triangulation [139]). Second, a direct link is established between fragments and segments. It is clear that a set of segments in an image imply a set of boundaries, but working in the opposite direction to obtain a segmentation from a set of disconnected boundary fragments is non-trivial.

We use a watershed-based over-segmentation driven by the output of a learned, statistical, multi-cue edge detector, after non-local maxima suppression. We have chosen the Pb detector [114], though others exist, *e.g.* [90]. The use of such a detector allows the over-segmentation simultaneously to consider color, brightness, and texture in choosing where to create initial segments, as described in Section 3.2.2. We chose the watershed approach for its more regularly-shaped segments as compared to other methods [35, 45],

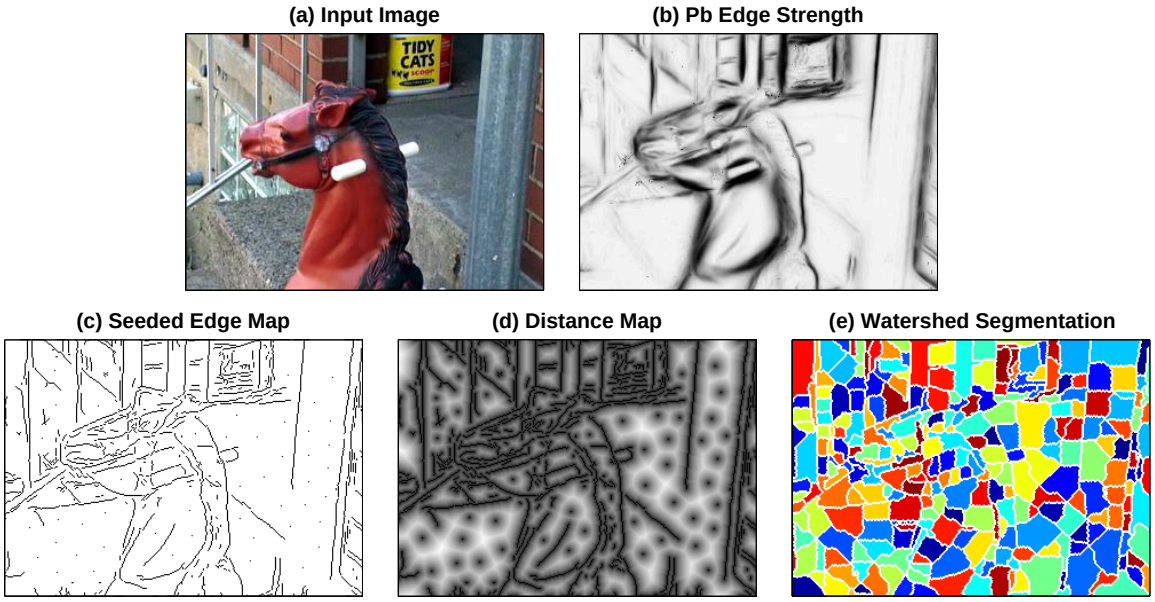


Figure 4.1: For an input image (a), we use the Pb edge detector to get edge strength at each pixel (b). After suppressing non-local maxima, we introduce random "seeds" to help break apart large uniform regions and increase segment regularity (c). Then, from a distance transform (d) of the resulting edge map, we use a standard watershed algorithm to produce our over-segmentation (e).

and its simplicity and speed compared to methods relying on normalized cuts [122, 123, 141]. We first use non-local maxima suppression on the raw Pb responses. Next we introduce random "seeds" into large empty regions in the resulting edge map. This helps break apart potentially large segments and also helps increase regularity. The watershed segmentation algorithm is then applied to the distance transform computed from this edge map. See Figure 4.1 for an example of this process. A similar technique, which uses the Pb detector's output directly in a watershed segmentation, is suggested in [7], but we found that our approach tends to produce segments with more regular shape. This allows for better cue extraction later, particularly in the case of motion.

From the over-segmentation, we produce a set of fragments along the boundaries of each segment, starting and stopping at junctions with other segments. Rather than operating at the level of pixels when chaining, however, we operate instead on the "cracks" between the pixels. These cracks naturally form a graph and offer a very simple, efficient domain on which to chain. For details on our crack-chaining procedure, see Appendix A. Besides efficiency, the fact that cracks have no width in the image offers important advantages. There is no need to decide which segment's pixels are boundary pixels (and thereby effectively assign boundary ownership prematurely). Explicit handling of "mixed pixels", where pixels at a boundary have fractional membership to the surfaces on either side of

that boundary, could be also be used. Using cracks, however, is far simpler and very efficient. Furthermore, their lack of width remains consistent even as an image is resized, *e.g.* for scale-space processing, as in the feature detection process for object recognition discussed in Chapter 6, where cracks are also used to represent boundary information. In addition, a maximum of four fragments can meet at any junction, limiting the number of junction labeling cases we must consider when performing the global inference described in Section 4.2 below.

4.2 A Global Boundary Model

Given segments and fragments from the over-segmentation, the next step is to compute a set of cues associated with each fragment. As before, these cues will be used in estimating the likelihood that each fragment is part of an occluding contour. Based on the findings in Chapter 3, we will incorporate two general types of cues: motion-based and appearance-based. The specific form of these cues will be explained in Section 4.3.1 below. A similar model and learning approach to the one described here is presented in [76], where boundary detection is formulated without using motion by instead employing geometric context cues [74].

For now, we assume that we have extracted a set of appearance and motion cues for each fragment, using the information found along the fragment and within the segments it separates. Our goal, then, is to classify which fragments are occlusion boundaries. While we hope our local appearance and motion cues will provide strong evidence for this classification, it is unreasonable to hope that a purely local solution will suffice. To facilitate global reasoning and propagation of local estimates, we define here a probabilistic graphical model to capture the structure of our problem.

Our objective is to find an interpretation of the scene's boundaries that is most probable given the observed appearance and motion cues. The toy example in Figure 4.2 depicts the type of labeling we will use. Each fragment can take three possible labels. First, the fragment can be labeled "off", indicated by a dashed line, meaning it is believed *not* to lie on an occlusion boundary, *i.e.* it is a surface marking, lighting edge, *etc.* Otherwise it can be labeled "on" (as an occlusion boundary) with the "foreground" — or closer surface — on a specified side indicated by $\texttt{on}_L$ or $\texttt{on}_R$ when appropriate. (We will let the label $\texttt{on}$ represent either, with the appropriate foreground side implied by context.) In the figure, this is indicated by a solid line with an arrow, where the left side of the arrow is the closer side. Where three or four fragments meet, we have a junction, indicated by a solid dot.

Note that fragments along the borders of the image itself are also included in our graph to allow for consistent reasoning about occlusion and closure throughout the image, *i.e.* without requiring special cases at the image borders. There is a special case, however, when

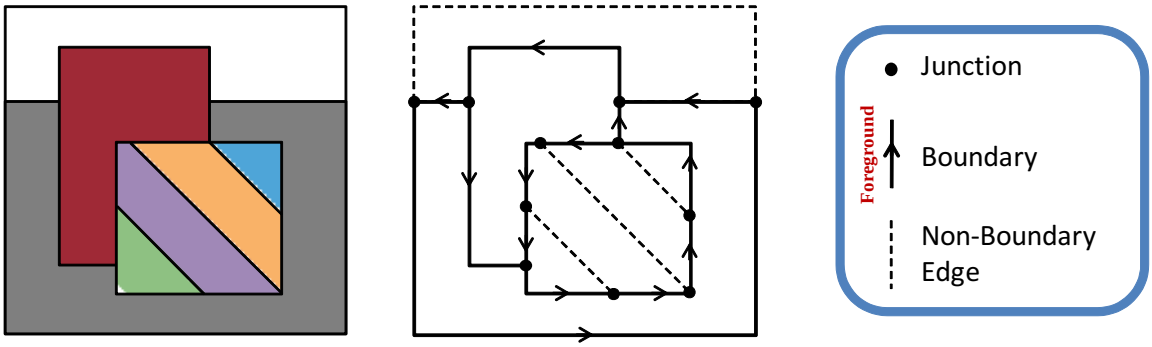


Figure 4.2: A toy labeling example showing the fragments of a simple scene labeled with one of three possible labels (non-occlusion or occlusion with specified foreground side). Note that the borders of the image itself are also considered fragments to be labeled.

a set of segments is fully enclosed within another segment. In this case, two disconnected graphs are formed. Though information will not flow between the two (in this work we have not explicitly avoided this situation), inference as described below can be performed on the two graphs simultaneously. In the extreme case, a single segment could be enclosed within another, and its enclosing fragment would thus be considered and classified completely independently (and therefore entirely based on local information).

To find the most likely labeling of a scene, we need to define the probability of a particular choice of labels for the fragments f given the cues x. Our goal is to maximize this probability, $\Pr(f|x)$, by selecting the best set of *consistent* labels. In our model, we will consider fragments labeled as boundaries and fragments labeled as non-boundaries to be independent. Furthermore, we will assume (1) that each complete boundary, B, is independent of the other boundaries and (2) that each non-boundary fragment is independent of the other fragments. Thus we can approximate the desired probability as

$$\Pr(f|x) \approx \prod_{B} \Pr(\{f \in B\} = \texttt{on}|x) \prod_{j} \Pr(f_j = \texttt{off}|x). \tag{4.1}$$

For a particular *closed* boundary, which is made up of a set of directed fragments (where the direction indicates the foreground side of the boundary), we assume that each fragment along the boundary is conditionally independent of the other fragments in that boundary, given the preceding fragment. Thus,

$$\Pr(\{f \in B\} = \texttt{on}|x) \approx \prod_{\{i \to j\} \in B} \Pr(f_j = \texttt{on}|f_i = \texttt{on}, x), \tag{4.2}$$

where $\{i \to j\} \in B$ represents the set of pairs of fragments in boundary B such that f_i precedes f_j when traversing that boundary according to its directed labeling. Of course, the "on" labels must also specify the same foreground side in order to be consistent.

If B corresponds to an *open* boundary, *i.e.* the boundary of an object which is occluded by another object in the scene (or by the border of the image), the starting fragment of that boundary does not have a preceding fragment on which to condition its probability. So we must consider it separately:

$$\Pr(\{f \in B\} = \texttt{on}|x) \approx \Pr(f_{B0} = \texttt{on}|x) \prod_{\{i \to j\} \in B} \Pr(f_j = \texttt{on}|f_i = \texttt{on}, x), \tag{4.3}$$

where f_{B0} represents the initial fragment of boundary B. For closed fragments, this term is ignored, as in (4.2), since the set of fragments forms a loop where each fragment *can* be conditioned on its predecessor.

Now we can compute the probability of a given labeling of all fragments f using (4.1), (4.2), and (4.3). Direct inference using this model, which would require explicit identification and chaining together of open and closed boundaries as well as a varying graph structure depending on the current labeling, would be quite cumbersome. We can instead simplify matters by focusing on the labeling of fragments and *junctions of those fragments*. Thus we can write the desired probability as a product of fragment potentials and junction potentials,

$$\Pr(f|x) \propto \prod_i \psi(f_i) \prod_k \phi_k, \tag{4.4}$$

where $\psi(f_i)$ represents the potential function for an individual fragment f_i and ϕ_k represents the potential function for the set of fragments meeting at junction k, $\{f_j\}_{j=1}^{N_k}$, where $N_k \in \{3, 4\}$. In practice, it is more convenient (and numerically feasible) to take the negative *log* of this probability and work in terms of minimizing an energy function,

$$E(f) = -\log\left(\Pr(f|x)\right) = \sum_i \log\left(\psi(f_i)\right) + \sum_k \log\left(\phi_k\right). \tag{4.5}$$

Defined carefully, these potentials can capture exactly the same model as in (4.1), (4.2), and (4.3). First, we define the unary fragment potential, based on its fragment's label, as

$$\psi(f_i) = \begin{cases} 1 & f_i = \texttt{on} \\ \Pr(f_i = \texttt{off}|x) & f_i = \texttt{off} \end{cases} \tag{4.6}$$

The junction potential ϕ is defined and evaluated for all junctions depending on the labelings of their constituent fragments. A junction of three fragments, each with three labels, would imply a set of 27 possible label combinations for that junction. However, only 11 of those are unique under circular permutations. Of those 11, we consider five to be physically valid and consistent with our mdel. These are shown in Figure 4.3, with the shaded regions indicating foreground (*i.e.* the left side of the fragment, when moving in the direction of the arrow), and the darkest region being the closest one. These junctions consist of a single continuous boundary occluding another boundary (i)–(ii), the meeting

of three non-boundary edges (iii), or a single continuous boundary passing through the junction with an adjacent non-boundary edge (iv)–(v), .

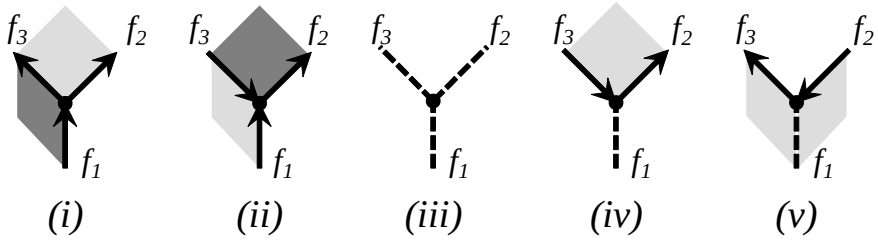


Figure 4.3: The five types of *valid* three-fragment junctions. The shaded regions are the foreground regions, with darker being closer. Similar types exist for four-fragment junctions.

The other six types of three-fragment junctions are shown in Figure 4.4. These junctions are considered "invalid" because the foreground/background labels of their constituent fragments are inconsistent or physically impossible, as in (vi)–(ix), or because we have chosen to discourage the abrupt starting or stopping of a boundary, as in (x)–(xi), in favor of closed boundaries.

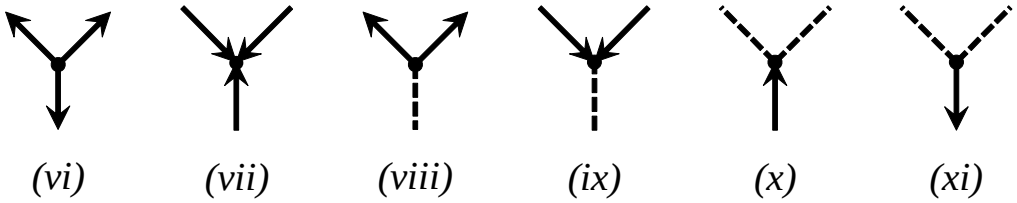


Figure 4.4: The six types of *invalid* three-fragment junctions. Similar types exist for four-fragment junctions.

Analogous reasoning is used for the four-fragment junctions as well, though we will not explicitly list all possible labelings here for the sake of brevity.

Now we can define the potentials for each type of junction implied by a particular labeling. First, we strongly discourage the invalid labelings shown in Figure 4.4 by setting the potentials of such junctions to a very small value (10^{-4} in our experiments[1]). The five valid junctions in Figure 4.3 have potentials shown in Table 4.1. The combination of these junction potentials with the fragment potential defined in (4.6) results in the same set of terms in the original model from (4.1), (4.2), and (4.3); *i.e.* we consider "off" fragments independently and effectively walk along the "on" fragments, which make up boundaries, *without double-counting any fragments*. In particular, the definition of certain terms in

[1] We avoid using potentials with values of exactly zero since we work in practice with *log* probabilities, as indicated in (4.5).

Type	Potential (ϕ)
(i)	$\Pr(f_3 = \texttt{on} \mid f_1 = \texttt{on}, x) \Pr(f_2 = \texttt{on} \mid x)$
(ii)	$\Pr(f_2 = \texttt{on} \mid f_3 = \texttt{on}, x) \Pr(f_1 = \texttt{on} \mid x)$
(iii)	1
(iv)	$\Pr(f_2 = \texttt{on} \mid f_3 = \texttt{on}, x)$
(v)	$\Pr(f_3 = \texttt{on} \mid f_2 = \texttt{on}, x)$

Table 4.1: Junction potentials corresponding to junction types in Figure 4.3.

some potentials as having a value of one — such that they have no effect on the products in (4.4) — helps make this possible, at the expense of a somewhat obfuscated model at first glance.

The overall labeling probability $\Pr(f|x)$ can now be computed for any assignment of labels to fragments: for each junction we find the type that is induced by its fragments' labels, and we use the corresponding expression of ϕ to compute the contribution of that junction. The resulting inference problem (finding the assignment of labels f that maximizes $\Pr(f|x)$) is intractable in its exact form on our loopy graph. However, an approximation of the MAP solution can be found by loopy belief propagation [58, 108, 130, 131, 185, 189], where the messages passed are based on the potentials defined above for each fragment or junction. Specifically, we use an implementation of the sum-product message-passing algorithm suggested in [72], and also exponentiate each potential factor in (4.4) by $1/T$ (with $T = 0.5$ in our experiments) to provide "soft-max" estimates, according to the mean field approximation suggested by [197].

We note that this approach bears some similarity to classical approaches to line drawing interpretation [66, 180]: we have essentially reduced the problem to labeling the junctions using a dictionary of five junction types. A key difference here is that we use soft potentials in a machine learning framework for our reasoning, instead of relying on hard constraint propagation as in the more classical setting.

4.3 Fragment Classification

Despite the various complex junction cases described in the previous section, note that all the potentials in our model are defined in terms of just two probabilities: a three-class unary probability that a fragment is "on" (and which side is foreground) or "off", $\Pr(f_i = \{\texttt{on}_L, \texttt{on}_R, \texttt{off}\}|x)$, and a pairwise probability that a fragment is "on" given that the preceding fragment is also "on", $\Pr(f_j = \texttt{on}_s | f_i = \texttt{on}_s, x)$, such that the foreground side, s, is consistent. (Though we have not listed them here, one can also define the four-fragment junctions in terms of these same probabilities by following analogous reasoning.)

We will now describe a classifier used to estimate these probabilities, as functions of features (x) extracted from labeled training data. For simplicity, we will return to omitting the foreground-side indicator for "on" labels in the following discussion.

We employ Adaboost [34] to learn our classifiers, where the weak learners used by the boosting process are decision trees [59, 76, 167], instead of using the simple likelihood ratio as in Section 3.2.3. Passing the results through a logistic function and normalizing such that they sum to one over our classes yields estimates of the desired probabilities for each class.[2] Since boosted decision trees are well-suited to feature selection, we can provide a variety of potentially over-complete, or redundant, appearance and motion features, as described below, and allow the classifier to choose the most discriminating ones based on the training data. Such an approach has also been shown to be successful for reasoning about the 3D structure and layout of single images [75].

4.3.1 Appearance and Motion Cues

All of the probabilities used to define the potentials in our model are conditioned on features or cues, x. In this section, we will describe the corresponding set of appearance and motion cues extracted from the vicinity of the fragments and their associated segments.

For our *appearance* features, which also include some "geometric" information, we provide the following to the unary classifier, $\Pr(f_i = \{\texttt{on}, \texttt{off}\}|x)$:

- Average *Pb*-strength along the fragment.
- Fragment length.
- Ratio of fragment length to the perimeter of the larger of the two neighboring segments. (Intuitively, we may want the classifier to be reluctant to turn "off" a small fragment which would effectively merge one large segment into another.)
- Difference in area between neighboring segments. (This captures some of the same intuition as the previous cue.)
- Euclidean distance between the average colors (in L*a*b* space) of the neighboring segments.
- The χ^2-distance between L*a*b* color distributions provided by kernel density estimates within the neighboring segments. (This is akin to the patch-based edge detection techniques described earlier, such as *Pb* strength, but with color only and regions of support given by the over-segmentation instead of a standard circular patch. Texture/brightness could be added as well.)

[2] Henceforth, we will adopt a slight abuse of terminology and refer somewhat interchangeably to the estimated probabilities and the classifiers from which they are estimated.

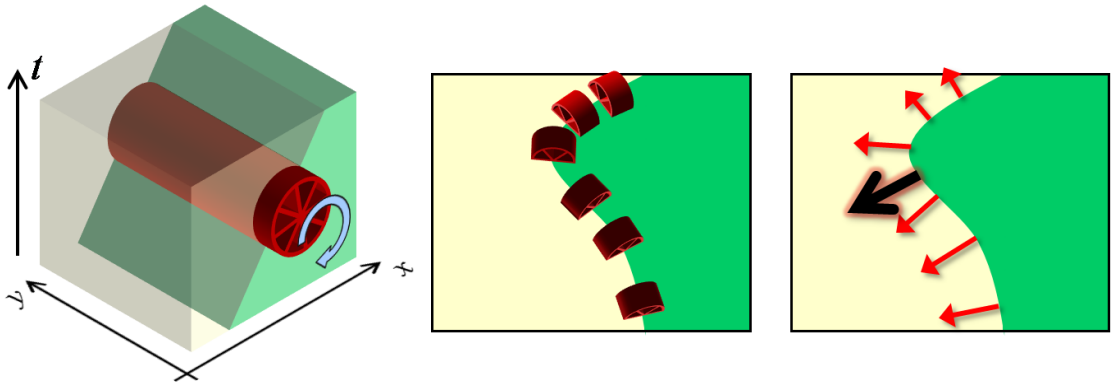


Figure 4.5: We can use a cylindrical, spatio-temporal edge detector (left), as described in Section 2.2, at each point along a boundary fragment (middle), aligned to the fragment's local orientation. Each detector will yield a normal speed estimate (right, red arrows), which can be combined into a single motion vector for the fragment (larger black arrow) using Equation 4.8.

The first type of *motion* cue we employ is analogous to the relative patch motion used in the previous chapter. Now, however, the segments from Section 4.1 naturally specify the spatial support for estimating left- and right-side motions for each boundary fragment (as opposed to the oriented, rectangular patches used before). And since each segment is bounded by multiple fragments, we need only compute the motion for each segment in the scene once, instead of repeatedly computing motions from overlapping neighboring patches. We use the same motion estimation procedure as before (see Section 2.1), now with a spatial weighting function $w(x, y)$ which decreases the contribution of pixels near the boundaries of the segment.

Unlike the initial classifier in Chapter 3, here we will also use *fragment* motion estimates as a second type of motion cue. Thus we will employ the cylindrical, spatio-temporal edge detection approach described in Section 2.2 (the local orientation is already specified by the fragments' normal vectors, so the spherical detector is unnecessary here). Recall that this approach only offers (1D) estimates of normal motions at each location due to the aperture problem. We can, however, combine normal estimates along the fragment to get a full 2D (u, v) motion estimate for the whole fragment [42, 182, 183], as shown in Figure 4.5. We use a linear system of equations based on comparisons of the individual 1D speed estimates (computed at each pixel along the fragment) to the fragment's overall motion vector projected onto the local normal vectors:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \arg\min \sum_{i \in F} w(i) \left(n_{x,i} u + n_{y,i} v - s_i \right)^2 , \tag{4.7}$$

where $n_{x,i}$ and $n_{y,i}$ are the components of the normal at point i on the fragment F, and s_i is the corresponding speed from the spatio-temporal detector. Each point contributes to

the solution with a weight, $w(i)$, corresponding to the underlying edge strength reported by the local detector.

As was described in Section 2.1, in practice we employ the M-estimator in (2.10) instead of simply squaring the error term in (4.7), and again we use Iteratively Re-weighted Least Squares to find a robust solution to this system. At each iteration, we solve the following system, incorporating the current M-estimator weights into w at each step (see Section 2.1 for further details):

$$\underbrace{\begin{bmatrix} \sum wn_x^2 & \sum wn_x n_y \\ \sum wn_y n_x & \sum wn_y^2 \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum wn_x s \\ \sum wn_y s \end{bmatrix}. \tag{4.8}$$

As with the patch or segment motions, we would like to have a notion of confidence associated with these fragment motion estimates. For example, we are more sure of the motion vector computed for a corner fragment, which constrains the fragment's motion in two dimensions, than for a straight fragment, which is unconstrained *along* the fragment and thus effectively continues to suffer from the aperture problem despite combining multiple estimates via (4.8) [160]. Thus the confidence is tied to the variation of the local normals along the fragment (see Figure 4.6). By employing a least squares formulation, our motion estimate for the fragment is the maximum likelihood solution for (u, v) under a Gaussian assumption. The matrix H from (4.8) — which is constructed from the local normal vectors along the fragment — defines the covariance of that Gaussian, and thus captures our confidence in the estimated motion components. See Section 3.1 for further details, where we used G from (2.8) to model patch motion uncertainty in an analogous setting.

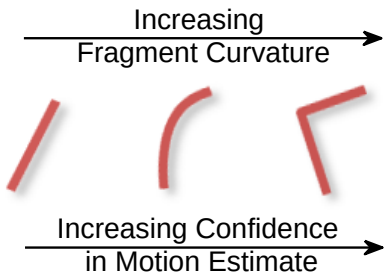


Figure 4.6: Our confidence in the estimate of a fragment's motion increases the more corner-like that fragment is.

Given the fragment and segment motion estimates, we can compute a set of motion features suitable for our classifier. We compare the fragment motion estimate with each of its neighboring segment motion estimates in addition to comparing the segment motions to each other (as with the simple rectangular patches in Chapter 3). For each of these comparisons, we compute the following features to add to the appearance features listed above:

- Absolute difference between individual u and v motion components.
- Simple Euclidean distance between motion vectors.

- Confidences of motion estimates.
- The Mahalanobis-like motion inconsistency score used in Section 3.1.

Once again, these cues are somewhat redundant, deferring the selection of the "right" set of features to the boosted classifier.

For the pairwise classifier, $\Pr(f_j = \texttt{on} | f_i = \texttt{on}, x)$, there are a few additional pair-specific features we can use. Thus the feature set for this term includes the unary cues above for each fragment of the pair, augmented by the following:

- The relative angle between the two fragments (to capture a notion of continuity).
- The difference between the motions of the two fragments.
- The motion and color differences between the two fragments' foreground-side segments.

4.3.2 Training the Classifiers

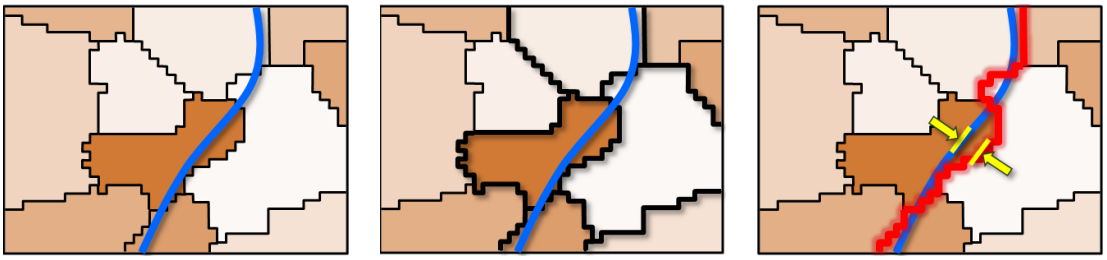
Given a set of training data, we apply the over-segmentation and fragment-chaining approaches described in Section 4.1 and compute each cue defined above. We then train the *unary* classifier, $\Pr(f_i = \{\texttt{on}, \texttt{off}\} | x)$, directly from the individual ground truth labels using the extracted set of cues.

For the *pairwise* classifier, we first extract pairs of fragments in the ground truth for which f_j follows f_i and both are labeled "on." These pairs are our positive examples while negative examples are collected from those pairs for which f_j is "off" but is connected via the graph to an f_i that is "on". From these examples, we learn the second classifier, $\Pr(f_j = \{\texttt{on}, \texttt{off}\} | f_i = \texttt{on}, x)$. For all our experiments, we allow ten iterations of boosting with ten-node decision trees, each of which is fit using an implementation of [26] available in Matlab.[3]

For training each of these classifiers, we must match the fragments created by our over-segmentation with ground truth boundaries drawn by hand. This is accomplished by first determining through which segments the hand-drawn boundaries pass. Then, the fragments corresponding to those segments are used to get the closest approximation to the ground truth boundary. This process is depicted in Figure 4.7.

In Figures 4.8 and 4.9, we plot histograms indicating the frequency with which each of our features was used by the boosted decision tree classifiers learned in the experiments below. As shown, each classifier frequently uses motion *and* appearance features. In fact, the two most-used features employed by the unary classifier (Figure 4.8) are the Euclidean distance between segment motion vectors and the difference in color distributions on either

[3] See treefit, treefitw, treeval, and the related suite of commands.



(a) (b) (c)

Figure 4.7: For the over-segmentation and overlaid human-drawn boundary in (a), we find the segments through which that boundary passes (b). In (c), the fragments associated with those segments (shown in red) are then used to approximate the human-specified boundary (shown in blue). The error between the two, indicated by the yellow arrows, has a mean of 2.6 pixels and a median of 0.8 pixels across our dataset. Substantial errors (deviations greater than ten pixels, *e.g.* where the over-segmentation completely misses a true boundary) occur in only 4.9% of the boundaries, measured pixel-wise.

side of the fragment. Furthermore, the classifier regularly uses different forms of the confidence information derived from computing segment and fragment motion estimates.

Perhaps somewhat surprisingly, the "relative angle feature" used for capturing the continuity between a pair of fragments is not often utilized by the pairwise classifier, despite being a common Gestalt cue for reasoning about saliency and continuation in perceptual organization, *e.g.* [99, 109, 139] and references therein. Its limited utility here may be tied to the fact that the graph implied by our over-segmentation procedure, which seeks regularly-shaped segments, tends to have plausible relative angles for almost *any* pair of fragments meeting at a given junction. Thus, the relative angle may not be a very *discriminating* feature in our construction.

Finally, the addition in this chapter of features based on *fragment* motion, rather than patch or segment motion only, does appear to be helpful. Fragment motion cues are used regularly, especially those estimated for fragment j in the pairwise classifier.

4.4 Experiments

We first over-segment each scene's reference frame and extract fragments as described. There are approximately 20,000 total fragments available in our dataset for testing and training. For each fragment and segment, we compute the set of appearance and motion cues listed in the previous section. From a training set constructed using half the scenes in the database, we then learn the two classifiers which define the potentials used by our model. Using these classifiers for the other half of the scenes, we can estimate the probability of labeling each fragment as an occlusion boundary, first by using the learned unary classifier

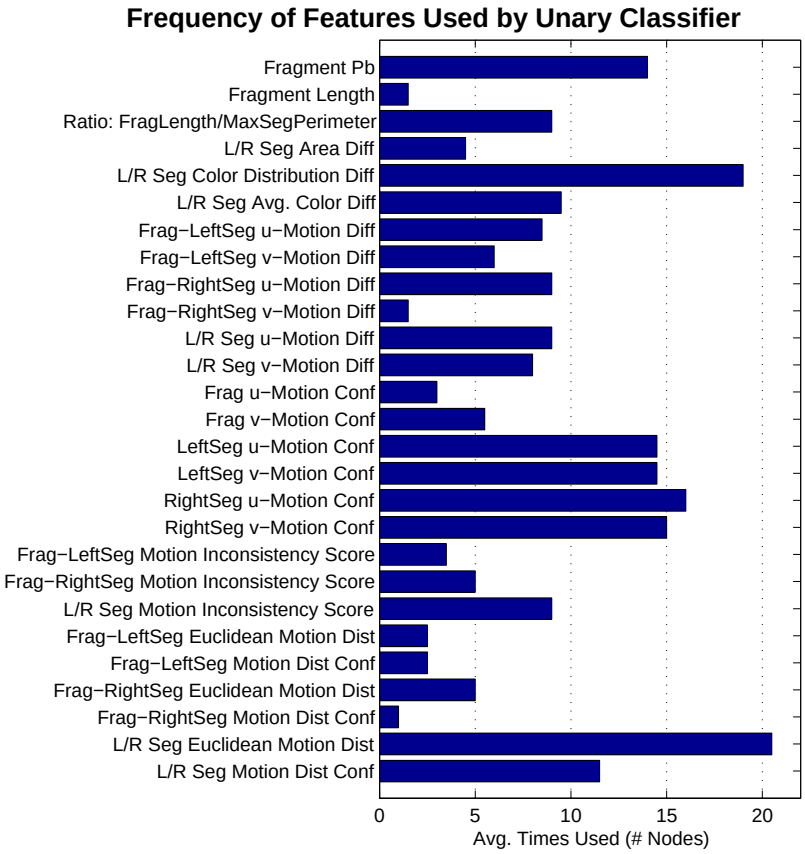


Figure 4.8: Frequency with which each computed feature is used by the learned unary fragment classifier.

in isolation and then within the global inference model. We repeat this procedure, swapping the test and training sets in order to obtain results for all scenes in the database.

We would like to verify that both global inference and motion information result in improved performance overall. We see in Figure 4.10 that this is indeed the case in a plot of precision versus recall for final fragment labelings of the entire dataset in aggregate. The parameter varied in creating each plot is the threshold on the likelihood of each fragment being "on". We see that using appearance cues alone results in the worst performance on the graph.[4] In fact, including motion cues but only reasoning on individual fragments, *without* global inference, offers equivalent or superior results to using global inference with appearance cues alone. Finally, global inference with combined motion and appearance information consistently yields the highest precision. Also note that the low precision point at 100% recall (corresponding to the trivial solution of labeling *all* fragments as occlusion

[4]We do not show results using motion cues alone, but as with the initial classifier from the previous chapter, this yields similar or worse performance than using appearance alone.

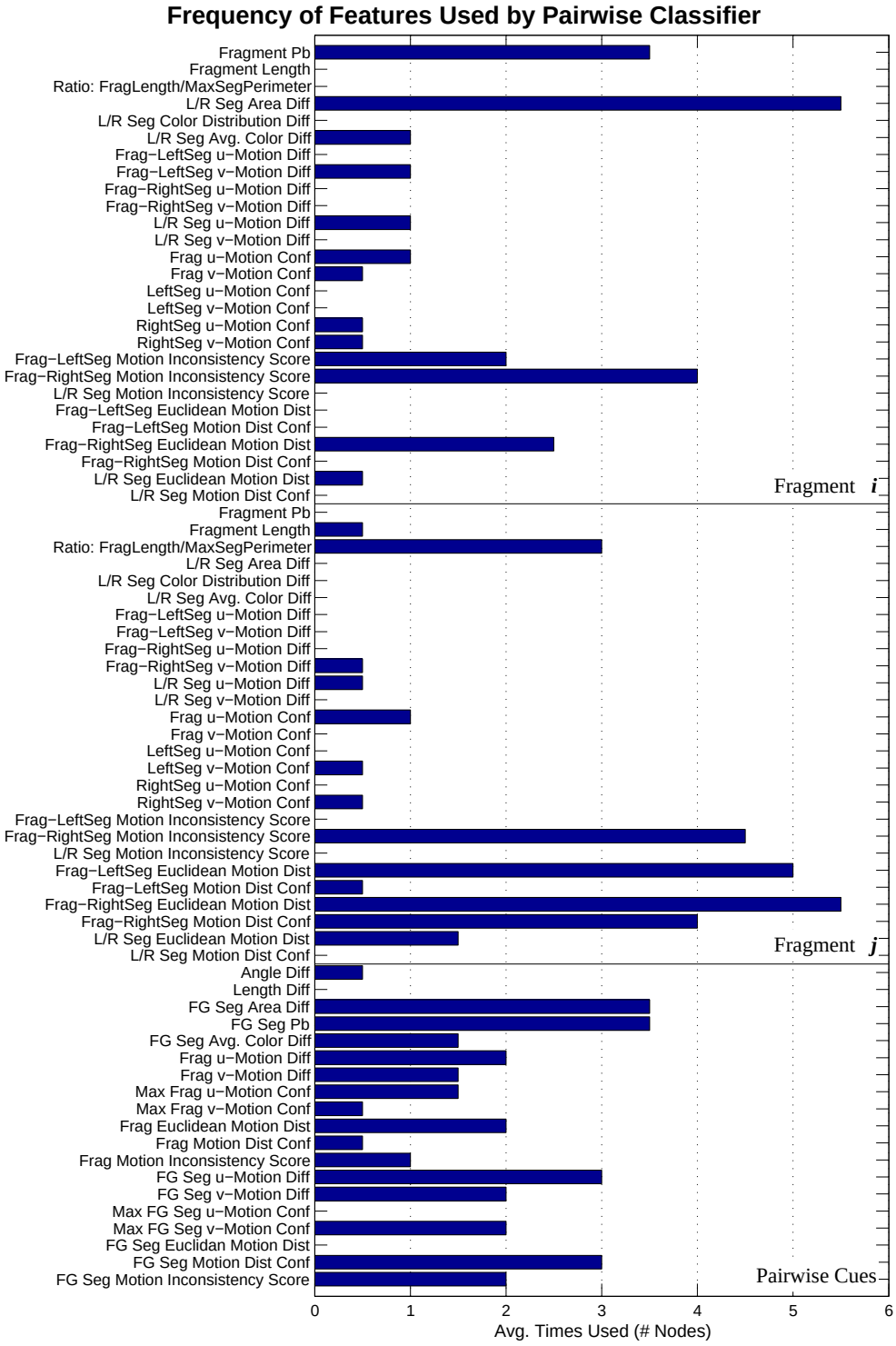


Figure 4.9: Frequency with which each computed feature is used by the learned pairwise fragment classifier.

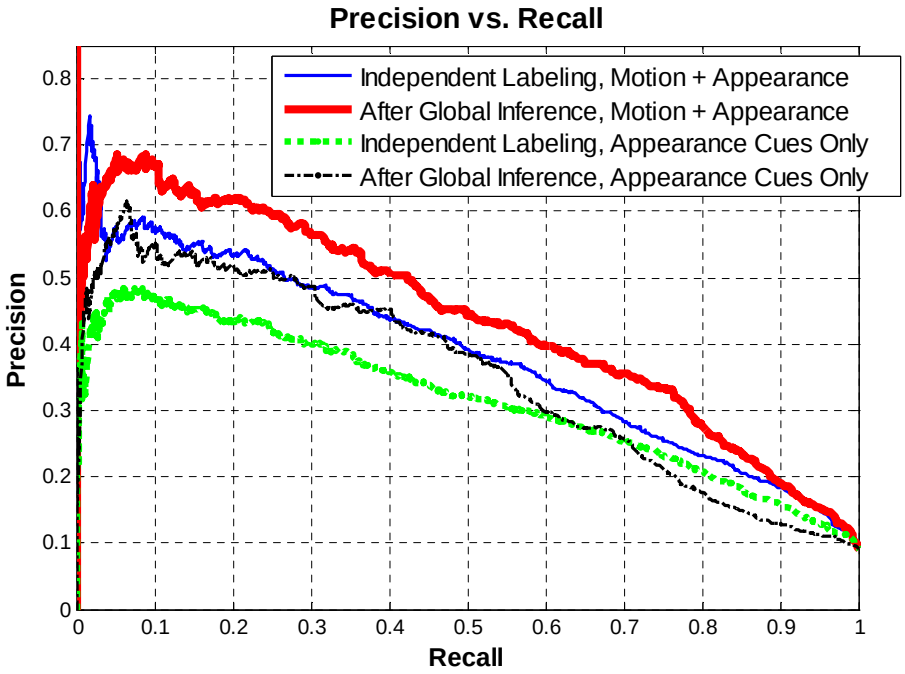


Figure 4.10: Precision vs. Recall for the entire dataset, showing that using motion and global inference results in the most accurate identification of those edge fragments which are occlusion/object boundaries.

boundaries) provides some indication of the difficulty of our task and our dataset.

While aggregate statistics captured by precision-recall plots are useful for understanding performance in general, they do hide important semantic measures of quality which can only be understood by looking at individual results. In Figures 4.11-4.14, we provide a few such examples out of the 30 in our database. (Additional examples can be found in Appendix B.) In each, the reference image of the sequence and the ground truth labeling are provided in the top row. In the remaining rows we compare the use of appearance only (left column) to that of appearance and motion combined (right column). The second row displays fragments overlaid on the image with brightness and line width proportional to the confidence that they are occlusion boundaries according to their *independent* classification results (*i.e.* before global inference). Thus, the brighter red and thicker a fragment is, the more the system believes it to be an occluding boundary. The next row displays the same type of result but *after* performing global inference on the initial classifications. The final two rows show these global inference results thresholded at equal recall rates for fair comparison. Note how the motion-plus-appearance approach sustains higher precision (*i.e.* fewer false positives) even as the recall is increased. This indicates that the motion adds significant confidence to the classifier's decision.

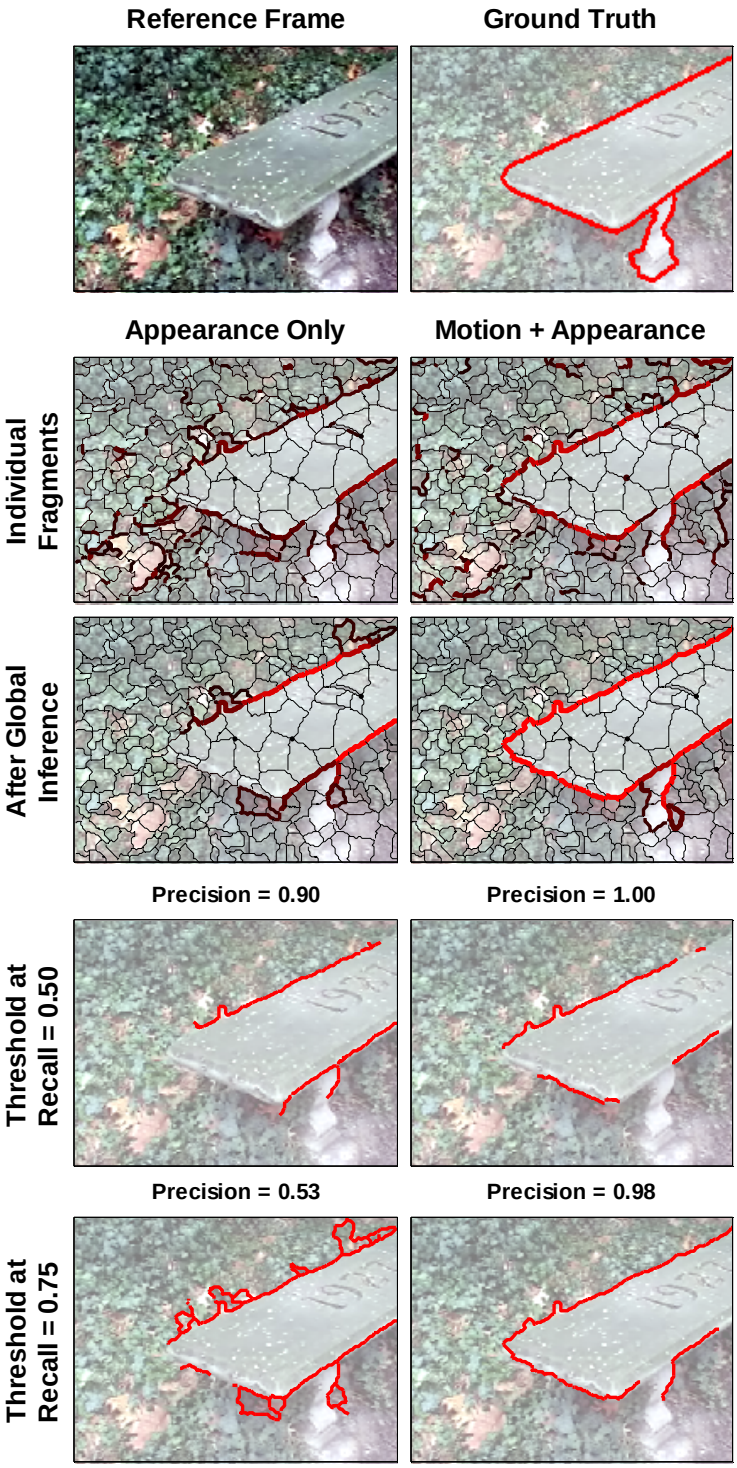


Figure 4.11: Example result: The appearance-only classifier's lack of confidence becomes obvious when we use use a higher-recall operating point. With the addition of motion, very high precision is maintained.

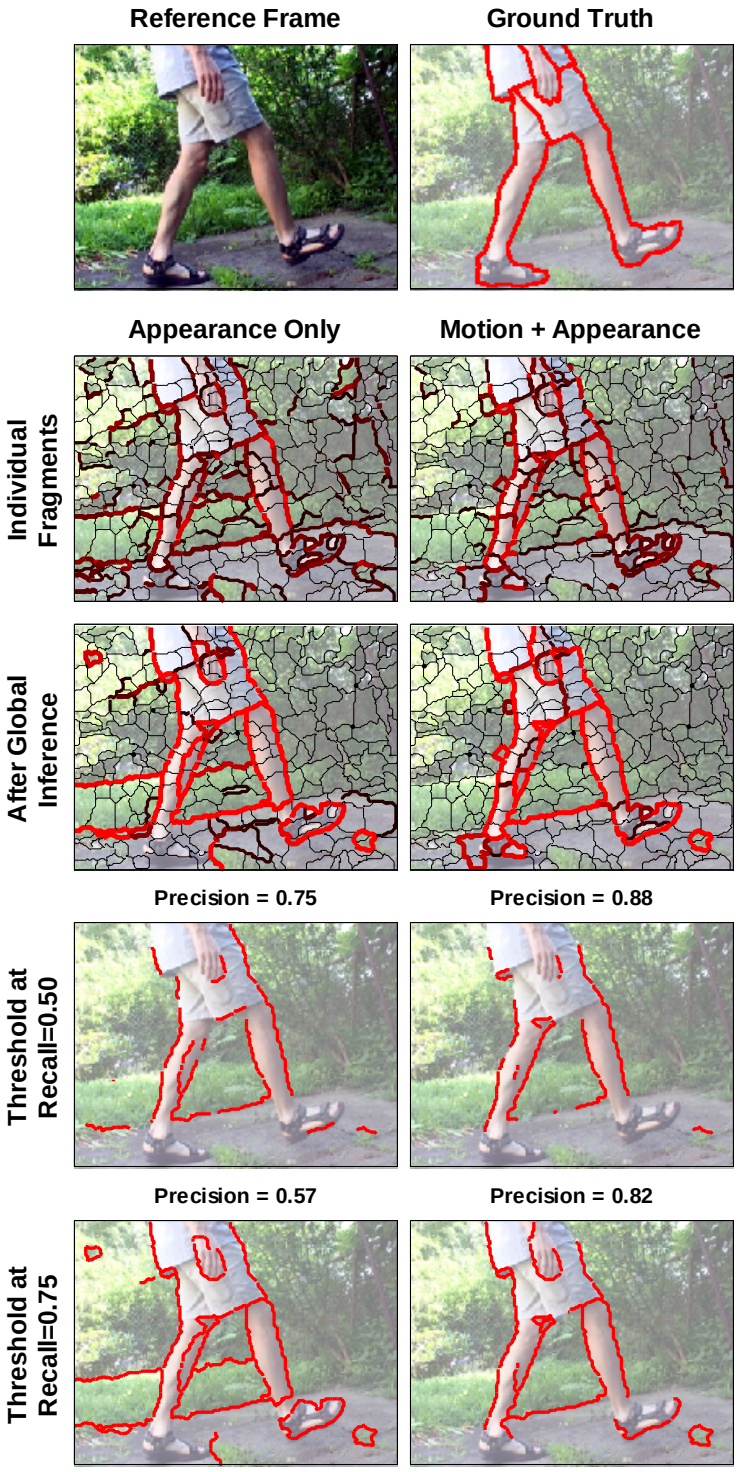


Figure 4.12: Example result: Using motion allows for sustained high precision, even at higher recall.

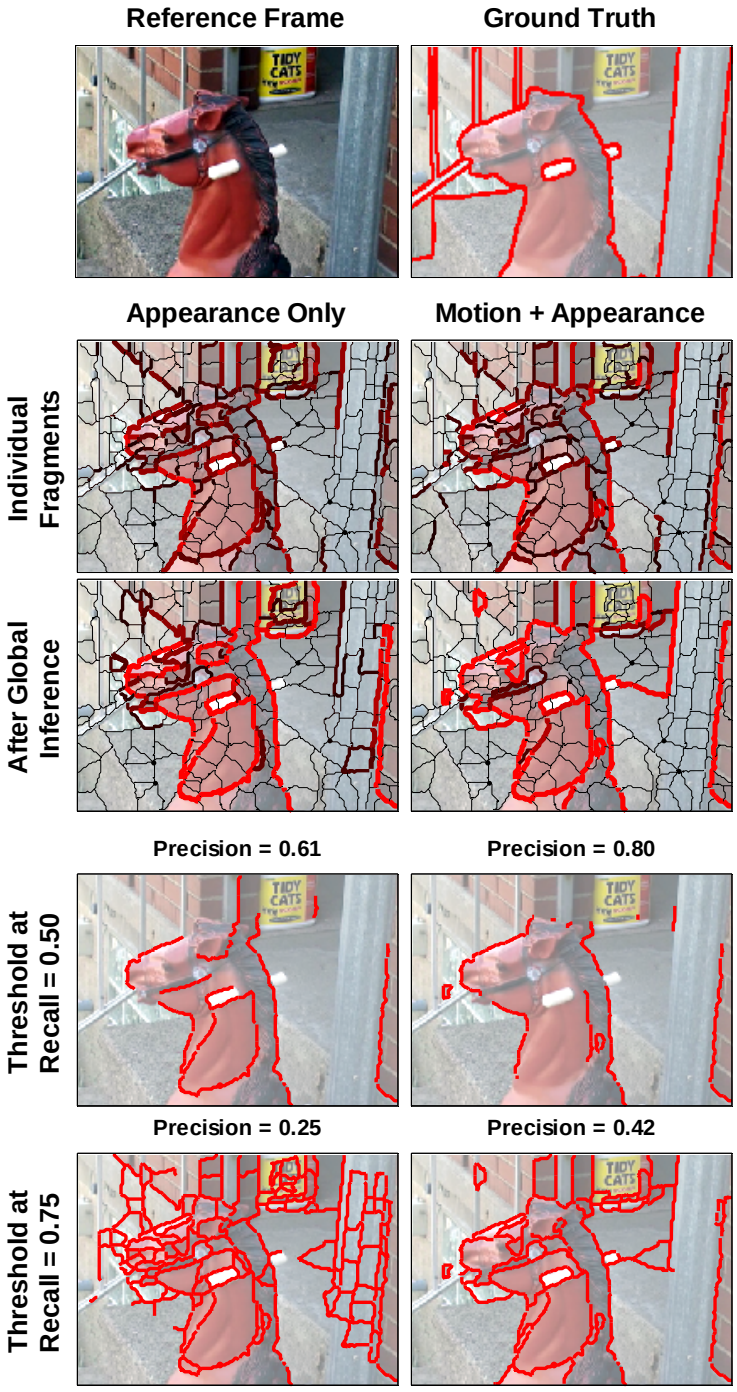


Figure 4.13: Example result: On this more difficult example, the motion+appearance classifier still performs best.

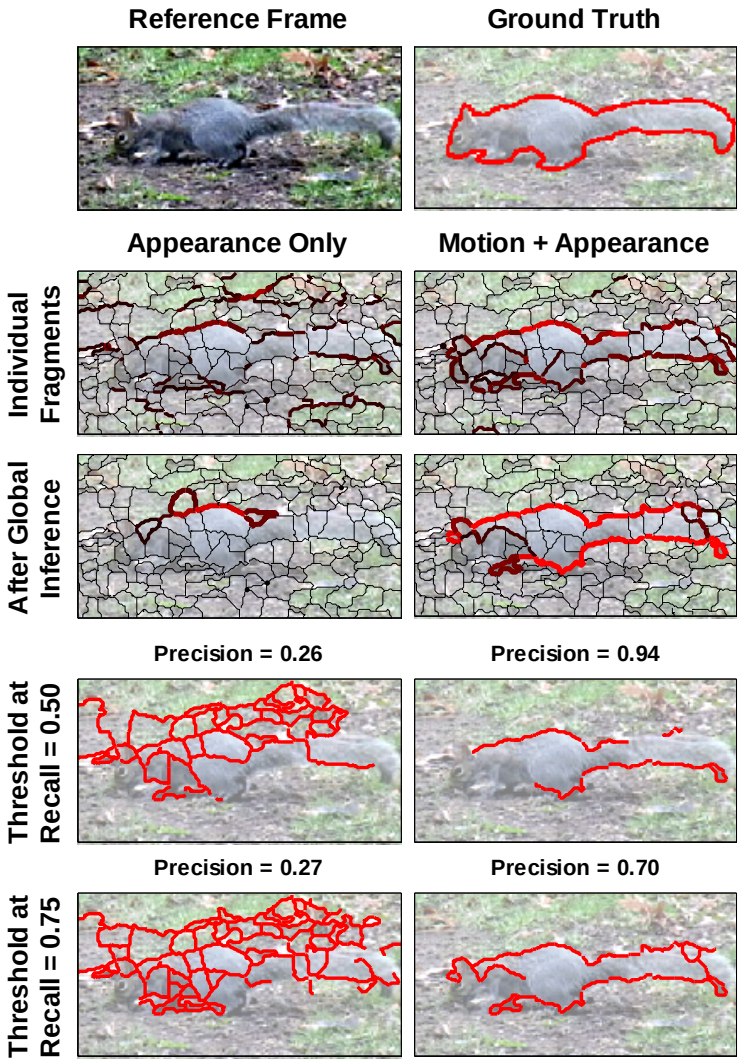


Figure 4.14: Example result: The squirrel in this scene is nearly invisible to the appearance-based classifier, but its movement makes its boundaries much more readily detectable when also using motion cues.

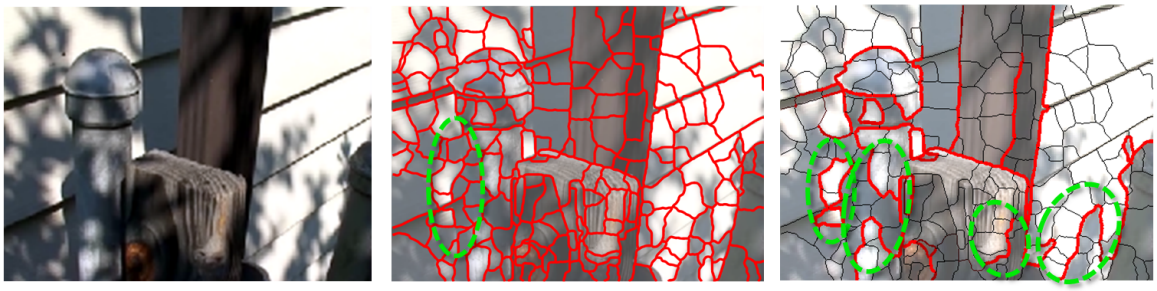


Figure 4.15: A harshly-lit, low-texture scene (left) causes difficulties for our approach. The middle example shows the over-segmentation failing to offer the true occlusion boundary as a hypothesized fragment for classification. At the right, we see false positives resulting from a combination of harsh lighting and lack of texture which would provide good motion estimates.

In Figure 4.15, we provide an example of a scene on which performance is consistently poor using our approach. This stems largely from two problems. First, the extremely harsh lighting creates high-contrast edges which result in an over-segmentation whose fragments do not correspond to the true occlusion boundaries in the scene. In such cases (which are rare, see Figure 4.7), there is no hope of the subsequent classifier succeeding, since the true boundaries are not even in the set of fragments to be classified. Secondly, the lack of texture (in combination with the lighting) results in several false positives since it prevents confident motion estimation which could otherwise be used to disambiguate and correctly classify lighting-based edges as *non*-boundaries.

Due to the variety of scenes in our dataset, it is not surprising that there exist cases for which motion does not help. Some object boundaries simply *are* easily identified by basic appearance cues, such as color, and the scenes may lack enough texture or depth variation to provide the necessary relative motion cues. However, it is also very rare that using motion *hurts* performance, and in those cases where appearance information simply does not capture the properties of occlusion boundaries well, motion cues often provide substantial improvement. This improvement is realized in reduced false positives since, in many cases, only motion information may allow the system to recognize and filter out high-contrast surface markings which confuse an appearance-only approach.

4.5 Discussion

In this chapter, we presented a mid-level framework capable of reasoning more globally about the existence of occlusion boundaries. In so doing, we alleviated deficiencies in earlier approaches by improving spatial support based on an initial over-segmentation, by incorporating boundary motion as well as motion confidence information, and by

introducing a graphical model with learned classifiers as potentials for propagating local information derived from a combination of appearance and motion cues.

In the future, it may be interesting to explore the use of feedback between the final boundary labeling and the over-segmentation processes in order to improve iteratively the hypothesis of boundaries, the subsequent delineation of spatial regions of support for feature extraction, and the labeling of those hypotheses based on the improved features. Initial results using this type of reasoning for segmentation, via iterative region merging, was recently presented in [76].

Now armed with a capable boundary detection approach, we will present in the following chapters improvement in higher-level processes, including object segmentation (Chapter 5) and object recognition (Chapter 6), by incorporating boundary information.

Chapter 5

Boundary-Based Matting for Improved Object Segmentation

It is well known that general *scene* segmentation is a poorly-defined problem whose "correct" solution is largely dependent on the application, if not completely subjective. Objective evaluation of segmentations is itself the subject of significant research (see [177] for a recent review). Here we consider instead the more specific problem of *whole object* segmentation; *i.e.*, our goal is to accurately and concisely segment the foreground objects or "things" without necessarily worrying about the background or "stuff" [2]. As we will explain, we use hypothesized boundary fragments to suggest partial segmentation "hints" to achieve this goal. Once the objects of interest are identified (which admittedly could itself involve some subjectivity), it becomes somewhat easier to define natural and intuitive measures of segmentation quality on a per-object basis.

In addition, segmentation results are intimately tied to the selection of the parameter controlling the granularity of the segmentation (*e.g.*, the number of segments or clusters, or a bandwidth for kernel-based methods). Instead of seeking a single perfect segmentation, the integration of *multiple* segmentations obtained over a range of parameter settings has become common [75, 110, 137, 146, 172]. In such approaches, segmentation is treated as a mid-level processing step rather than an end goal. This reduces the pressure to obtain — or even define — the one "best" result, while also side-stepping the problem of parameter selection.

We are motivated by approaches such as [146], in which Russell *et al.* showed that it is possible to discover objects in an unsupervised fashion from a collection of images by relying on multiple segmentations. Systems using segmentation in this way, effectively as a *proposal mechanism*, should benefit from an underlying segmentation method which accurately and frequently delineates whole objects. Thus we will present a novel segmentation strategy designed to outperform existing methods in terms of two inter-related and intuitive measures

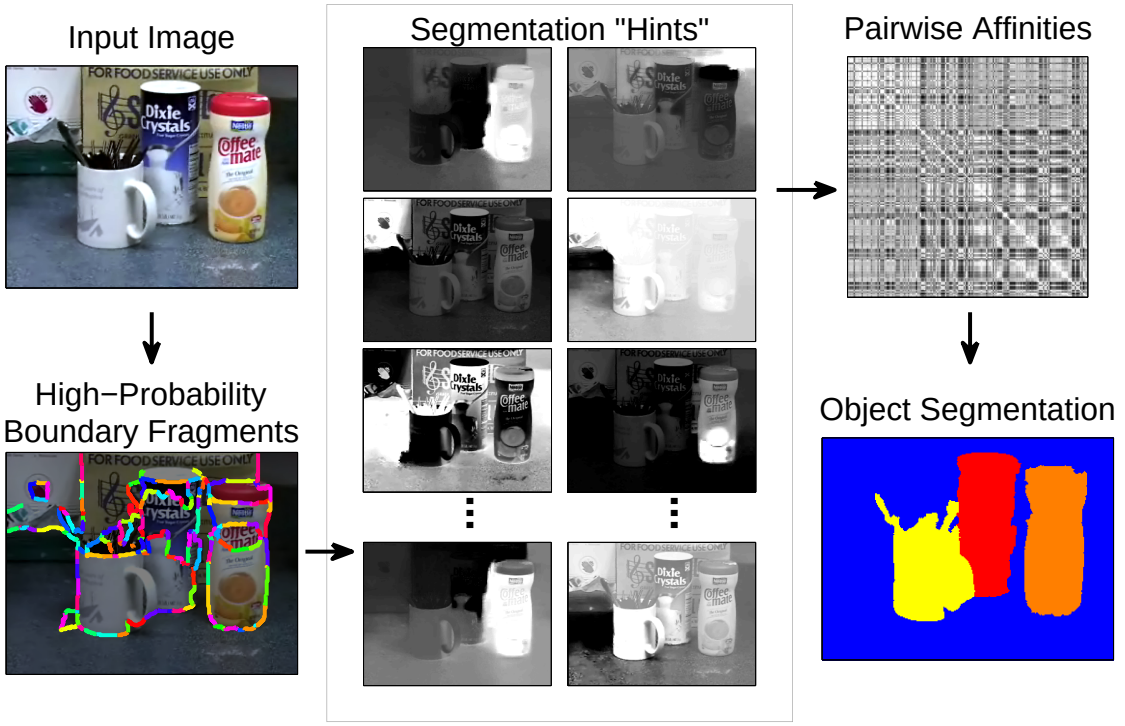


Figure 5.1: Starting with an image and hypothesized boundary fragments (only high-probability fragments shown for clarity), we generate a large set of segmentation "hints" using automated matting. Combining information from those mattes into an affinity matrix, we can then generate a segmentation for each of the foreground objects in the scene.

of object segmentation quality. First, we take into account accuracy in terms of pixelwise, *per-object* agreement between segments. Second, we also consider conciseness in terms of the number of segments needed to capture an object.

Following the classical Gestalt view [145] as well as Marr's theories [112], we recognize that a key cue in differentiating objects from each other and their background lies in the discovery of their boundaries. Therefore we use the hypothesized boundary fragments described in the previous chapter as input for our segmentations. Of course, perfect boundary extraction would imply perfect object segmentations, but the challenge in practice is to derive strong evidence for segmentation from imperfect, fragmented boundary information. Not only do boundaries indicate the spatial extent of objects, however, they also suggest natural regions to use in modeling those objects' appearances. The pixels on either side of a boundary provide evidence which can offer a "hint" of the correct segmentation by indicating discriminative appearance information, even if that information is only local and partial in nature. The overall procedure is illustrated in Figure 5.1.

In practice, we use the soft classification offered by α-matting approaches to generate

Figure 5.2: Example matting results from [100]. The input image is on the left of each pair, shown with user-marked foreground (white) and background (black) pixels.

these segmentation hints. There has been a recent surge of interest and impressive results in *interactive* approaches for matting [5, 9, 32, 100, 149, 171]. Using only a sparse set of user-specified constraints, usually in the form of "scribbles" or a "tri-map", these methods produce a soft foreground/ background matte of the entire image. Example matting results from [100], the method we will use in our work, can be found in Figure 5.2.

As initially demonstrated in [6], such methods can potentially be automated by replacing user-specified "scribbles" with constraints indicated by local occlusion information. Thus we will use each hypothesized boundary fragment from the approach presented in Chapter 4 to provide the matting constraints. Based on the combination of the set of mattes suggested by all of the fragments, we then derive an affinity measure which is suitable for use with any subsequent clustering algorithm, *e.g.* Normalized Cuts [159]. Through this combination of individually-weaker sources of information (*i.e.* fragmented boundaries and impoverished, binary mattes), we relieve the pressure to extract the one "perfect" matte or the "true" set of extended boundary contours, and yet we are still able to segment multiple objects in the scene accurately and concisely. Though beyond the scope of this work, it is our hope that such improved segmentation strategies will benefit methods which rely on multiple segmentations to generate, for example, object models automatically from unlabeled data.

In the remainder of the chapter, we will describe the details of how we generate a set of multiple mattes from given boundary fragments, combine those mattes into affinities, and then segment scene objects. We will also provide an intuitive method of evaluating the ability of a set of segmentations to depict whole objects accurately and parsimoniously according to ground truth. Finally, we offer experiments comparing our method to other existing segmentation methods from the literature.

5.1 Segmentation "Hints" via Multiple Mattes

As discussed above, we will use image matting to produce segmentation "hints" for generating affinities useful in segmentation via clustering. After first providing an overview of matting, we will explain how we use our boundary fragments to imply *multiple* mattes for estimating these affinities.

5.1.1 α-Matting

In a standard matting approach, one assumes that each observed pixel in an image, $\mathcal{I}(x, y)$, is explained by the convex combination of two unknown color classes, F and B. A soft weight, α, controls this combination:

$$\mathcal{I}(x, y) = \alpha(x, y)F(x, y) + (1 - \alpha(x, y))\, B(x, y) \tag{5.1}$$

In this formulation, then, pixels with an α-value near one are clearly part of the F class, while those with an α-value near zero are clearly part of the B class. Values near 0.5 indicate mixed pixels whose "membership" may be considered unknown. Typically, F and B correspond to notions of "foreground" and "background", but we will explain in the next section that these semantic assignments are not necessary for our approach.

Simultaneously solving for F, B, and α in (5.1) is of course not feasible. In general, a user specifies a small set of pixels, often referred to as "scribbles" or a "tri-map", which are then constrained to belong to one class or the other. These hard constraints are then combined with assumptions about α (*e.g.*, smoothness) to find a solution at the unspecified pixels [5, 9, 32, 100, 149, 171]. We have adopted the approach proposed by Levin *et al.* [100], which offers excellent results via a closed-form solution for α based on reasonable assumptions about the local distributions of color in natural images.

Note that methods also exist for constrained "hard" segmentations, *e.g.* [24] — potentially with some soft matting at the boundaries enforced in post-processing [143] — but as we will see, using fully-soft α-mattes allows us to exploit the uncertainty of mixed pixels (*i.e.* those with α values near 0.5) rather than arbitrarily (and erroneously) assigning them to one class or the other. In keeping with the philosophy of the rest of this thesis, we follow the conventional wisdom of avoiding this kind of early commitment throughout our approach. Hard thresholds or grouping decisions are avoided in favor of maintaining soft weights until the final segmentation procedure. In addition to retaining the maximum amount of information for the entire process, this methodology also avoids the many extra parameters required for typical *ad hoc* decisions or thresholds.

5.1.2 Multiple Mattes → Affinities

In an *automated* matting approach, the goal is to provide a good set of constraints without requiring human intervention. Since object boundaries separate two different objects by definition, they are natural indicators of potential constraint locations: F on one side and B on the other. In [6], T-junctions were used to suggest sparse constraints in a similar manner. An example input image and its boundary fragments (shown in differing colors), can be found on the left side of Figure 5.1. For clarity, only the high-probability fragments are displayed, though *all* are used for the matting procedure described below. The benefit of matting techniques here is their ability to propagate throughout the image the appearance information offered by such local, sparse constraints.

For each fragment, we can generate an image-wide matte according to [100] as described in Section 5.1. The middle of Figure 5.1 depicts a sampling of such mattes generated from the differing constraints (or "scribbles") implied by various fragments. The *super-pixels*[1] on either side of a fragment naturally designate spatial support for the required labeling constraints. Since super-pixels from an over-segmentation generally capture fairly homogeneous regions, however, we have found it better to expand both the F and B constraint sets for a fragment. For each, we therefore use a *triplet* of super-pixels formed by incorporating the super-pixels of the two neighboring fragments most likely to be boundaries. This process is illustrated in Figure 5.3. Note also that our approach avoids any need to choose the "right" boundary fragments (*e.g.* by thresholding), nor does it explicitly attempt to extract extended contours using morphological techniques or some sort of *ad hoc* chaining procedure, both of which are quite brittle in practice, as discussed previously. Instead we consider only the individual short fragments, which have an average length of 18 pixels in our experiments.

A remaining problem is that the approach described thus far only offers a binary (albeit soft) decision about a pixel's membership: it must belong to either F or B, which are usually understood to represent foreground and background. How then can we deal with layered, multi-object scenes?

We recognize that the actual class membership of a particular pixel, as indicated by its α value in a single matte, is rather meaningless in isolation. What we wish to capture, however, is that locations with similar α values in a given matte (whether both zero *or* one) are more likely to belong to the same object, while locations with different values are more likely to be part of different objects. While existing methods, such as Intervening Contours [36, 56, 99], may attempt to perform this type of reasoning over short- and medium-ranges within the image using standard edge maps, our use of mattes simultaneously factors in the boundaries

[1] In this chapter, we will refer to the "segments" of the *over*-segmentations from Chapter 4 as "super-pixels" instead [141]. This is to avoid confusion with the "segments" of the final *object* segmentations we seek in this chapter.

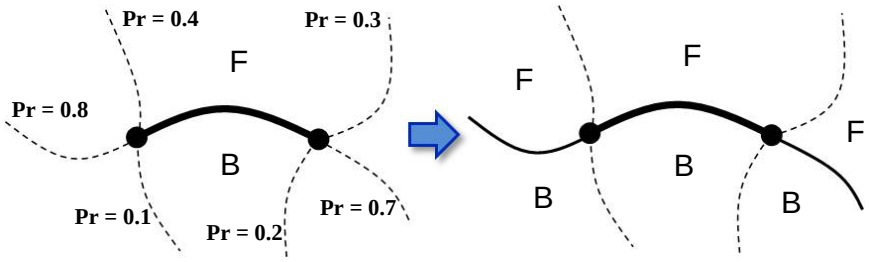


Figure 5.3: The F and B constraint sets for a given fragment are initialized to its two neighboring super-pixels (left). To improve the mattes, these constraint sets are expanded to also include the super-pixels of the fragment's immediate neighbors with the highest probability of also being object boundaries (right). Thus we use a triplet of super-pixels to specify both the F and B constraint sets at a fragment.

themselves as well as the *global appearance discrimination* they imply. Moreover, matte values near 0.5 indicate *uncertainty* about the relationship between locations.

Each of the N_F potential boundary fragments in the scene generates an image-wide matte in this manner, yielding an N_F-length vector, v_i, of matting α-values at each pixel i. If we scale the α-values to be between -1 and $+1$ (instead of 0 and 1), such that zero now represents "don't know", then the agreement, or *affinity*, between two pixels i and j can be written in terms of the normalized correlation between their two scaled matting vectors:

$$A_{ij} = \frac{v_i^T W v_j}{|v_i||v_j|}, \tag{5.2}$$

where W is a diagonal matrix of weights corresponding to the likelihood of each fragment actually being an object boundary, as reported by the approach described in Chapter 4. Thus mattes derived from fragments less likely to be object boundaries will not significantly affect the final affinity. Note that this combines *all* hypothesized fragments in a soft manner, avoiding the need to choose some ideal subset, *e.g.*, via thresholding. Figure 5.4 provides an example schematic describing the overall process of employing boundary fragments to suggest mattes, which in turn generate an affinity matrix. In [44], a different but related method for image segmentation is presented which also combines various individually-weaker partitions (provided by proposals of sources and sinks in a min-cut framework).

The value of A_{ij} will be maximized when the matting vectors for a pair of pixels usually put them in the same class. When a pair's vectors usually put the two pixels in opposite classes, the affinity will be minimized. The normalization effectively handles discounting the "don't know" (near-zero) entries which arise from mattes that do not provide strong evidence for one or both pixels of the pair. Our use of pairwise comparisons in this manner captures whether locations in the image are part of the same or different objects. Therefore, despite the fact that each *individual* hint only suggests a (soft) binary segmentation of the

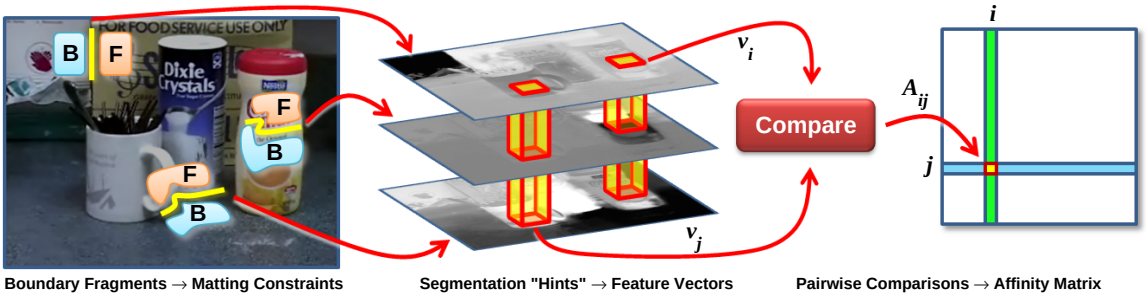


Figure 5.4: Each potential boundary fragment found in the original image (three shown in yellow here) implies a set of constraints (F and B) used to generate a matte. Vectors of matting results at each location in the image (v_i and v_j) can be compared in a pairwise fashion to generate an affinity matrix, A, suitable for clustering or segmentation.

image, the *combination* of many different hints allows the actual number of objects to remain unknown at this point — and certainly more than two (*i.e.* simple "foreground" and "background"). Thus, using these subtle pairwise relationships in aggregate can yield segmentations with more than two classes.

We have now defined a novel matting-based affinity measure which can be used with any off-the-shelf clustering technique. This affinity takes into account boundary knowledge and utilizes global appearance reasoning. An additional unique and noteworthy aspect of our affinities is that they are defined based on feature vectors in a space whose dimension is a *function of the image content* — *i.e.* the number of detected fragments, N_F — as opposed to an arbitrarily pre-defined feature space of fixed dimension. In our view, it is intuitively preferable to use this type of data-driven feature space, whose complexity is likely better matched to the image content, rather than defining an *ad hoc* feature space which may not be appropriate for all images.

5.2 Image Segmentation by Normalized Cuts

Given our pairwise affinity matrix A, which defines a *fully-connected*, weighted graph over the elements in our image, we will use spectral clustering according to the Normalized Cut Criterion to partition it into an image segmentation with K segments [159]. To obtain a set of multiple segmentations, we can simply vary this parameter K within a specified range.

Using the boundary-detection approach described in Chapter 4, we not only obtain a better set of hypothesized boundary fragments and probabilities of those fragments corresponding to physical object boundaries (*i.e.* for use in W from (5.2)), but we can also use the computed super-pixels as our basic image elements instead of relying on individual pixels. In addition to offering improved, data-driven spatial support, this *drastically* reduces

the computational burden of constructing (and storing) A_{ij}, making it possible to compute *all* pairwise affinities between super-pixels instead of having to *sample* pixelwise affinities very sparsely.

The benefit here is more than reduced computation, however, particularly when using the popular spectral clustering technique of Normalized Cuts ("NCuts"). Other authors have noted that non-intuitive segmentations typical of NCuts stem from an inability to fully populate the affinity matrix or the topology of a simple, four-connected, pixel-based graph [174]. By computing affinities between all pairs of *super*-pixels, we alleviate somewhat both of these problems while also improving speed.

We will not review here all the details of spectral clustering via the Normalized Cut criterion (see [127] for details), but the general approach involves computing the eigensystem of the graph Laplacian $L = D^{-1}A$, where D is a diagonal matrix whose i^{th} entry along the diagonal is the sum of the i^{th} row of our matting-based affinity matrix, A. The final clustering of the data is determined by discretizing into K clusters the eigenvectors of L corresponding to the largest eigenvalues. Different methods exist for this discretization step, but we use k-means on the rows of the matrix whose columns are the top K eigenvectors of L [127].[2] For multiple segmentations, we can compute the eigensystem of L just once and simply repeat the discretization for each specified value of K.

We compare our segmentations obtained using NCuts on our matting-based affinities to two popular segmentation approaches from the literature, each of which also uses NCuts and offers a publicly-available implementation. First is the recent multiscale approach of Cour *et al.* [36], in which affinities are based on Intervening Contours [99]. Second, we compare to the segmentations produced by the Berkeley Segmentation Engine (BSE) [56, 114], which relies on a combined set of patch-based and gradient-based cues (including Intervening Contours). For discretizing the eigenvectors of the graph Laplacian, the BSE also uses k-means, while the multiscale NCuts method uses a discretization procedure which attempts to find an optimal rotation of the normalized eigenspace [196]. Finally, both methods compute a sparsely-sampled, *pixelwise* affinity matrix.

Certainly, other image segmentation techniques exist. We performed some initial experiments using mean-shift [35], as well as medoid-shift [157], directly in the space of matting vectors, v. These methods did not perform as well for our goal of whole-object segmentation, however, and may be better-suited to scene segmentation or *over*-segmentation. This may be in part due to the sparse population of the high-dimensional matting-vector space, particularly when super-pixels are used (this is especially true for medoid-shift). We also found it more natural to specify a range of K for NCuts than to specify the bandwidth parameters for mean-shift. Finally, the use of NCuts offered a

[2] We also note here the connection between the use of k-means to cluster the graph Laplacian's eigenspace and the grouping of our matting vectors captured by our affinity measure in (5.2).

straightforward comparison to the two recent methods described above.

It is also worth noting that methods do exist for attempting to select automatically the best K for a particular clustering using NCuts. We implemented the approach described in [8], which considers a random walk on the graph implied by the affinity matrix in order to suggest a small set of values for K. This method is geared towards use in multi-scale setting, so there are still multiple values (typically three to five) for K. Thus, we are still left with a set of *multiple* segmentations, albeit a much smaller one. Though theoretically appealing, it was not clear that using this smaller set of segmentations offered any advantage over using the full set resulting from using the whole range of K — beyond *potential* computational savings. In other words, the set of K chosen by the algorithm in [8] did not necessarily correspond to the best settings for producing segments corresponding to whole objects, as discussed in more detail in the next section.

5.3 Evaluating Object Segmentations

In this section, we will present an intuitive approach for determining whether one set of segmentations is "better" than another set generated using a different method. For the eventual goal of object segmentation and discovery, we propose that the best set of segmentations will contain at least one result for each object in the scene, in which that object can be extracted accurately and with as few segments as possible. Ideally, each object would be represented by a single segment which is perfectly consistent with the shape of that object's ground truth segmentation.

Thus, we define two measures to capture these concepts: *consistency* and *efficiency*. For consistency, we adopt a common metric for comparing segmentations, which conveniently lies on the interval $[0, 1]$ and captures the degree to which two sets of pixels, R and $\mathcal{G}$, agree based on their intersection and union:

$$c(R, \mathcal{G}) = \frac{|R \cap \mathcal{G}|}{|R \cup \mathcal{G}|}. \tag{5.3}$$

Here, $R = \{A, B, C, \dots\} \subseteq \mathcal{S}$ is a subset of segments from a given (over-)segmentation $\mathcal{S}$, and $\mathcal{G}$ is the ground truth object segmentation.

At one extreme, if our segmentation were to suggest a single segment for each pixel in the image, we could always reconstruct the object perfectly by selecting those segments (or in this case, pixels) which corresponded exactly to the ground-truth object's segmentation. But this nearly-perfect consistency would come at the cost of an unacceptably high number of constituent segments, as indicated in the rightmost example in Figure 5.5. At the opposite extreme, our segmentation could offer a single segment which covers the entire object, as shown in the leftmost example. In this case, we would achieve the desired minimum of one segment to capture the whole object, but with very poor consistency.

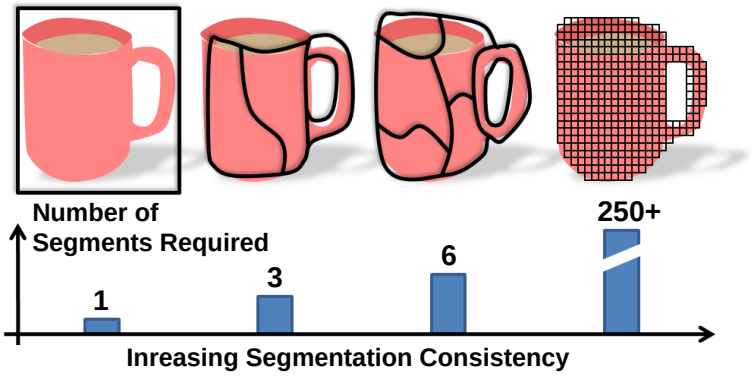


Figure 5.5: The trade-off between segmentation consistency (or accuracy) and efficiency (or the number of segments required to achieve that consistency). As desired consistency increases, so too does the number of segments required to achieve that consistency.

Thus we see that there exists a fundamental tradeoff between consistency and the number of segments needed to cover the object, or *efficiency*.[3] Therefore, when evaluating the quality of a scene's (over-)segmentation $\mathcal{S}$, we must take into account both measures. We define the efficiency as the size of the minimal subset of segments, R, required to achieve at least the desired consistency, $\mathbf{c}_d$, according to the ground truth object segmentation, $\mathcal{G}$:

$$e_{\mathbf{c}_d}(\mathcal{S}, \mathcal{G}) = \min |R|, \text{such that } c(R, \mathcal{G}) \geq \mathbf{c}_d. \tag{5.4}$$

Note that with this definition, a *lower* value of $e(\mathcal{S}, \mathcal{G})$ implies a *more* efficient (or parsimonious) segmentation.

We can now specify $\mathbf{c}_d$ and compute the corresponding $e_{\mathbf{c}_d}$, which is equivalent to asking, "what is the minimum number of segments required to achieve the desired consistency for each object in this scene?" By asking this question for a variety of consistency levels, we can evaluate the quality of a particular method and compare it to other methods' performance at equivalent operating points.

Referring once again to Figure 5.5, the middle two examples indicate that we can achieve a reasonable level of consistency with only three segments, or if we raise the desired consistency a bit higher (perhaps in order to capture the top of the mug), it will require us to use a different segmentation from our set which covers the mug with six segments. In general, the best method would be capable of providing at least one segmentation capable of yielding a desired high level of consistency with the minimum degree of over-segmentation of any particular object, *i.e.*, we seek minimal $e_{\mathbf{c}_d}(S, \mathcal{G})$ at maximal $\mathbf{c}_d$.

The search over all segment subsets R possibly required to achieve a particular $\mathbf{c}_d$ is combinatorial. This is only an issue for *evaluation* and not at run time, but it is still desirable

[3] It may be helpful to think of these measures and their tradeoff as being roughly analogous to the common notions of precision and recall, *e.g.*, in object recognition.

to avoid a computational explosion that could make assessment of results intractable. In practice, we only need to consider those segments which actually overlap the ground truth object (other segments can only *decrease* consistency if included). We also limit the number of those overlapping segments we are willing to use, and we select them in order of their individual consistencies with the object of interest, *i.e.* according to (5.3). Using no more than twenty such segments keeps computation tractable and still allows us to consider *every* overlapping segment for nearly all examples. Finally, since we are really only interested in segmentations which require a relatively small number of segments to capture an object, we limit the size of R to a maximum of ten segments. Thus, in practice we evaluate the consistency of all 20-choose-k combinations of our selected subset of overlapping segments and choose the combination with the highest consistency for each value of $k \in \{1, \ldots, 10\}$.

5.4 Experiments

For each of a set of test scenes, we have labeled the ground truth segmentation for foreground objects of interest, which we roughly defined as those objects for which the majority of their boundaries are visible. From the scenes in our dataset, we have labeled ground truth segmentations for 50 foreground objects on which we will evaluate our approach.

We generate a set of segmentations for each scene by varying K between 2 and 20 while using either our matting-based approach, multiscale NCuts [36], or the BSE approach [56, 114].[4] Recall that the latter two methods compute affinities in a *pixelwise* fashion. To verify that any improvement offered by our approach is not *solely* due to our use of super-pixels, we also implemented a baseline approach which computes affinities by comparing pairs of L*a*b* color distributions, obtained via kernel density estimates within each super-pixel. As was used earlier in this thesis, we compare distributions with the χ^2-distance metric.

For our proposed approach, we use each of the individual boundary fragments to suggest constraints for mattes as described in Section 5.1.2. In practice, we ignore those fragments with an extremely low probability (less than 0.01) of being boundaries, since the resulting mattes in those cases would have almost no effect on computed affinities anyway. From an initial set of 350-1000, this yields approximately 90-350 fragments (and mattes) per image for computing pairwise, segment affinities according to (5.2).

We first selected a set of ten desired consistency levels, from 0.5 to 0.95. Then for each labeled object in a scene and for each segmentation method, we pose the question, "what is the minimum number of segments required to achieve each desired consistency in segmenting this object?" We can then graph and compare the methods' best-case efficiencies

[4] Note that, via post-processing, we force segments to be spatially contiguous and to have a minimum area of 0.1% of the image area. This may add or remove segments, meaning that the actual total number of segments in a final segmentation may not strictly match the specified K.

as a function of the desired consistencies.

A typical graph is provided at the top of Figure 5.6, in which bars of different colors correspond to different segmentation methods. Each group of bars corresponds to a certain consistency level, and the height of the bars indicates the smallest number of segments required (or maximum efficiency possible) to achieve that consistency. Thus, *lower bars are better*. Bars extend to the top of the graph when a method required more than the maximum ten segments to achieve a particular desired consistency. Thus we see that our approach is able to achieve similar consistency with fewer segments — until $\mathbf{c}_d$ reaches 0.85, at which point all methods fail on this image. Not surprisingly, the relatively simplistic appearance model of the color-only baseline tends to over-segment objects the most.

We can also examine the actual segmentations produced by each method at corresponding desired consistency levels, as shown at the bottom of Figure 5.6. For the input image and selected ground truth object shown, we provide for comparison the individual segmentations produced by each method. Each displayed segmentation is the one which used the minimum number of segments while still achieving *at least* the desired consistency indicated to the left of the row. Also shown are the segments used for our method and the color distribution approach, along with a high-probability subset of the boundary fragments used in the proposed approach. We display only a subset of the fragments actually used simply for clarity. Below each segmentation are the actual consistencies and efficiencies (*i.e.* number of segments) obtained. For visualization, the individual segments corresponding to the object, *i.e.* those used to compute the displayed c and e values, have been colored with a red-yellow colormap, while background segments are colored blue-green. Note how our method achieves comparable consistencies with fewer segments — even when other methods may not be able to achieve that consistency at all. More results are provided in Figures 5.7-5.8 and in Appendix C.

Not surprisingly, when the desired consistency is fairly low, any method is usually capable of capturing an object with a minimal number of segments. But as the desired consistency increases, it becomes more difficult, or even impossible, to find a small number of segments to recover that object so accurately. Finally, as the desired consistency becomes too high, all methods begin to fail. We find, however, that for many objects our matting-based approach tends to maintain better efficiency (*i.e.* decreased over-segmentation) into a higher-consistency regime than the other methods.

To capture this more quantitatively over all objects, we can compute the difference between the number of segments our method requires at each consistency level and the number required by the other methods. We would like to see that our method often requires significantly fewer segments to achieve the same consistency. Certainly, there are some "easier" objects for which the choice of segmentation method may not matter, so we would also expect to find that our method regularly performs only as well as other methods. But

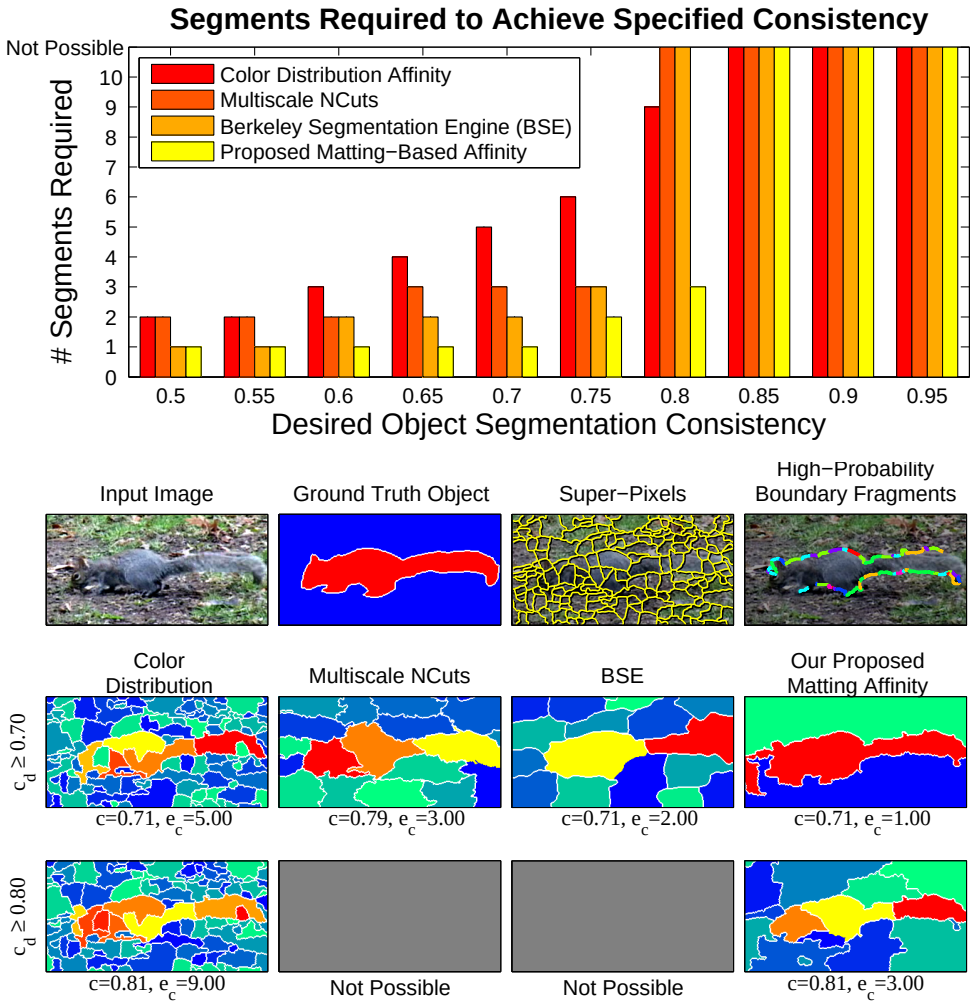


Figure 5.6: Top: A typical graph of segmentation efficiency vs. consistency for a set of desired consistency levels. Our method is able to achieve similar object segmentation consistency with fewer segments. **Bottom:** The corresponding input data (1^{st} row) and the resulting segmentations at two consistency levels (2^{nd}, 3^{rd} rows), as indicated by $\mathbf{c}_d = \{0.70, 0.80\}$ to the left of each row. For clarity, only high-probability boundary fragments used by our approach are shown, using a different color for each fragment. For visualization, the individual segments corresponding to the object, *i.e.* those used to compute the c and e values displayed below each segmentation, have been colored with a red-yellow colormap, while background segments are colored blue-green.

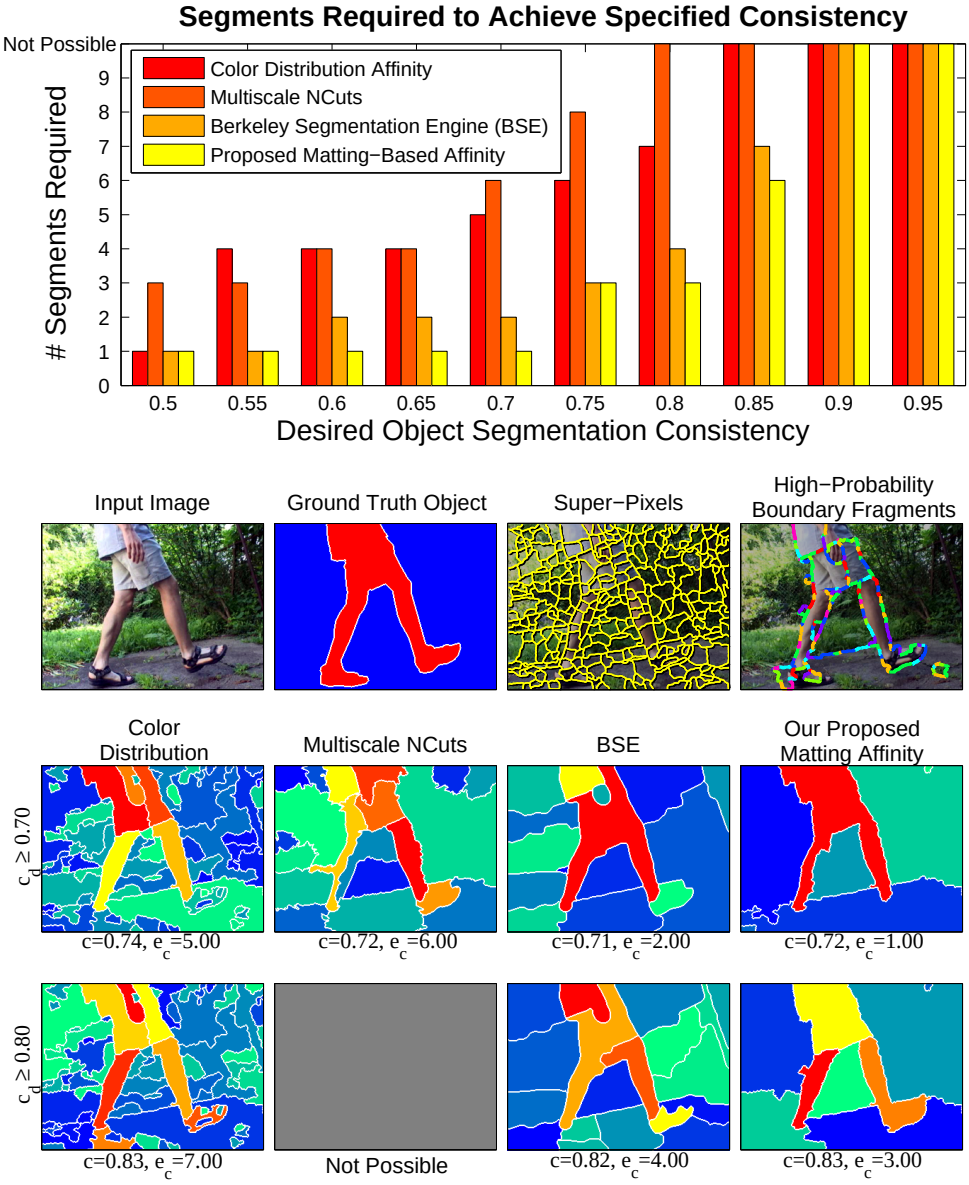


Figure 5.7: Another example showing our method's ability to achieve comparable consistency with fewer segments.

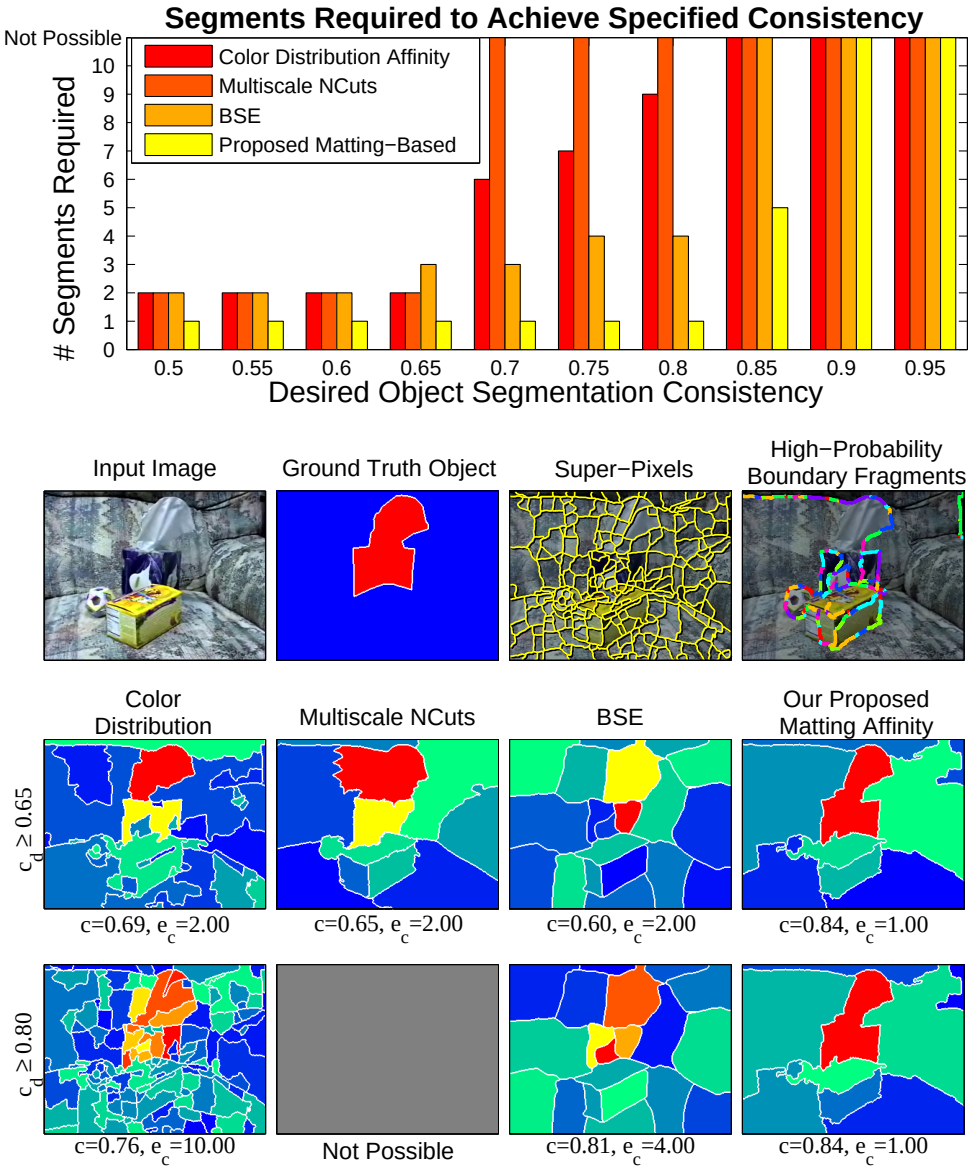


Figure 5.8: Another example like those in Figures 5.6-5.7. Note only one of the three objects in the scene is shown here.

we also must ensure that we rarely do *worse*. We also expect the potential benefit of our approach to be most obvious within a reasonable range of desired consistency: all methods will tend to perform equally as well at low consistencies, and all methods will tend to fail equally as often at very high consistencies.

Figures 5.9 and 5.10 offer evidence that our approach does indeed outperform the other methods as desired. Histograms in Figure 5.9 display the *relative* number of segments required by our approach as compared to the other methods. The height of the bars corresponds to the fraction of the total number of objects for which we achieve the specified relative efficiency on the x-axis. Thus, bars at zero, in the center of the graph, correspond to cases when we perform just as well as the other approaches. Bars to the right (left) correspond to cases where we perform better (*resp.*, worse), using fewer (*resp.*, more) segments than the competition. As indicated, each plot corresponds to a different desired consistency level (increasing down the rows). Bars at the extreme left and right also include complete failure cases. Summary statistics averaged over all objects and methods for our improvement or degradation in performance are also provided on the plots in Figure 5.9 and in separate graphs in top half of Figure 5.10. As expected, we perform just as well (*e.g.*, Figure C.1 or C.22) — or poorly (*e.g.*, Figure C.32) — as other methods for many of the objects, particularly at very low or very high desired consistencies. But we require fewer segments per object in a significant number of cases. Furthermore, our approach rarely *hurts*; we do not often require *more* segments on average than the other methods.

In the bottom half of Figure 5.10, we plot the same relative performance summary, but for separate experiments performed with the weights, W in (5.2), which control the contribution of each segmentation hint, derived from the average Pb strength along each fragment (*i.e.*, instead of using the boundary probability for our approach in Chapter 4). Note that performance is similar, indicating the power of the matting-based affinities independent from the method of boundary detection. However, improvement is achieved for fewer of the objects (top plot), and the discrepancy between the relative number of segments required is lessened (bottom plot). This indicates that our method of boundary reasoning also offers useful information, beyond the Pb measure, for effectively weighting the combination of individual segmentation hints.

5.5 Discussion

Debate continues over whether high-level reasoning, *e.g.* object *recognition*, should precede figure-ground perception [134], or if purely bottom-up, local reasoning can account for this process [135, 164]. Our work takes the more bottom-up perspective, since knowledge of specific objects is not required (unlike [140]), but our use of matting incorporates more powerful, semi-global reasoning as compared to purely-local methods.

Our experiments indicate that by maintaining "soft" reasoning throughout, and by combining multiple, individually-imperfect sources of information in the form of fragmented boundary hypotheses and oft-uncertain mattes, our novel method for addressing object segmentation yields promising results for use in subsequent work on unsupervised object discovery or scene understanding. While here we have evaluated our affinities in isolation, as compared to existing methods, it is certainly possible that *combining* multiple types of affinities would offer further improvement.

In this work, we employ the motion-based techniques for boundary fragment identification described in previous chapters in order to improve performance by offering better boundary estimates and, in turn, better mattes. We are *not*, however, incorporating motion estimates (*e.g.*, optical or normal flow) directly into our segmentation affinities [158, 188], nor is the approach described in this chapter fundamentally tied to the use of motion. Furthermore, for close-up static scenes where the only "motion" cue arises from the slight parallax generated by small camera movements, the macro-scale character of this motion may not be suitable for motion segmentation techniques which are designed to model or differentiate between gross, object-wide movements over longer timeframes.

Using boundaries and mattes as described simultaneously implies the *grouping* and *separation* of segments on the same or opposite sides of a given boundary, respectively. We also performed experiments with the technique described in [194] to incorporate *separately* such "attraction" and "repulsion" evidence via spectral clustering, rather than simply treating the two as equivalent sources of information with opposite signs. This sometimes yielded very good results, but it was fickle: when it failed, it often failed completely (*i.e.*, at all consistency levels). Future research in this direction is warranted, however.

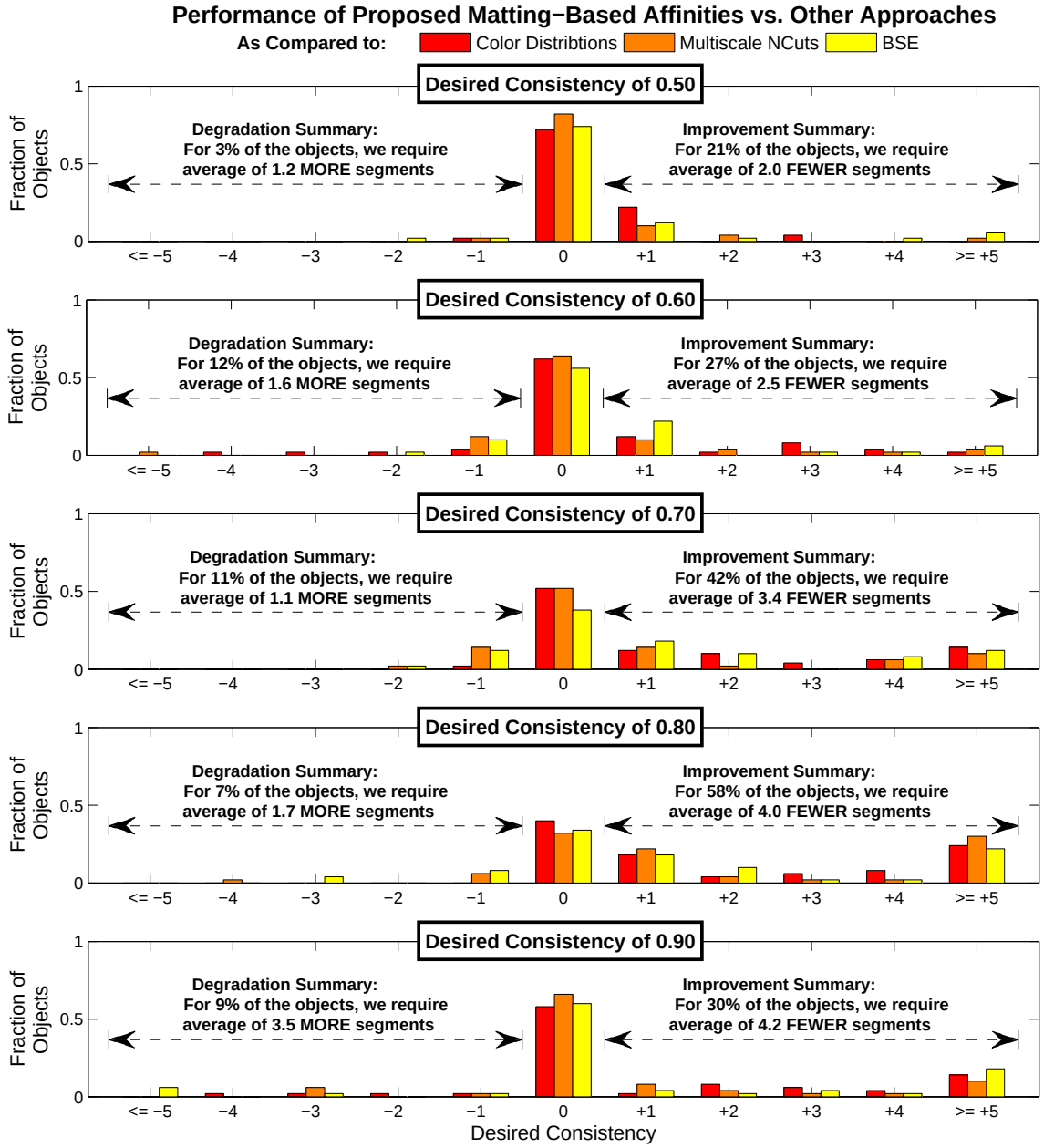


Figure 5.9: Overall Performance. Histograms of the *relative* number of segments required by our approach as compared to the other methods. The height of the bars corresponds to the fraction of the total number of objects for which we achieve the specified relative efficiency on the x-axis. Thus, bars at zero, in the center of the graph, correspond to cases when we perform just as well as the other approaches. Bars to the right (left) correspond to cases where we perform better (*resp.*, worse), using fewer (*resp.*, more) segments than the competition. As indicated, each plot corresponds to a different desired consistency level (increasing down the rows). Bars at the extreme left and right also include complete failure cases.

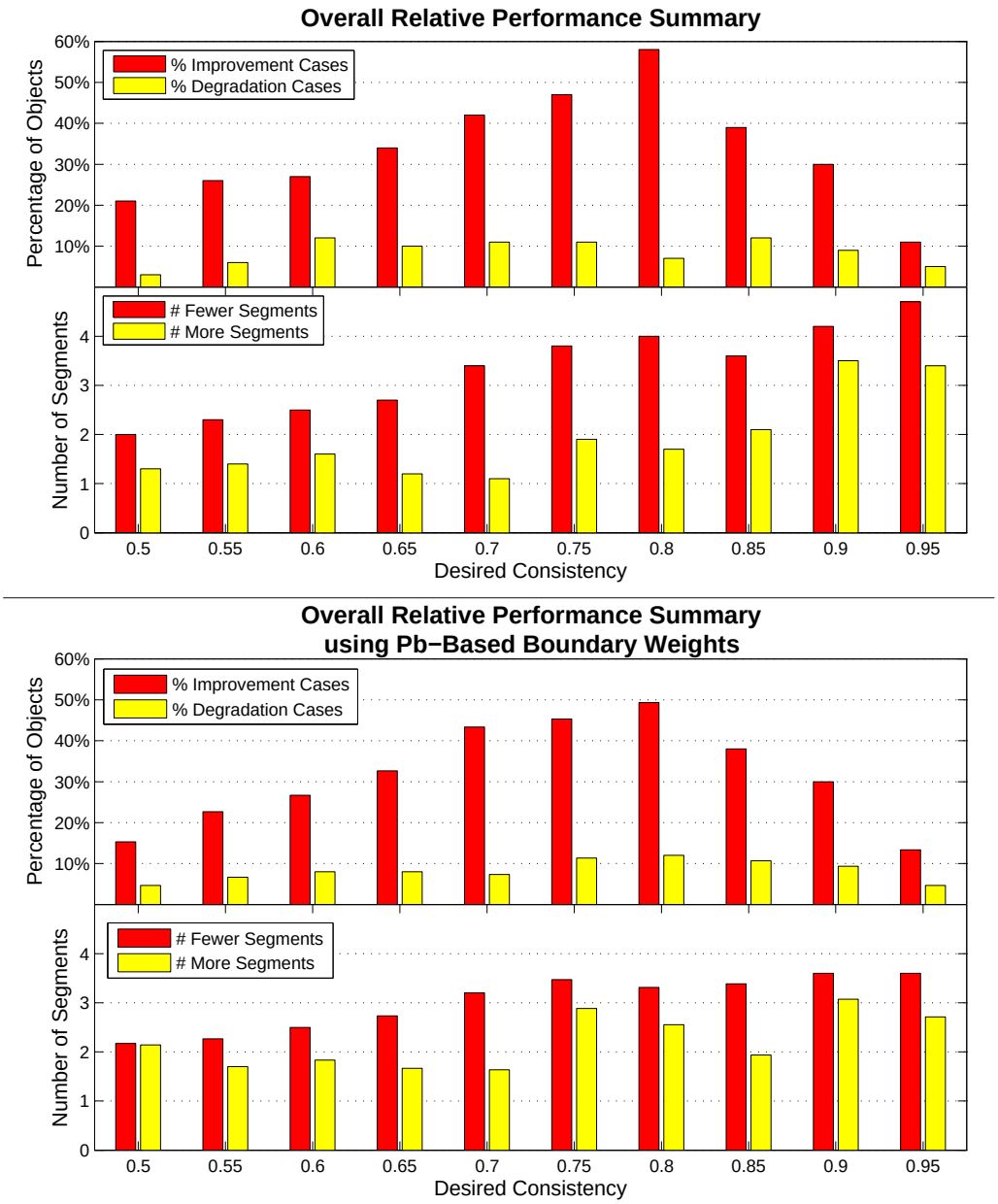


Figure 5.10: Top Half: Summary statistics for the relative performance of our matting-based approach as compared to the other methods, corresponding to the results in Figure 5.9. For the desired consistency levels along the x-axis, the upper graph compares the percentage of objects for which our method offers improvement or degradation. Note that the percentages do not sum to one since cases where there is no change in performance are not counted. At the corresponding consistencies, the lower graph indicates how much better or worse our performance is, in terms of number of segments required. For both graphs, we want the red bars to be as high as possible and the yellow bars to be as low as possible. **Bottom Half:** The same summary statistics, but using the average Pb strength along each fragment to provide the weights W in (5.2) — instead of our boundary probabilities from Chapter 4.

Chapter 6

Boundary Reasoning for Object Recognition

A popular approach to object recognition is to match objects from a database efficiently to those seen in novel images using a sparse set of information-rich *features* extracted from images [15, 19, 28, 31, 49, 87, 105, 106, 120, 121, 124]. Because the features only require local support, matching can be successful even in cluttered scenes. And when the features are specifically designed to be discriminative, only small numbers of matches are necessary to provide high confidence in an object's detection. These methods have also been motivated from a biological perspective [106, 176].

There are two broad steps involved in any feature-based recognition approach. First, features — also referred to as *keypoints* or *interest points* — are detected within an image. For this step, repeatability of detection is crucial [118]. If the detected locations (or scales/orientations) of interest points on an object vary from image to image, there is no hope of successful matching later. Second, descriptive signatures — or *descriptors* — are computed from local image values for each detected interest point in order to distinguish between them. The goal is to design a highly distinctive descriptor for each interest point in order to facilitate meaningful matches, while simultaneously ensuring that a given interest point will have the same descriptor regardless of the object's pose, the lighting in the environment, *etc*. So both steps, detection and description, rely on robustness to changes in various properties for success.

Much past work in this area has focused on producing interest points and descriptors that are invariant to scaling of the image [43, 83, 106, 118]. All of these approaches operate in scale-space to detect each interest point's characteristic scale [54, 55, 103]. Rotation invariance during *detection* is usually accomplished "for free" by using rotationally invariant image measures, such as the Laplacian. There are generally two approaches for creating rotationally invariant *descriptors*. Lowe [106] attaches a coordinate frame to each descriptor

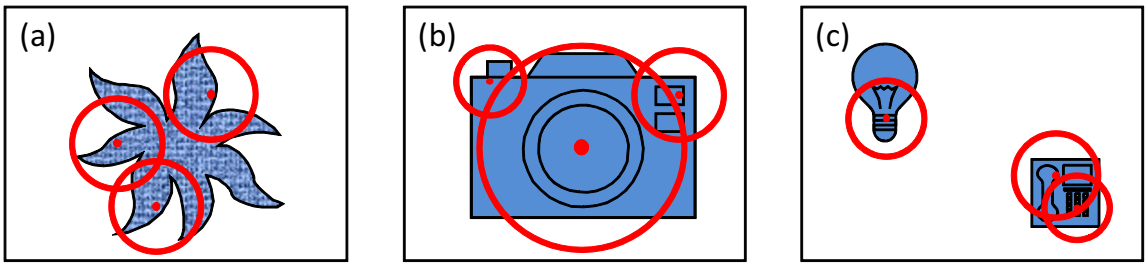


Figure 6.1: The need for background invariance in feature-based recognition methods arises particularly in cases of (a) objects with concavities, (b) objects with little texture and correspondingly few features which we therefore cannot afford to miss, and (c) objects appearing relatively small in an image, for which we must rely on larger features for which overlap with the background is more likely.

while others, *e.g.* [118], again use rotationally-invariant measures computed locally, such as local jets [54, 152]. The former produces more distinctive features, while the latter produces more invariant ones. It is important to note that this balance between the degree of invariance and the level of descriptiveness is a fundamental tradeoff.

Lowe's features also have some invariance to affine transformation engineered into them, but others have designed truly affine-invariant features. In [15], Baumberg describes a method for creating affine-invariant descriptors around scale-invariant interest point detections. Mikolajczyk and Schmid go a step further in [119], combining the detection and description phases into an iterative scheme such that *both* steps are invariant to affine transformation.

Unfortunately, because all of these methods rely on the use of local image information at various scales, both detection and description of features near object boundaries will incorporate information from both the object and the background, as shown in Figure 6.1. Therefore, when the object is seen with different backgrounds, its features would necessarily be different (both in location and description). In particular, this problem is exacerbated as the size of the object relative to the image decreases. As small features completely contained within the object become impossible to detect for lack of resolution, we must depend on larger-scale features, which are more likely to overlap with the background.

Ideally, the detection and description of an object's features would be the same regardless of the background on which it is seen. This implies that there is another type of invariance which would be desirable for feature-based methods: background invariance. To that end, we have developed a method for incorporating knowledge of (partial) object

boundary information, of the type discussed earlier in this thesis, into the detection and description processes of Lowe's popular SIFT feature [106]. Our approach, which we call the Background and Scale Invariant Feature Transform (BSIFT) [165], is the subject of this chapter.

It should be noted that the discussion thus far has centered around features derived from image intensity information directly. Recently, edge-based features have emerged which exhibit some degree of background invariance in order to recognize wiry shapes in cluttered scenes [31, 82, 121]. Such methods build local features from edge maps in order to capture shape rather than texture. It is likely that future feature-based systems will leverage both shape and texture information for successful recognition of a wide variety of objects. To our knowledge, the work desribed here is the first to address background invariance when using appearance/texture information.

6.1 Background Invariant Feature Detection

Our method builds directly on Lowe's Scale Invariant Feature Transform (SIFT) because of its popularity. Relevant portions of the SIFT method will be briefly reviewed here since our modifications occur at a fairly low level, but for complete details see [106].

Initial keypoints are detected as local extrema of a scale-space Laplacian pyramid. In practice, this Laplacian pyramid is efficiently approximated by the construction of a Difference-of-Gaussian pyramid instead. It is here, at the very beginning of the detection process, where we introduce our first modification.

Note that smoothing an image with a Gaussian filter blurs information across object-background boundaries. We can, however, replace this isotropic smoothing with more local process which respects arbitrary boundary conditions. This idea of using *anisotropic* smoothing has been extensively studied, *e.g.* [21, 133, 192]. We will only present here the basic concepts relevant to our specific goals. The key (classical) observation is that Gaussian smoothing is equivalent to performing iterative *local* heat diffusion by updating each pixel according to:

$$I^{(k+1)}(x,y) \leftarrow I^{(k)}(x,y) + \tau \nabla^2 I^{(k)}(x,y), \tag{6.1}$$

where

$$\nabla^2 I(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}. \tag{6.2}$$

If we diffuse at each pixel (x,y) in an image I for k iterations according to (6.1), then as the "timestep" $\tau \to 0$, we have equivalently smoothed the image by a Gaussian with $\sigma = \sqrt{2k\tau}$. To keep the diffusion process numerically stable, we use $\tau = 0.2$. The advantage of using a local diffusion process over standard Gaussian filtering (*i.e.* via convolution with a Gaussian kernel) is that we can naturally enforce arbitrary boundary conditions anywhere

in the image, namely at object boundaries. In practice, this is accomplished by using Neumann boundary conditions when computing the second derivatives necessary for (6.2), allowing us to switch the diffusion process on or off according to local boundary information. In contrast to most prior work using anisotropic smoothing (*e.g.*, for noise removal), we use boundary information from a *separate* reasoning process, as opposed to diffusing the image based simply on local intensity gradients. Otherwise, *every* edge visible in the image could suppress information flow across it, while we are only interested in preventing information flow across those edges that correspond to physical object boundaries. Figure 6.2 shows an example of smoothing the image of a Sony Aibo[1] by simple Gaussian filtering as compared to a diffusion process which incorporates boundary information. Note how diffusion degrades gracefully as holes are introduced in the boundary information. Thus we do not require perfect boundary detection (nor, equivalently, a perfect segmentation).

We can now build a boundary-respecting Gaussian pyramid by pausing our diffusion process at appropriate intervals to obtain each desired level of smoothing. As pointed out by Lowe [106], if we start with an initial smoothing of σ_0, the desired scale-normalized Laplacian pyramid can be closely approximated by a Difference-of-Gaussian pyramid with the degree of smoothing at each level L chosen according to $\sigma_L = 2^{L/s}\sigma_0$, where s is the number of samples per scale octave (typically 3 or 4).

Once the Difference-of-Gaussian pyramid is constructed, sub-pixel local extrema are identified in scale-space and low-contrast and edge-like features are filtered out, all following [106]. A synthetic example is shown in Figure 6.3, where the Aibo image has been pasted onto two differently-textured backgrounds. Interest points are plotted as X's with circles representing their corresponding scales. Only the detections whose locations and scales are the same regardless of background are plotted, with lines connecting them. In the center we see that when using regular SIFT, with no boundary information, only 38.3% of the detections with the brick background are also detected at the same location with the rocks background. On the right, where boundary information (here derived from the known silhouette of the Aibo) was incorporated into the detection process as described above, 98.4% of the detections are the same regardless of background. We emphasize that correspondence here is only in the sense of location and scale. No notion of descriptor matching has been incorporated into this example. The use of boundary information enables the detection of more features along the narrow legs of the Aibo. In addition, the boundary information has allowed for consistent detection of larger features, which are arguably more meaningful [176].

[1] *Aibo* is a registered trademark of Sony Corporation.

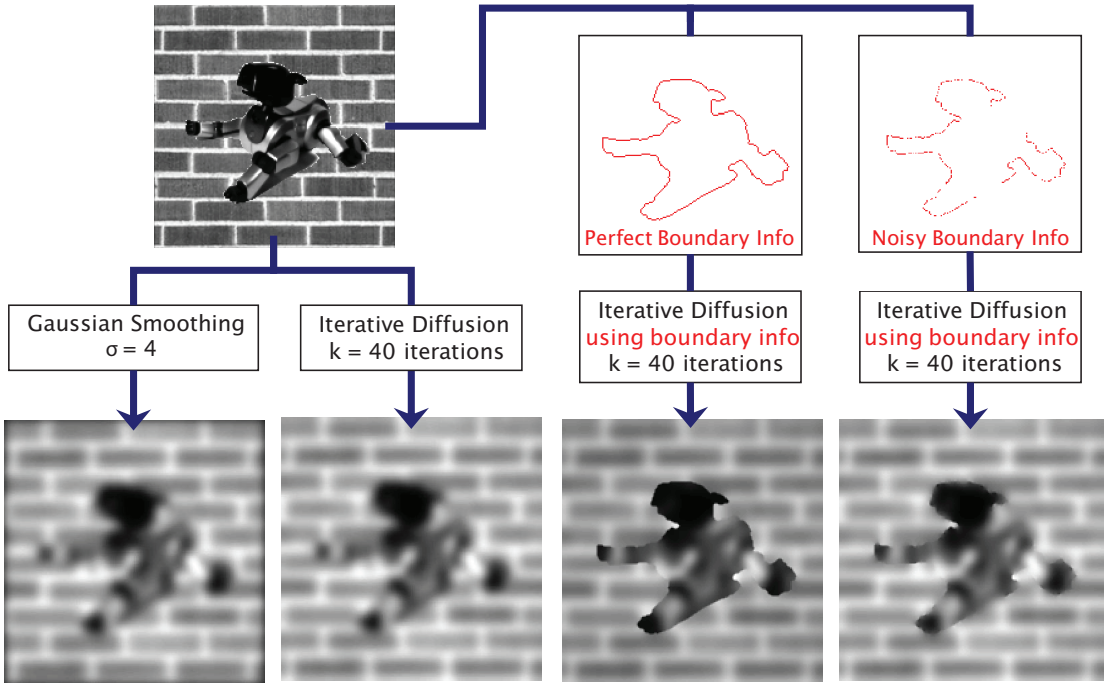


Figure 6.2: A comparison of Gaussian smoothing by simple filtering versus diffusion with boundary knowledge. With no boundary information diffusion and filtering produce identical results. With boundaries, however, diffusion prevents information from bleeding between the object and background, degrading gracefully as noisy holes are introduced.

6.2 Background Invariant Feature Description

Now that we have a method for detecting interest points in a background-invariant manner, we need to generate descriptors for those interest points which will also remain the same regardless of background. Again, our descriptors are direct modifications of Lowe's SIFT descriptors, so further details may be found in [106].

For each interest point, image gradient magnitudes and orientations are extracted within a patch whose size is determined by the scale of the interest point. The orientations are binned into a histogram weighted by their magnitudes and by distance from the patch center in order to determine a dominant orientation for each interest point. (Note that, in practice, multiple peaks in this orientation histogram may result in multiple interest points at the same location, but with differing orientations.) At this point, without any

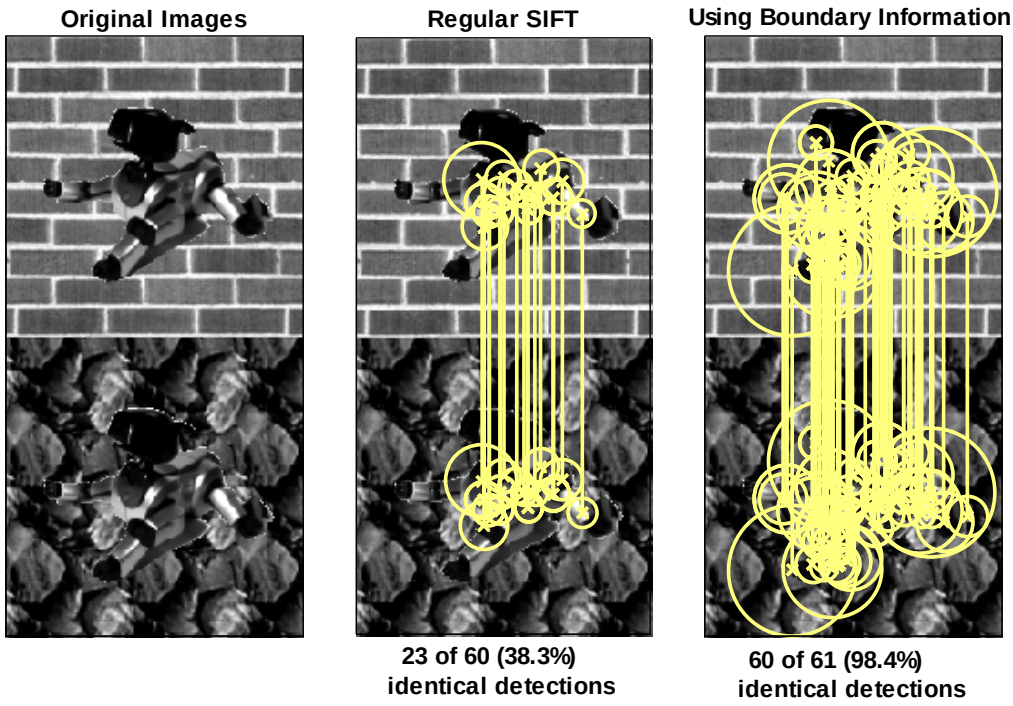


Figure 6.3: The incorporation of boundary information allows for more consistent feature detection on the same Sony Aibo object image pasted onto different background images.

modification, background invariance is violated since we are working with a patch of image values which will often overlap object and background pixels, particularly for detections near object boundaries. In order to impose background invariance on this step of the descriptor generation, we use a boundary-respecting weighting mask in the creation of the orientation histogram, rather than a simple Gaussian. This lessens the importance of samples further from the interest point while also effectively removing samples not on the object.

The creation of a weighting mask by heat diffusion as above might seem to be the natural

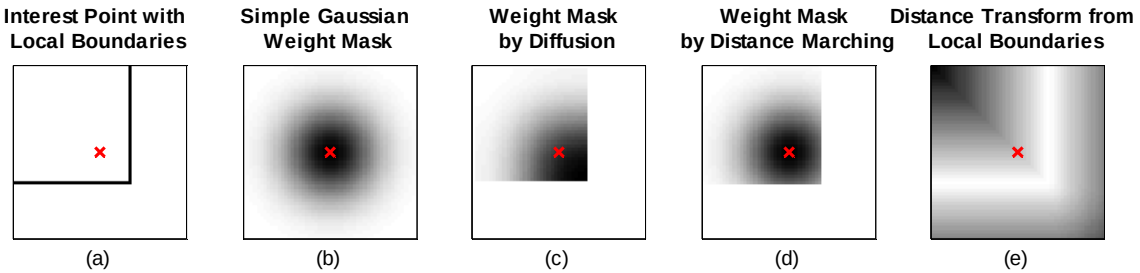


Figure 6.4: Weighting masks around an interest point for incorporating a nearby object boundary into descriptor creation.

choice here, given its use in detection above, but it turns out not to have the desired effect in this case. Figure 6.4 compares different weighting masks for an interest point marked by the red X with local boundary information as shown in (a). Without utilizing the local boundary information, a simple Gaussian weighting mask (b) as used by SIFT clearly gives non-zero weight to pixels which do not lie on the object, resulting in a non-background-invariant descriptor. But if we diffuse outward from the interest point according to (6.1) in order to produce a boundary-respecting Gaussian mask (c), we see that the weight values build up like a "snowdrift" at the boundaries, resulting in a weight mask which is no longer truly a function of distance from the interest point (note that the darkest part of the mask is not at the X as we would like). If we instead use a distance marching procedure [154] to propagate distances away from the interest point while respecting boundary information, however, we obtain the correct weight mask (d). Such a mask gives zero weight to pixels not on the object and weight proportional to distance from the interest point for pixels on the object. In addition, this procedure is more efficient than repeated diffusion at every interest point.

One might question the necessity of this method of weight mask generation. Why not just truncate the simple Gaussian weight mask in Figure 6.4 (b) with the boundaries from (a)? The problem with that straightforward approach becomes apparent when perfect boundary information is no longer available. Certainly we do not want our proposed method to be dependent on an unrealistic assumption that perfect boundaries are provided. But just as the iterative diffusion process used for feature detection degraded gracefully as holes in the boundary were introduced (see Figure 6.2), so too does our method of weight mask generation by distance marching. In Figure 6.5, we have introduced holes into the local boundaries around the interest point marked by the red X in (a). But we see that the resulting weight mask in (b) will still put much less emphasis on pixels on the opposite side of the boundary; only minor leakage occurs through the holes. It is not clear how to achieve this effect by starting with a full Gaussian weighting function and then attempting to "truncate" it using such imperfect boundary information.

Finally, we also found that adding a weighting mask based on distance from the nearest boundary increased performance. The idea here is to lessen the contribution of pixels which lie close to object boundaries, whose precise locations may not by known in practice. This weighting function can be computed using a distance transform from local boundaries and is point-wise multiplied by the Gaussian weighting mask described above. An example with perfect boundaries is given in Figure 6.4 (e) and with imperfect boundaries in Figure 6.5 (c) — here again we see that this weighting function is largely unaffected by the holes in the boundary. Thus the final weighting mask becomes a balance between increasing weight as we approach the interest point and decreasing weight as we approach a nearby boundary. An example of a combined weight mask is show in Figure 6.5 (d).

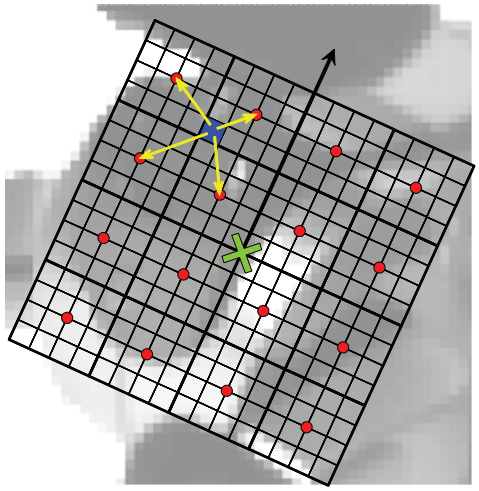


Figure 6.6: Descriptor creation: A set of sixteen histograms of gradient orientation (marked by red dots) are aligned to the dominant orientation of an interest point (green X). Each sample (*e.g.* the blue square) contributes its orientation to the four nearest histograms (yellow arrows) according to its magnitude, distance to each of those histograms, and distance to the interest point itself.

Once a dominant orientation is assigned, a coordinate system is aligned to that direction. The gradient orientations at each site within the local patch are placed into sixteen histograms, each with eight bins, positioned relative to the aligned coordinate system. Each orientation sample's contribution to a given histogram is weighted by its magnitude, distance from that histogram's position in the coordinate frame, and distance from the interest point itself. This process is illustrated in Figure 6.6. For the interest point marked by the green X at the center, a coordinate frame has been aligned to the dominant gradient orientation as found by the procedure outlined above. Each square in the grid represents a sample site for the patch surrounding the interest point, where a sample consists of the gradient orientation and magnitude for that location in the image. The sixteen histogram centers are designated by the red dots. The blue shaded sample is shown to contribute to its four nearest histograms with a weight which is bilinearly interpolated based on distances to those histograms' centers.

To add background invariance to this standard SIFT descriptor, a boundary-respecting weighting mask produced by fast marching is again used to weight each sample's histogram contribution according to its distance from the interest point. The distance-from-nearest-

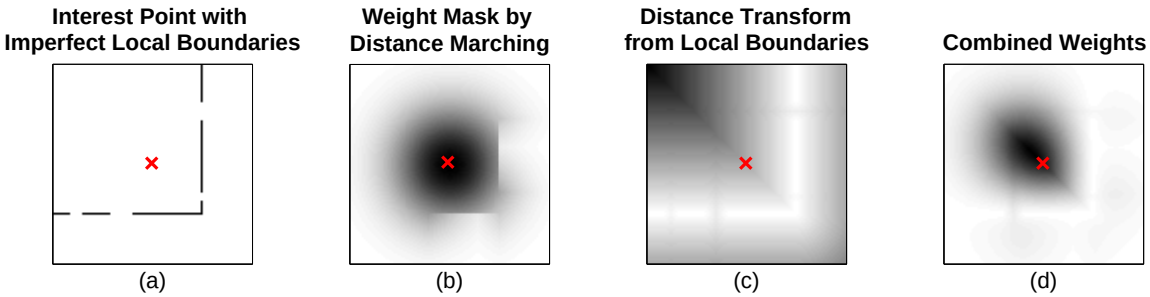


Figure 6.5: Descriptor weighting masks in the presence of imperfect boundary information, showing graceful degradation of the masks despite holes in the nearby boundary.

boundary weight as described above is used here as well. Finally, all sixteen histograms' bin values are concatenated into a 128-vector, which is then normalized into a unit vector. Following [106], this vector is further adjusted by clipping all values at 0.2 and then renormalizing. The resulting vector forms the 128-dimensional descriptor for an interest point.

In Figure 6.7, we have manually chosen the same interest point for the Sony Aibo object, but with different backgrounds. Note that the large scale of the interest point and the shape of the object cause significant inclusion of background pixels. In comparing the SIFT descriptors computed with the brick background (upper bars) and the rocks background (lower bars), we see that the two are very different. There is no hope of matching them as the same descriptor. But when boundary information is incorporated into the weight masks as described, we see that our resulting descriptors, are exactly the same (*i.e.* they have a Euclidean match distance of zero in descriptor space).

We now have all the pieces necessary for what we refer to as a Background and Scale Invariant Feature Transform (BSIFT). In the following sections, we will compare the performance of our BSIFT method to standard SIFT.

6.3 Synthetic Experiments

In order to evaluate our method's performance, we need to know ground truth boundaries for a library of objects. To facilitate large-scale evaluation with a database of objects and a non-trivial number of example images, we artificially paste object images from a database of 110 objects onto a set of 25 background images. The set of backgrounds consists of about half indoor office/lab scenes and half outdoor natural scenes. The database of objects and backgrounds is a concatenation of some of our own images with many others from various online databases [60, 97, 125]. Each object has been hand segmented into foreground and background pixels, so precise object boundaries are known. Furthermore, features have been extracted from the objects with and without using boundary information (*i.e.* using BSIFT and SIFT, respectively), and all of these features are stored in one large database consisting of over 8000 features for each method.

In order to guard against implementation differences and ensure the only variable being tested is the inclusion or lack of boundary information, we do not use Lowe's publicly-available SIFT library to test against our BSIFT implementation. Rather, we use only our own code, supplying boundary information to test BSIFT or withholding boundary information to test SIFT. (Without any boundary information, our method defaults to using standard Gaussian smoothing and Gaussian weighting masks.) We therefore expect that our baseline performance will differ from the more mature SIFT library available from Lowe, though we have verified that our implementation performs comparably with Lowe's.

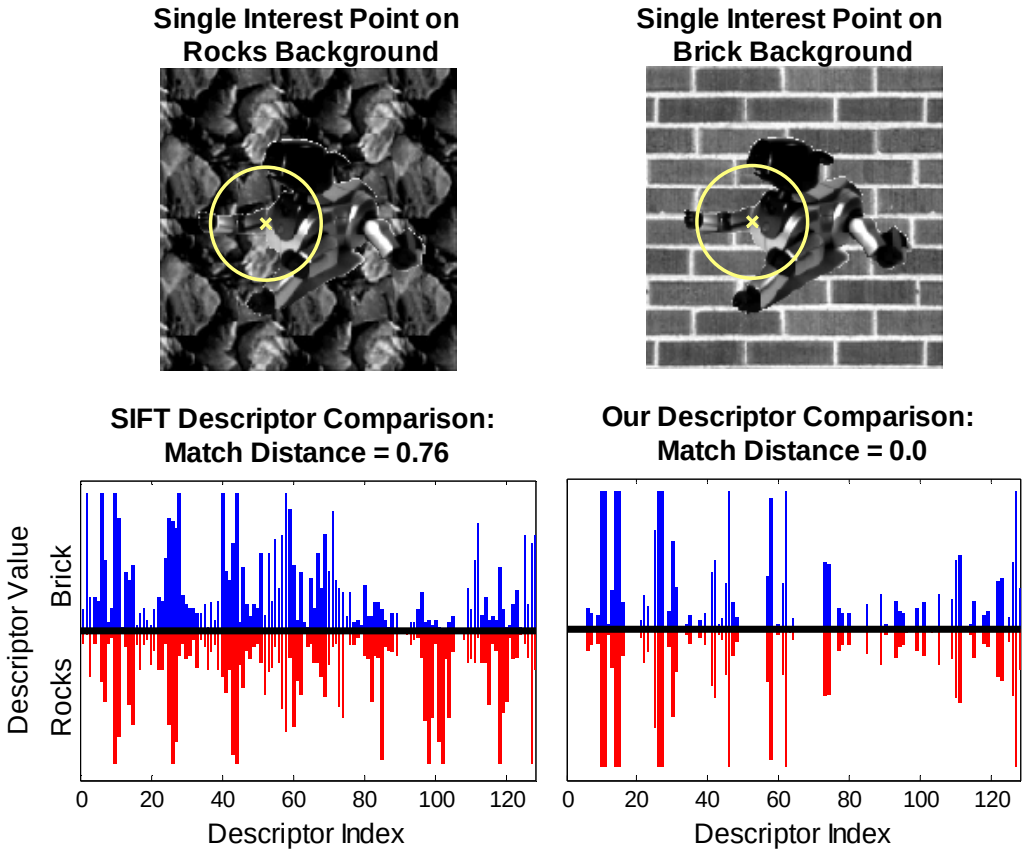


Figure 6.7: Incorporating boundary information into the descriptor creation allows us to compute exactly the same descriptor for an interest point, despite its overlap with two different backgrounds. SIFT, on the other hand, produces drastically different descriptors.

Finally, it is worth noting that the database includes many low-texture objects as well as objects with narrow appendages. Both are difficult for existing feature-based methods. A random selection of some of the objects in our database is shown in Figure 6.8.

For each test, a random background and object pair is chosen and the object is pasted onto the scene at a random location, orientation, and scaling (60-100%). In addition, we can simulate to some degree the effect of errors in object boundary extraction by introducing random contiguous breaks in the boundaries. This degradation is quantified by the percentage of the original boundary that is eroded away. Our performance criterion is the percentage of interest points present for the object in the database that are correctly matched to the object in the generated image — where a correct match implies that both the object's identification and location are correct. For reliable results, [106] suggests defining a match to be found when the ratio between the closest and second-closest descriptors in the database, using simple Euclidean distance, is less than a threshold (we use 0.6 in for all the experiments described here). The interesting cases are those when our method finds

Figure 6.8: Example objects from the database used in our synthetic recognition experiments. The objects used are a combination of objects from our lab and images from various online object databases. Note the inclusion of low-texture objects as well as objects with narrow appendages. Both are difficult for existing feature-based methods, which rely on image patches.

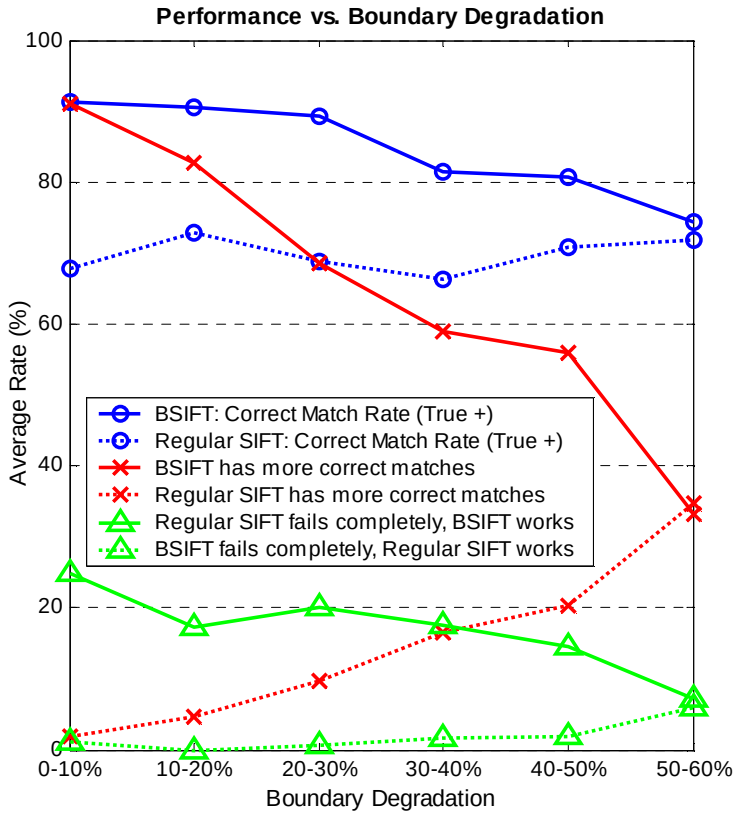


Figure 6.9: As expected, as boundary degradation increases, BSIFT's performance resembles that of regular SIFT more and more.

substantially more matches than SIFT, or better still, when SIFT fails to find any matches and our method succeeds.

For a set of 3000 experiments, we found that the average correct match (true positive) rate using our method was 80% while SIFT's was only 55%. Our method had a strictly greater number of correct matches (meaning more confident object recognition) 68% of the time. In fact, our method found at least one match while SIFT failed completely in 26% of the experiments. In 9% of the experiments SIFT found more matches than our method, and in 15% neither method found any matches (thus demonstrating the existence of some rather difficult objects in our database).

As expected, our method's performance and its superiority to regular SIFT decrease as boundary degradation increases since information derived from object boundary locations is the only difference between the two methods. Figure 6.9 shows a plot of performance values versus the amount of boundary degradation for a set of 1000 experiments in which scaling was set at 100% and boundary degradation varied between 0 and 60%. BSIFT's performance falls off gracefully rather than completely failing as boundaries are eroded

away and information leaks through during the diffusion and distance marching processes. Finally, in Figure 6.10, we also see that BSIFT is more resistant to the effects of scaling. For a set of 1000 experiments in which scaling varied uniformly between 60 and 100% (while edge degradation was disabled), we see that the matching rate for BSIFT does not fall off as significantly as for SIFT. For BSIFT the matching rate drops about 10% as the scaling decreases, whereas SIFT's performance — which is consistently much lower than our method's — drops more than 20%.

6.4 Real Experiments

Having verified in a controlled manner that the use of boundary information as described can improve recognition performance, we now turn to several proof-of-concept examples on real images. Noting that BSIFT is not strictly tied to the boundary detection methods presented in previous chapters, we present results using three different sources of boundary information: background subtraction, stereo disparity, and our approach from Chapter 4.

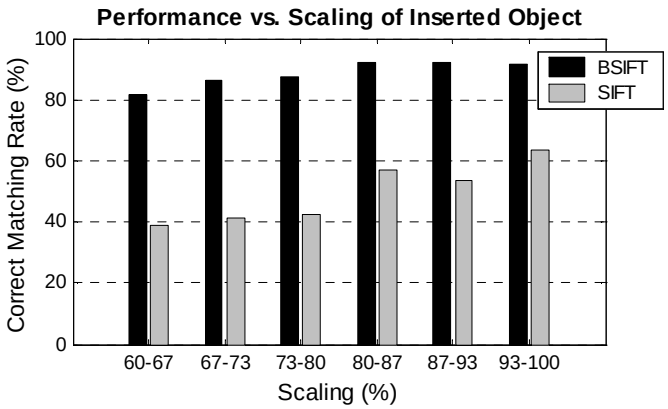


Figure 6.10: As scaling decreases, SIFT's performance remains below that of BSIFT and falls off more quickly. BSIFT achieves over 80% correct matching performance over the scaling range.

In the top row of Figure 6.11, a toy car has been placed on a desktop such that it is quite small with respect to the image size. Thus, there is not enough resolution to find many features on the object. We see that SIFT finds two feature matches while BSIFT finds six. In the bottom row, a set of dry-erase markers — which exhibit very little texture — have been placed in an office scene, again such that they are small with respect to the image size. Here, SIFT fails to find any matches, while our method finds four. Note that there are only nine and eleven *total* features detected for the markers' *training* image using SIFT and BSIFT, respectively.

Figure 6.12 shows results using boundary information derived from a stereo disparity map. In this case the test image has limited resolution as compared to the training image (which is not uncommon in practice). The smaller features normally available on the fan, which regular SIFT could correctly match, are of little use in this case because of the lack of

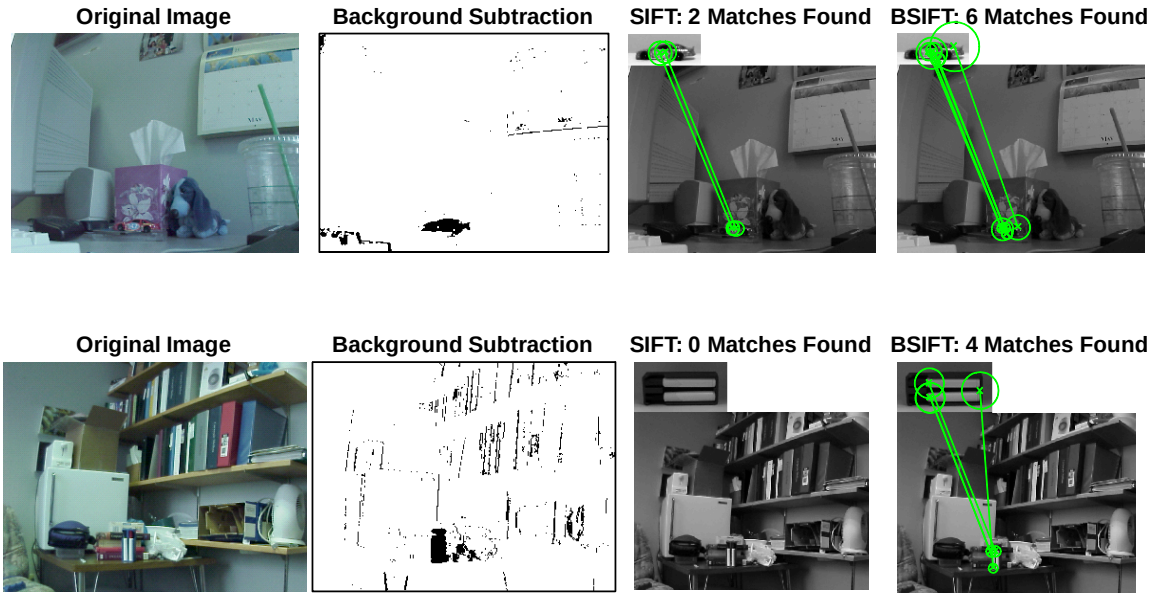


Figure 6.11: BSIFT outperforming SIFT using background subtraction for a toy car object (top row) and a set of dry-erase markers (bottom row). Note that only three matches for the dry-erase markers are visible because two are at the same location but with different orientations.

resolution. The texture of the couch in the background contaminates any large-scale SIFT features, but BSIFT is able to correctly match them. Indeed, SIFT fails to find a single match, while BSIFT finds four.

Finally, in Figure 6.13, we provide an example linking the BSIFT approach described in this chapter with the boundary detection process presented in Chapter 4. As with the previous example using stereo, the training image of the object here is much higher resolution than the scene image. Using regular SIFT, two correct matches are found and one incorrect match is also returned. Using BSIFT, with boundaries detected by our approach, however, we find seven correct matches.

6.5 Discussion

In this chapter we described a straightforward, intuitive method for incorporating boundary knowledge into the detection and description processes of feature-based object recognition. We demonstrated on controlled, synthetic experiments the desired improvement and graceful degradation offered by our approach. Finally, we presented initial results linking a few methods of boundary detection, including our own, to the BSIFT approach described here. Large-scale experiments integrating these approaches more completely into a single system would be an interesting line of future work.

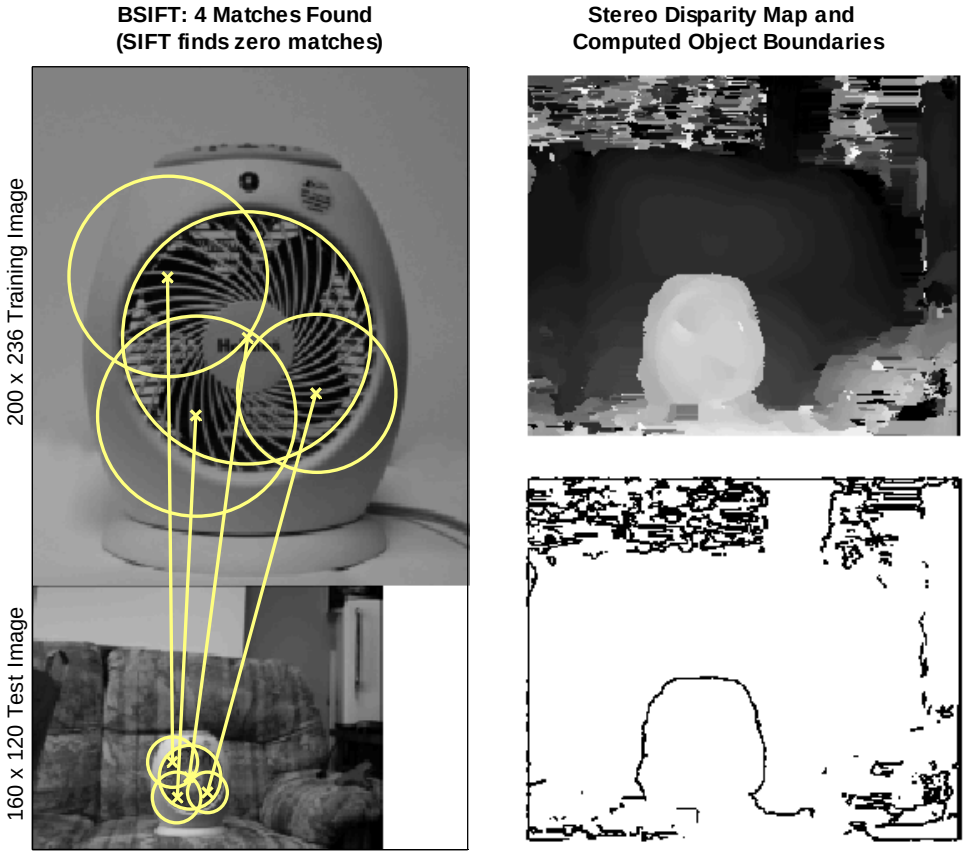


Figure 6.12: Using object boundary information derived from a stereo disparity map, BSIFT is able to find matches on the fan despite the low resolution/scale. SIFT fails completely on this example. Note that the training (top) and test image (bottom) are shown at actual relative scale.

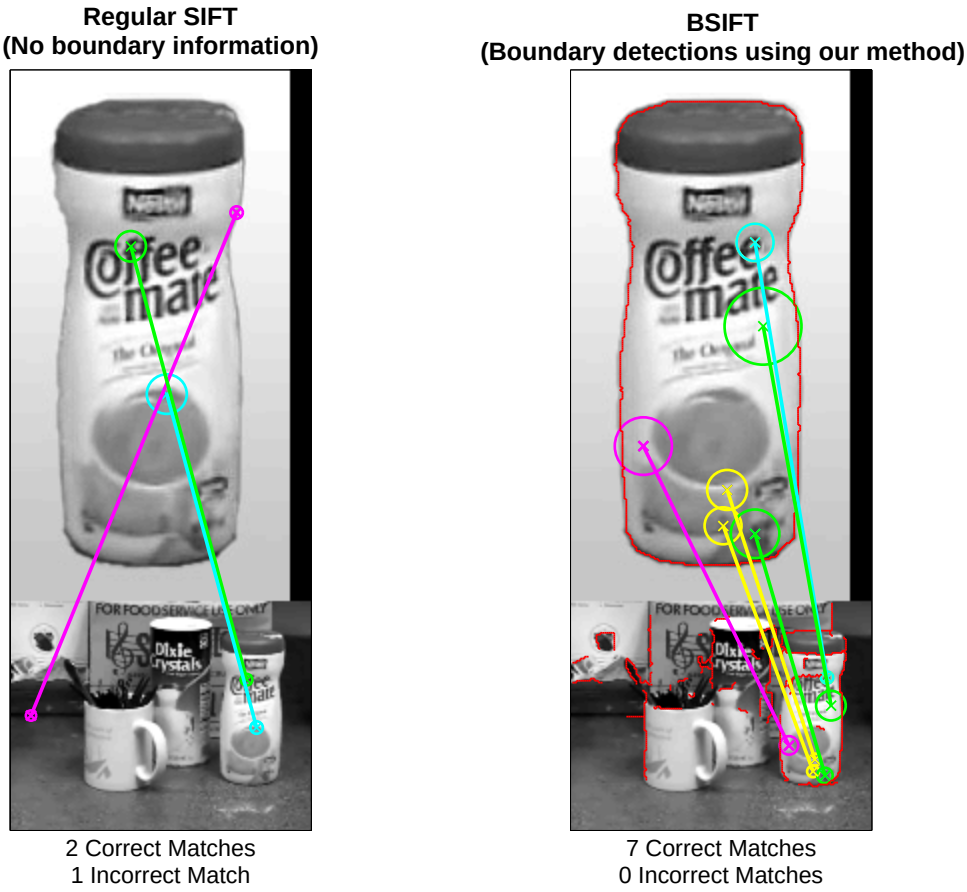


Figure 6.13: Using BSIFT with boundaries detected using the methods presented in this thesis, we find seven correct matches (right), as compared to two correct matches and one incorrect match found using regular SIFT (left). The boundary information used by BSIFT in the right example is indicated in red. Note that the training (top) and test image (bottom) are shown at actual relative scale.

Chapter 7

Conclusion and Future Directions

Occlusion is a pervasive, often-overlooked visual phenomenon which is a significant stumbling block for most, if not all, computer vision pursuits. But occlusion boundaries are also an important part of perceptual reasoning. To ignore or avoid them simply as noisy outliers throws away valuable information useful in higher level processing. In this thesis, we have described the various motion cues available at occlusion boundaries and presented methods for estimating them from short video clips, including an extension of powerful patch-based edge detection techniques to the spatio-temporal domain. In combination with appearance information, we have used those motion cues to demonstrate local, pixelwise boundary detection. By then incorporating improved spatial support and a novel graphical model, we have also suggested an approach for more global reasoning about boundaries.

Furthermore, we demonstrated methods for employing boundary information (from our described detection strategy or others) into higher-level processing. First, we used the boundaries to suggest segmentation "hints" via state-of-the-art matting techniques and combined a set of such hints into a pairwise affinity measure suitable for whole-object segmentation. In addition, we used boundary information to add a form of background invariance to feature-based recognition methods.

The dataset of video clips used throughout this work has also been made available publicly for use by other researches.

7.1 Improvements and Extensions

In this work, we have addressed occlusion boundaries from a wide range of perspectives including low-level feature extraction and local detection, mid-level reasoning and propagation, and high-level reasoning in the form of segmentation and recognition. Certainly, further research is possible for digging deeper at any of these levels. Below we will discuss potential enhancements and directions building on our work.

7.1.1 Low-Level Improvements

Complex Spatio-Temporal Edges

The use of a plane to divide our spatio-temporal patches for boundary speed estimation restricts us to detecting locally linear edges undergoing locally constant-velocity motion. Using those planes as an initialization, one could also adapt the shape of the dividing surface in space, time, or both to account for more complex edge shapes undergoing more complex edge motion. This could potentially be performed via energy minimization in an active-contour (snake) framework or using level sets [154]. Though this *may* lessen the need for stabilization (*i.e.* dominant motion removal) to some extent, it would likely be quite computationally expensive.

Improved Motion Estimation

Our multi-frame approach to patch motion estimation — see Equation (2.8) — currently incorporates only *spatial* derivatives I_x and I_y of the *reference* frame. Since, by construction, the location of the patch in the reference frame is constant as we iteratively update the motion estimate, the matrix G of spatial derivatives can potentially be pre-computed. This is computationally convenient, but it may be worthwhile to explore other motion estimation procedures which make use of the spatial derivatives across all frames [11, 53], or related techniques such as "congealing" [96]. In particular, heavy reliance on the reference frame can be problematic when motion blurs the data in that specific frame. In such cases, it is less likely that the necessary gradient information will be blurred out for the entire duration of the spatio-temporal patch, so utilizing all available spatial gradients could be helpful.

Super-pixel Matting

The main computational bottleneck of our whole-object segmentation strategy in Chapter 5 is currently in producing the mattes associated with each fragment using Levin's method [100]. Unfortunately, the mattes obtained with the much faster approach presented in [9] were far less useful. While for our experiments we achieved significant speed-up of Levin's publicly-available implementation by improving "vectorization" of the Matlab code, it should be much more advantageous to convert that pixelwise implementation to operate on super-pixels instead. This appears to be possible, though not completely trivial since the "propagation" effects offered by the use of overlapping 3×3 windows (as claimed in [100]) would need to be extended carefully to the super-pixel domain.

7.1.2 Mid-Level Enhancements

Using Regions

As discussed in the introduction, we have focused mainly on reasoning *at* boundaries themselves. The regions enclosed by those boundaries could be further integrated into

a system capable of utilizing both the techniques presented here and existing region-based methods. For example, it may be beneficial in some situations to incorporate other, region-based measures of affinity into the segmentation strategy described in Chapter 5.

Feature Improvement via Feedback

Continuing in this vein, one avenue of research here might be the feedback of our detected boundaries into the initial boundary hypothesis procedure. Confident boundary estimates could be used to merge initial segments, thereby producing longer fragments and larger regions of support for features [76]. Ideally, iterating the processes of segment/fragment hypothesis, feature extraction, propagation, and classification could potentially lead directly to further improvements in object segmentation — still without high-level, object-specific knowledge.

While this may be reasonably straightforward for many appearance/geometry features, care would need to be taken in updating fragments' and segments' *motion* estimates. We are able to rely on fairly simple translational models of motion in this work because translation is *locally* able to capture perceived motion quite well. As segments expand and fragments extend, such a model may not be appropriate. But venturing straight down the road of applying more complex motion models may be dangerous and could suffer the same pitfalls as many layered motion segmentation techniques.

In the extreme, non-rigid (or articulated) objects such as people, animals, or cloth may not adhere to any reasonably concise motion model. In other words, it may be quite difficult to find or estimate a single model which explains the motion of all the constituent pixels on such an object. For segmentation of this type of object, it may be more useful to rely largely on grouping of the boundaries (whose foreground/background motions may still be *locally* translational) and simply employ some form of (piecewise) smoothness on the enclosed segments' motion estimates.

Figure/Ground Analysis

We have not performed specific analysis and evaluation of figure/ground assignment to our boundaries in this work. While a notion of the foreground side of a boundary was an important part of label propagation in our factor graph model in Chapter 4, our focus has mainly been on the *detection* of boundaries and their utility for high-level tasks. Indeed, it is interesting to note that explicit figure/ground labels were not absolutely necessary to yield the results presented in Chapters 5 and 6 for object segmentation and recognition. Still, further research in this direction is warranted. In our experience, our method generally achieves qualitatively *consistent* foreground assignment along boundaries, but sometimes the entire boundary's foreground/background labeling is reversed. We believe this to be partly an artifact of impoverished local cues combined with the limited ability of existing

inference techniques to propagate local information effectively over large distances on dense, loopy graphs. Further incorporation of ideas from Ren's work [139, 140], which focuses on the figure-ground assignment problem *given* the object boundaries, may prove beneficial here.

Integrated Training

The classifiers needed to compute the potentials for our factor graph model in Chapter 4 are currently trained independently (from each other and from the inference procedure itself). It could be beneficial — though also quite expensive — to train them together and within an inference loop. This is akin to the training of parameters in Discriminative Random Fields [92] and could help avoid the over-confident fragment classifications and the over-eager closure of small boundary loops we have sometimes observed in our experiments. In addition, other machine learning techniques, such as bagging [25], could be employed.

Boundaries through Time

All of our work has sought to discover the occlusion boundaries present in a single reference frame of a short video clip. A next step could be the application of our techniques on a per-frame basis (or every n^{th} frame), with additional temporal reasoning/filtering, in order to utilize more fully the temporal aspect of video data. One difficulty in this domain may be the labeling of several (or all) frames of video for quantitative analysis or training.

7.1.3 High-Level Applications

Object Discovery

Our object segmentation results are encouraging in that they tend to delineate whole objects in small numbers of segments. A next step given such segmentations is the actual automatic discovery or "pop-out" of those objects from a scene or set of scenes, as discussed in [146]. Also interesting could be the incorporation of other cues or affinity measures, alongside our boundary-based "hints", into the segmentation procedure.

Shape-Based Recognition and Reasoning

We focused here on incorporating boundary knowledge into appearance-based recognition, but good boundary estimates should also be helpful for *shape-based* recognition [31, 50, 82, 121, 129, 155]. Indeed, the boundaries we seek are precisely those which correspond to an object's silhouette, which is nearly analogous to its 2D shape and can be used in 3D shape reasoning as well [95, 150, 153, 178].

7.2 Summary

To summarize our work in this thesis, we emphasize that though occlusion boundaries may at first glance be problematic for many tasks in computer vision, they are in fact themselves important sources of information and should not simply be avoided or treated as outliers. The relative motion cues observed across those boundaries, though quite subtle, can be carefully estimated and combined with typical appearance cues to help differentiate them from appearance-only edges which may lack structural meaning. Furthermore, their utility becomes apparent when boundary-based reasoning is carefully incorporated into segmentation and recognition tasks. Finally, we encourage for any task the explicit combination of a variety of information sources and cues, the use of data-driven spatial support and decision-making methods, the incorporation of uncertainty where available, and the delay of hard, irreversible decisions for as long as possible.

Appendix A

Crack Chaining

In this appendix we present in more detail the crack chaining approach used in extracting potential boundary fragments from an over-segmentation, as discussed in Chapter 4.

Consider the example segmentation of a simple image consisting of 4×4 pixels at the left of Figure A.1. There are four segments, each indicated by a different color, with thicker lines at their borders which lie along the cracks of the image's pixels. In the center of the figure, the four-way intersections of the pixels, where four cracks also meet, are labeled with circles. At the right, the corresponding "crack graph" is shown, whose vertexes and edges correspond to the pixel intersections and the cracks, respectively.

Each crack in this graph can take on a binary label indicating whether or not it corresponds to a segment border in the original segmentation, again indicated by thicker lines. Furthermore, each vertex $(a) - (i)$ can take on one of exactly twelve possible labels, depending on the labeling of its four cracks. These labels, representing the presence of local junctions, (non-)boundaries, and corners, are shown in Figure A.2. Note that the four cases corresponding to a single crack being labeled as a segment border are not possible

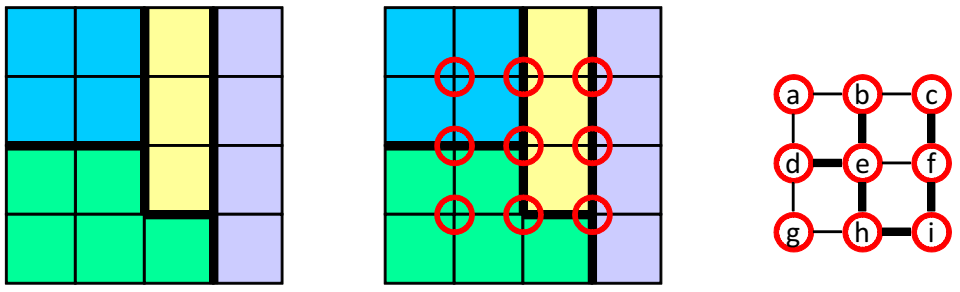


Figure A.1: The cracks between the pixels in a segmentation form a graph we can use for chaining.

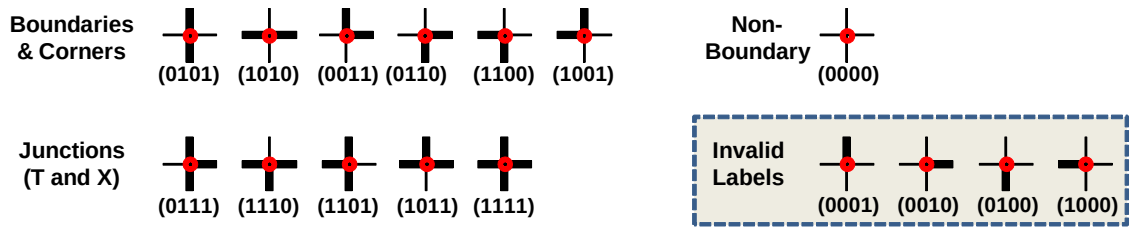


Figure A.2: The possible labels for each type of pixel intersection, based on the binary labelings of their four associated cracks. Some labels are invalid in our approach, which seeks closed boundaries.

when working with the closed borders of an over-segmentation. (Though not indicated in Figure A.1, we consider the borders of the image to be segment borders in practice.)

If we associate each crack at a particular vertex with a bit in that vertex's four-bit label, as indicated in Figure A.3 and below each intersection shown in Figure A.2, we can efficiently chain together intersections of the graph by simple logical bit checks. For example, we know vertex d in the graph from Figure A.1 is chained to vertex e by simply checking that the second bit of d's label and the fourth bit of e's label are both set. Vertex d is *not* connected to g, however, since neither d's third bit nor g's first bit are set.

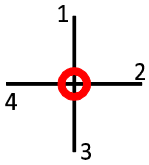


Figure A.3: Bits corresponding to each crack at an intersection.

Based on this reasoning, we start at those intersections labeled as junctions (those with three or more bits set) and chain along the crack graph until reaching another junction. Note that it is also necessary to consider the special case of a segment completely enclosed by another, larger segment. In this case, a single fragment encircles the whole inner segment, and does not begin or end at a junction.

The cracks linked together in this way form the graph of "fragments" used in our work. Note that there are no thresholds (*e.g.* on edge orientation variation, edge strength, *etc.*) required and that the resulting fragments will naturally form closed graph structures since they are based on an underlying over-segmentation of the image.

Appendix B

Additional Boundary Detection Results

We include here additional examples from our dataset for boundary detection via the global reasoning described in Chaper 4. For each example below, we provide the middle (reference) frame from the video clip, the precision versus recall curve, and boundary detections at various recall operating points (which are indicated on the precision vs. recall plots by vertical dashed lines). We show thresholded boundary detections after global inference when using appearance information alone (left column) and when appearance and motion information are combined (right column). We have chosen operating points for each example to highlight the performance increase when using motion information (*i.e.* when using motion, the number of false positives is generally lower – and thus the precision is higher – for a given recall point, as compared to using appearance information alone). Note that the selected recall point is always the *same* between the columns of a given example, to facilitate fair comparison.

The last two figures (B.10 and B.11), show difficult examples. In these cases, very harsh lighting conditions create high contrast shadow edges that confuse our system. In addition, large un-textured regions combined with reflections and specularities visible on the car make motion estimation difficult.

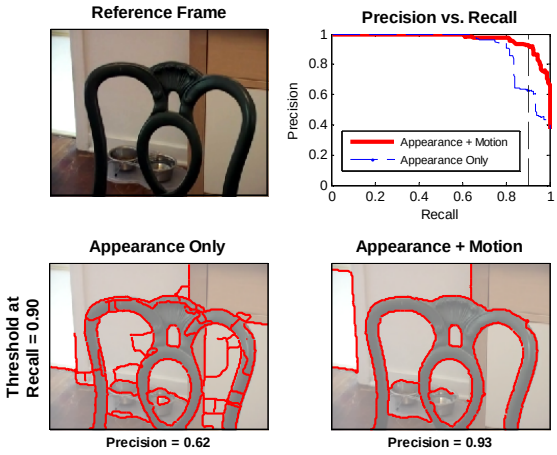


Figure B.1: Chair

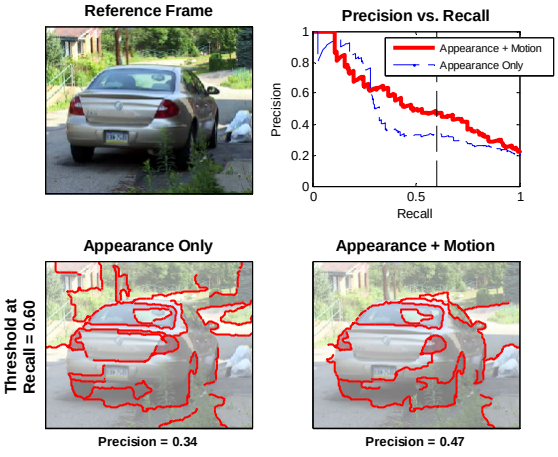


Figure B.2: Car

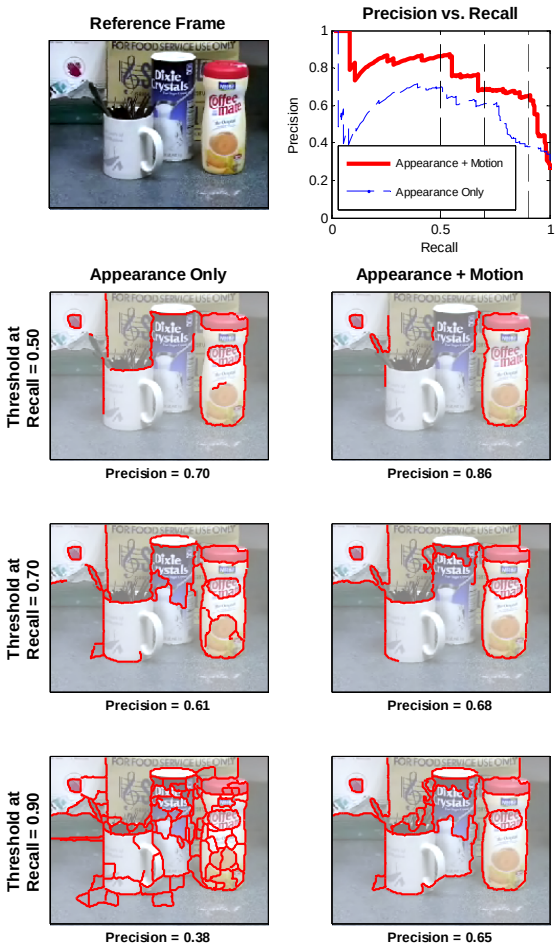


Figure B.3: Coffee Stuff

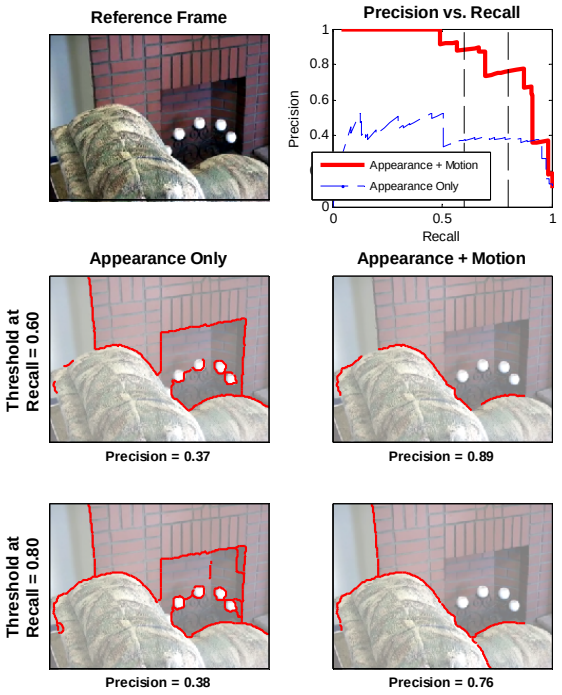


Figure B.4: Couch Corner

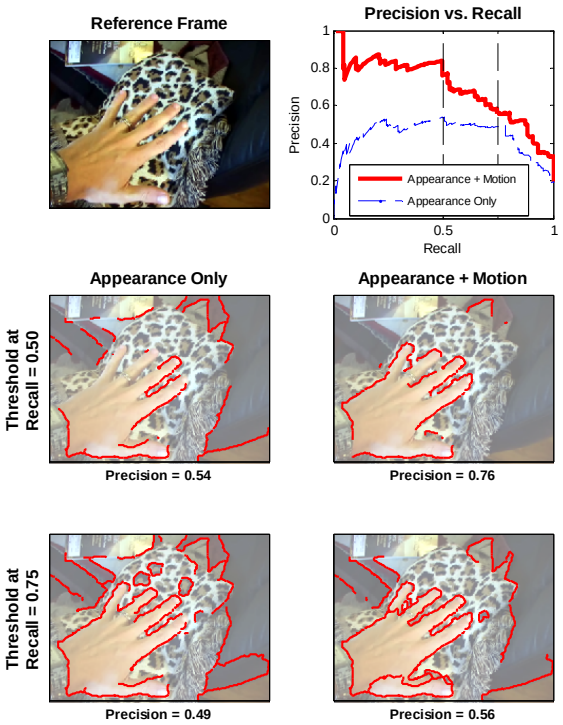


Figure B.6: Hand

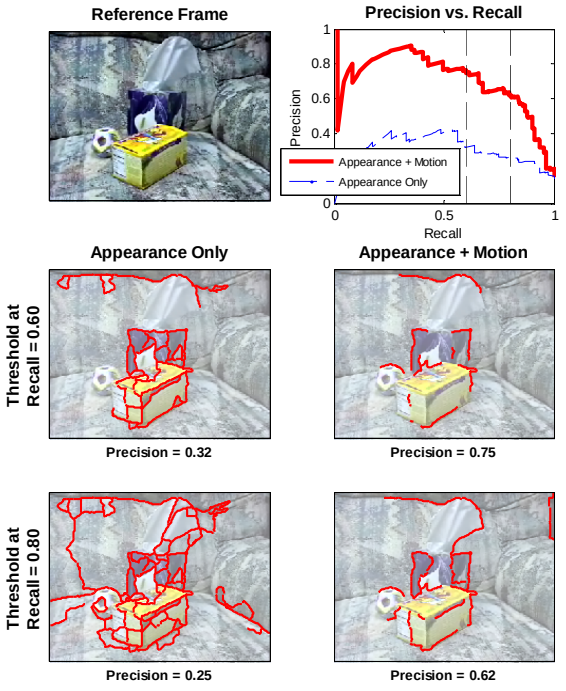


Figure B.5: Couch Objects

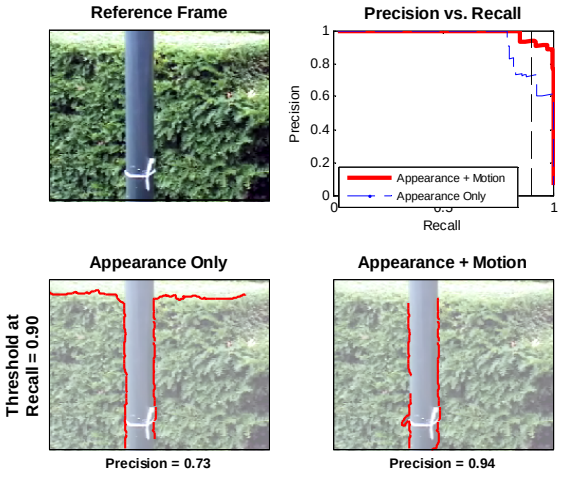


Figure B.7: Post

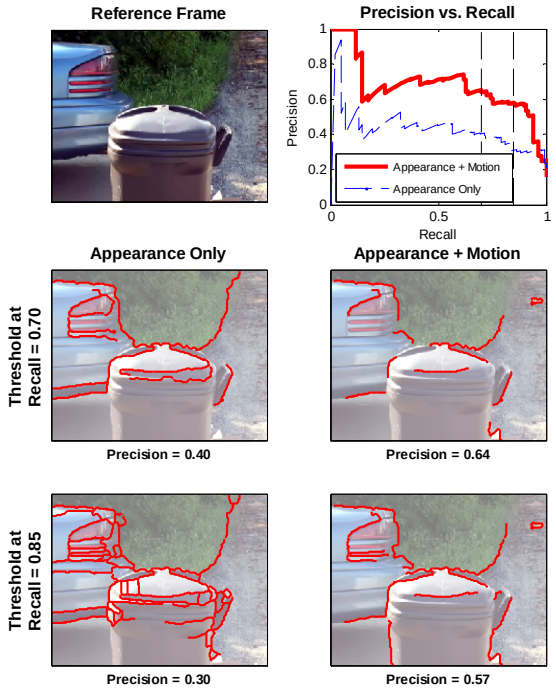


Figure B.8: Trash Can

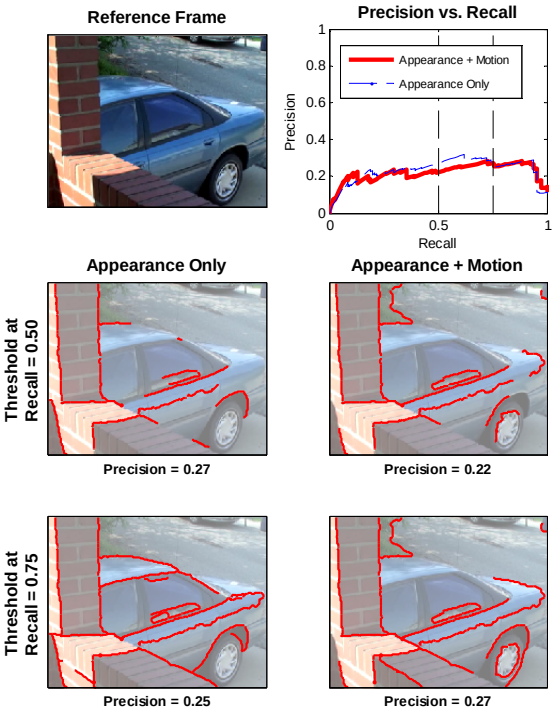


Figure B.10: Car (Difficult Case)

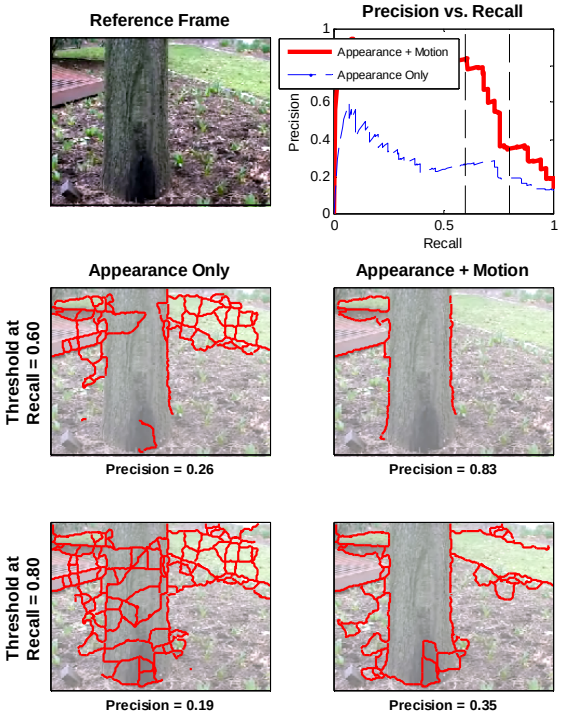


Figure B.9: Tree

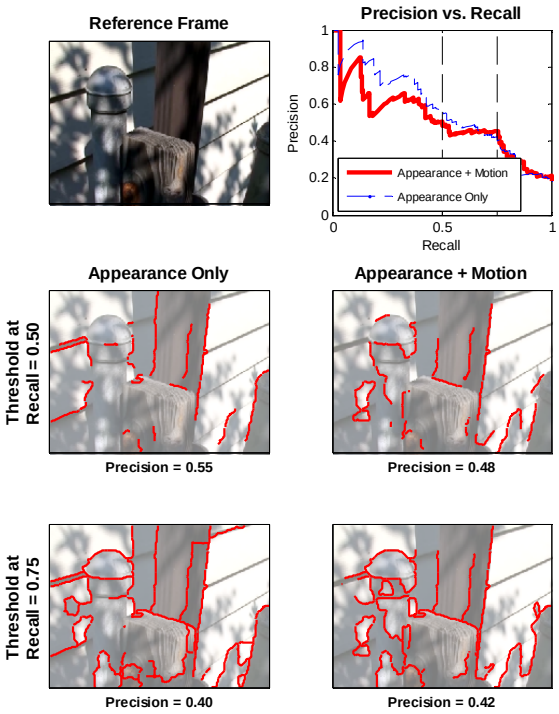


Figure B.11: Fence (Difficult Case)

Appendix C

Additional Object Segmentation Results

In this appendix, we provide additional example results (Figures C.1-C.38) of the same type as the examples in Chapter 5, but over a larger range of desired consistency. On individual examples, either BSE or the Multiscale NCuts approach sometimes outperforms our method, but neither does so in general. Our matting-based approach offers solid performance, if not marked improvement, more consistently. We also show examples of "easy" objects, for which all methods work well, and "hard" (often small) objects, which prove challenging for all methods.

At the top of each figure is the bar graph displaying the number of segments required by each method to achieve the desired consistency on the x-axis. Below this graph is a comparison of the segmentations produced by each method which achieve the desired consistency indicated at the left of each row, while using the minimum number of segments. Recall that the approaches shown in the two middle columns use pixelwise affinities, while our approach and the color-based approach both use super-pixels. For visualization, the segments used in computing the actual per-object consistency and efficiency values (displayed beneath each segmentation) are colored using a red-yellow colormap, while the non-object segments are colored with a blue-green colormap. Thus, all of these figures are best viewed in color. Note that a separate figure exists for each object of scenes with multiple objects (the Ground Truth Object Mask will differ in these cases).

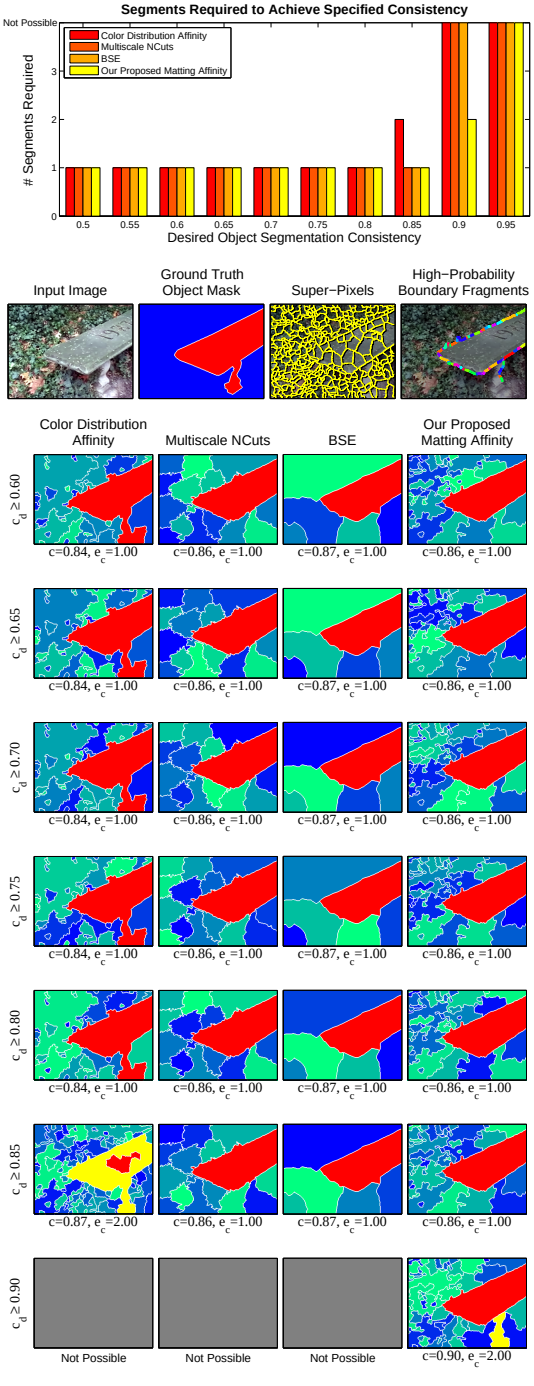


Figure C.1: Bench. An "easy" example for which all methods work quite well.

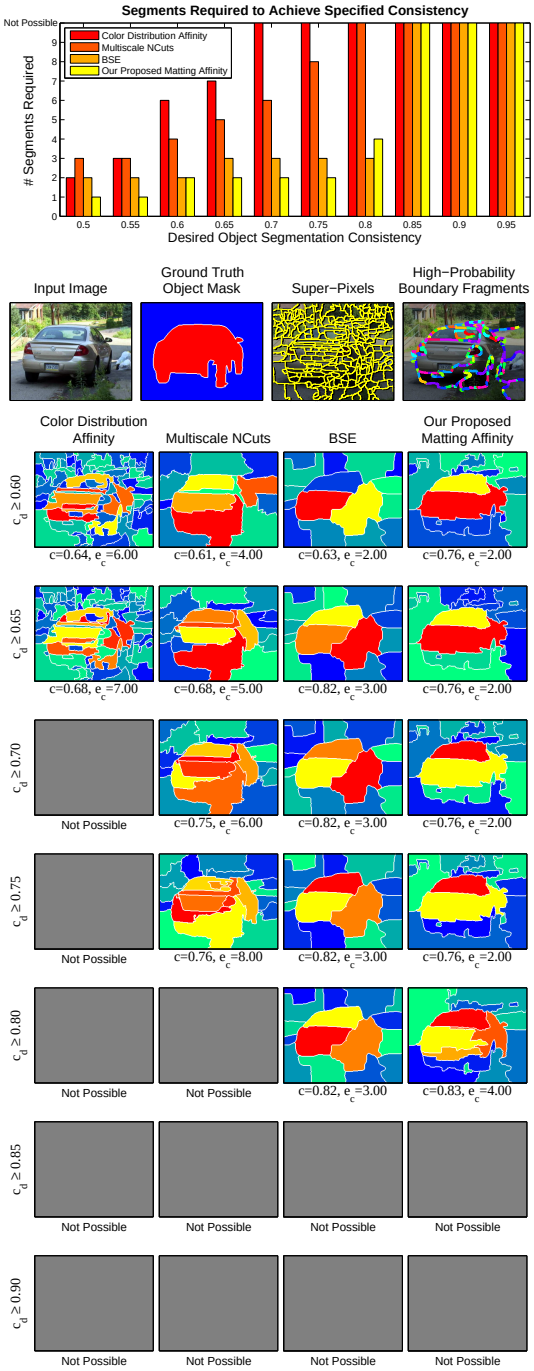


Figure C.2: Car.

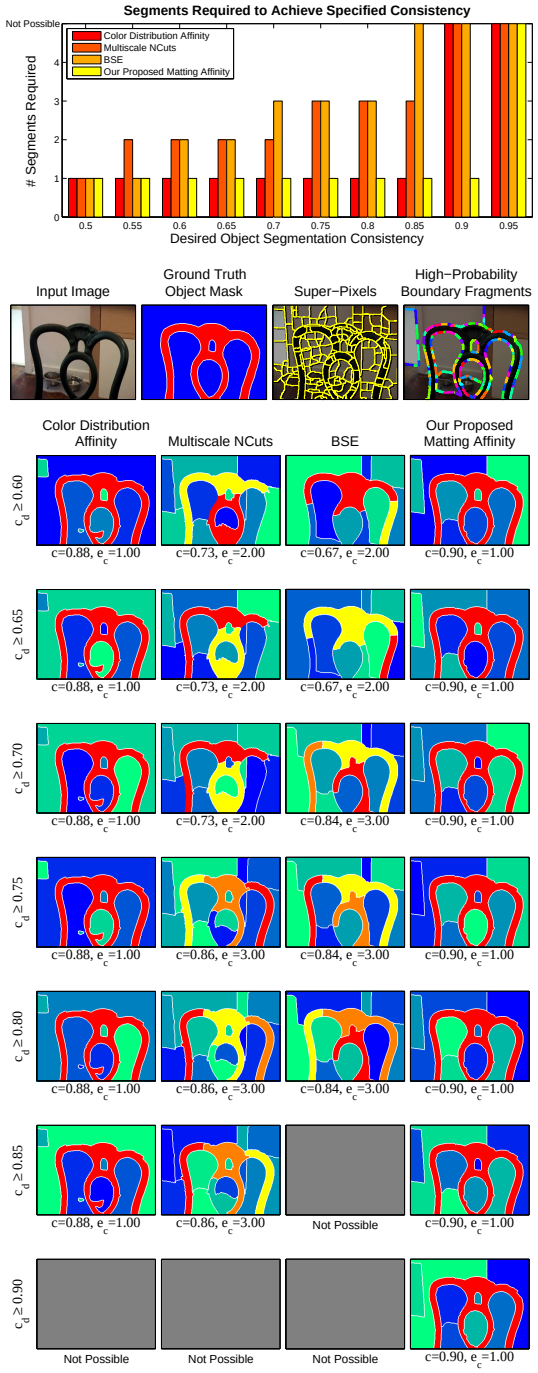


Figure C.3: Chair. Both super-pixel-based methods perform well on this (and other) objects with narrow structures, unlike the pixelwise approaches which tend to break such objects apart with "cheap" cuts across narrow parts. Not surprisingly, color alone is sufficient for this object, though it is worth noting that the matting-based approach does not *hurt* (and actually helps some).

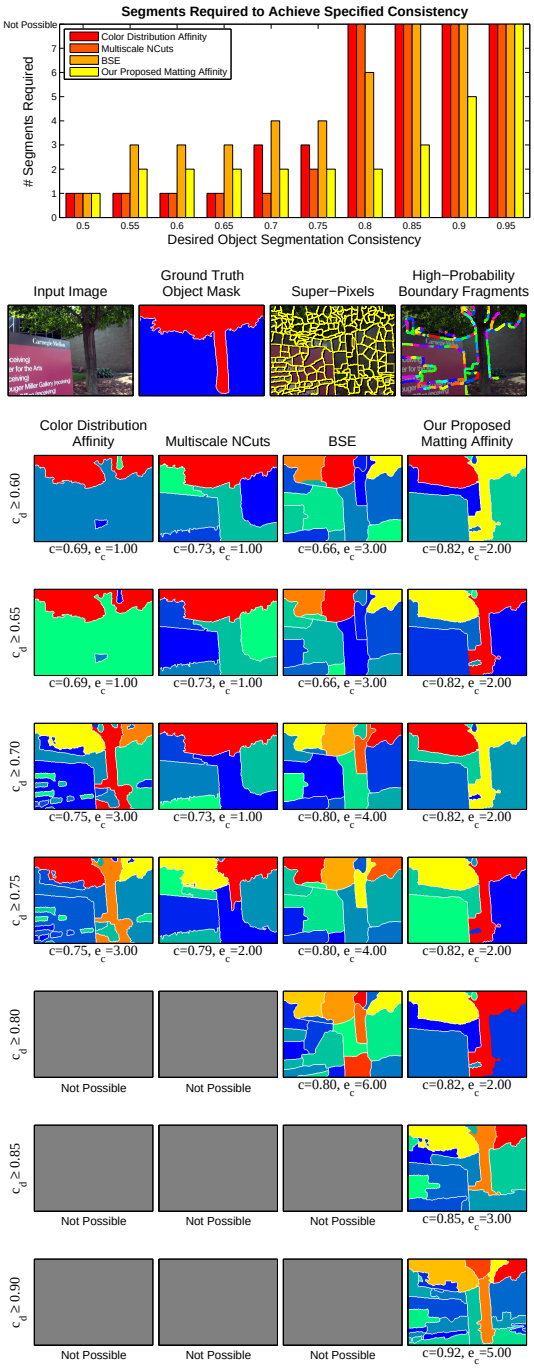


Figure C.4: Tree

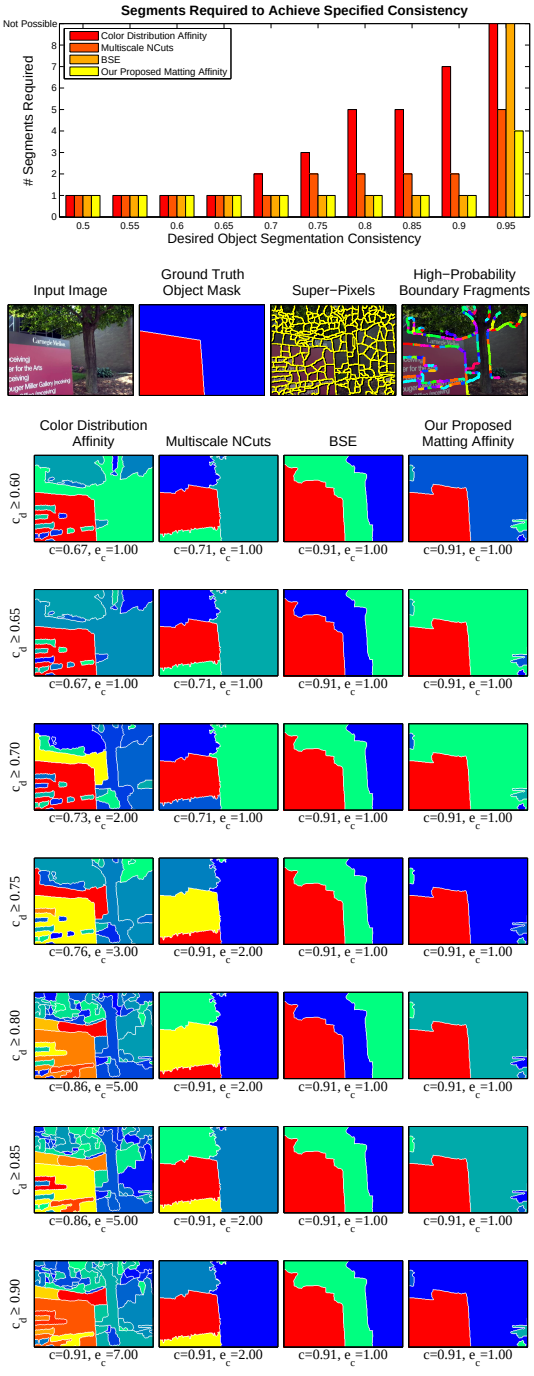


Figure C.5: Sign. As discussed in Chapter 5, evaluating segmentation is difficult. Numerical results do not always tell the whole story: our approach and BSE offer the same consistency (0.91) with only one segment, but qualitatively, our segmentation seems "better."

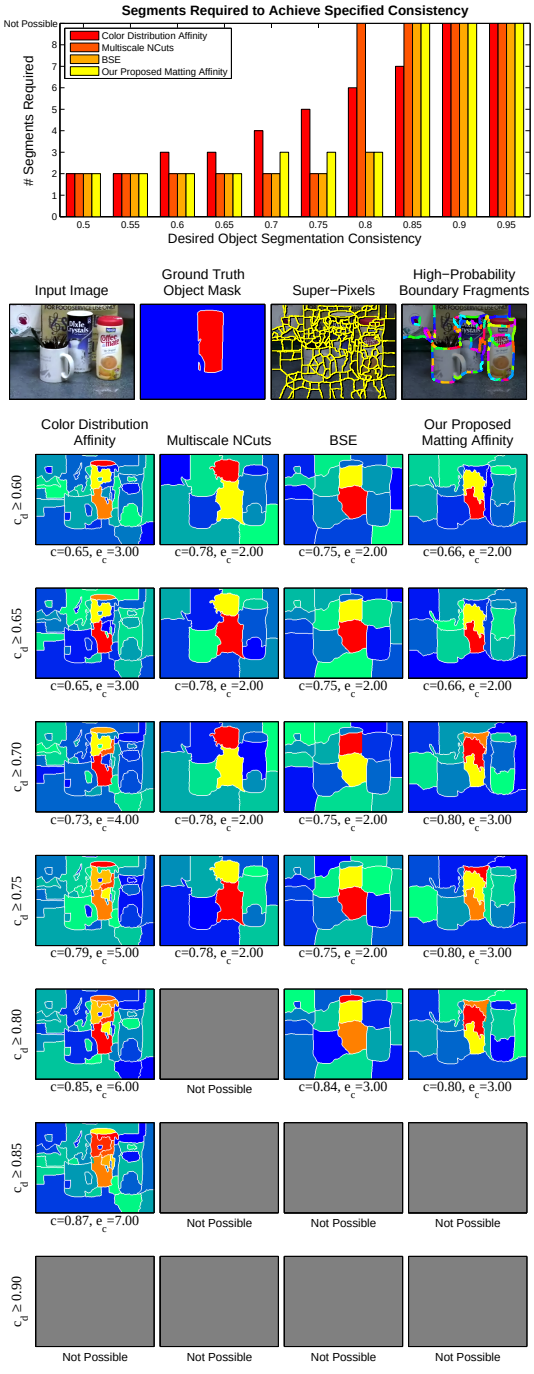


Figure C.6: Coffee sugar.

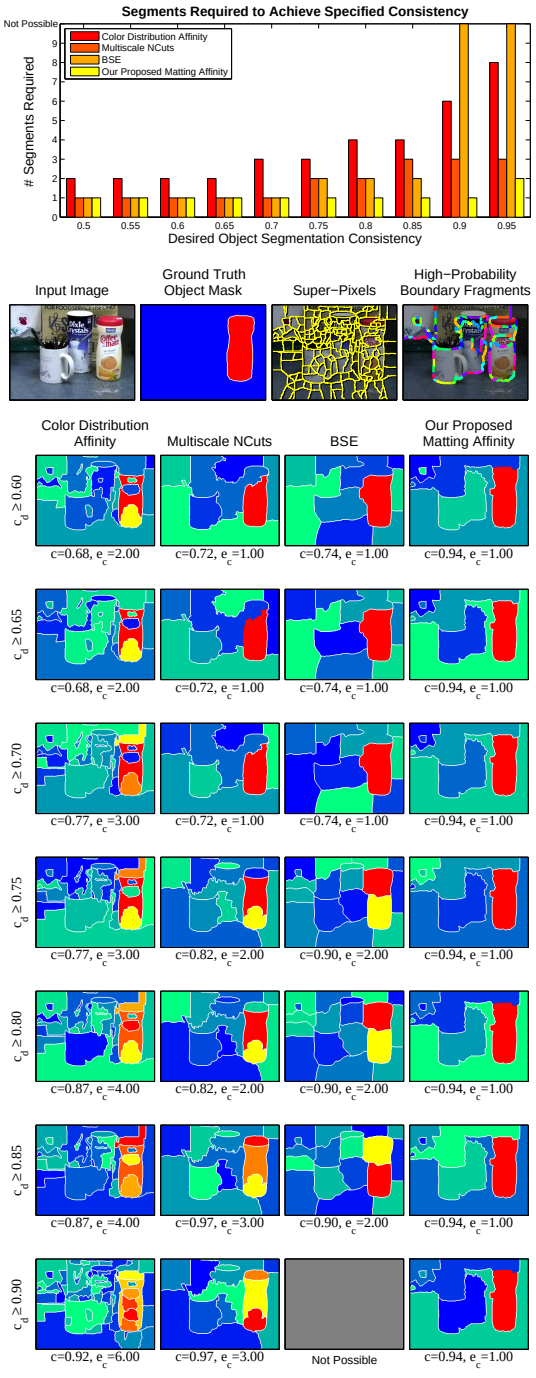


Figure C.7: Coffee creamer.

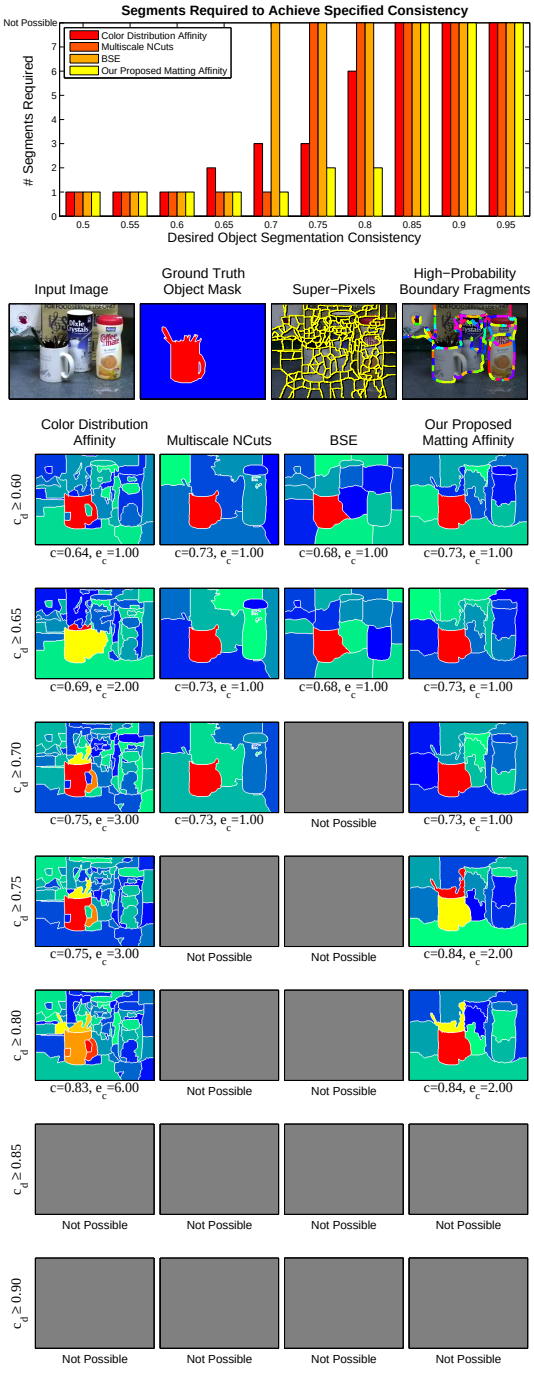


Figure C.8: Coffee mug. Some subjectivity in selecting "whole objects" remains: the mug has been labeled together with straws and plastic-ware inside it, but each method clearly – and justifiably – attempts to separate the white mug from the black contents.

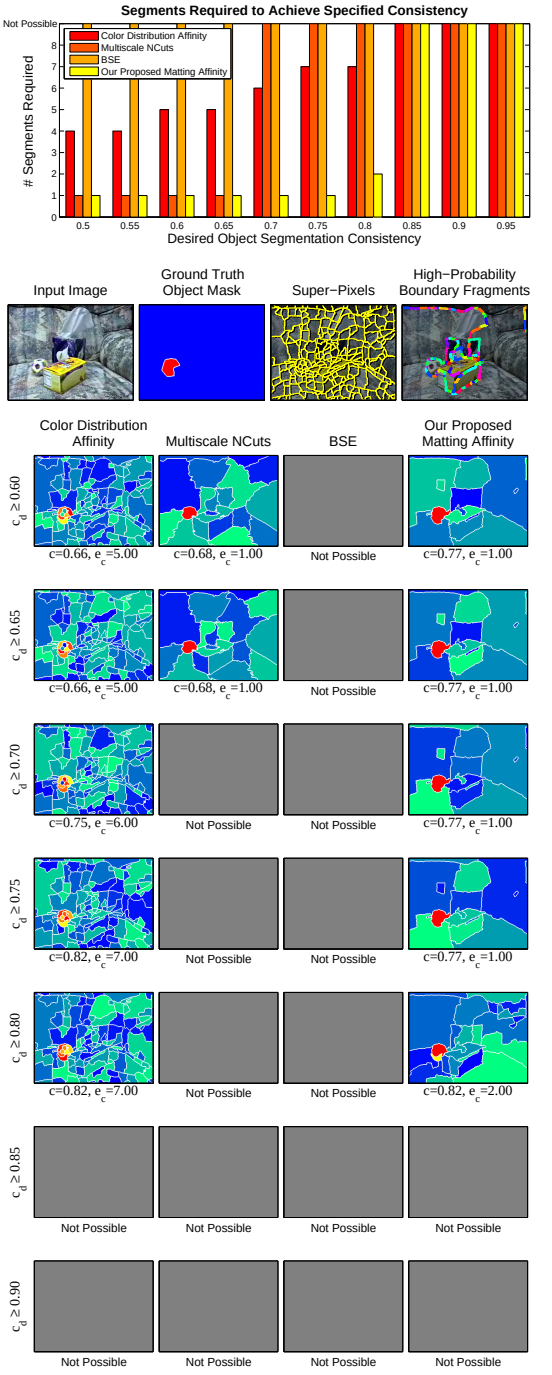


Figure C.9: Origami ball. Both pixel-based methods have trouble with this small object. The super-pixel approaches fare better, but the matting-based approach achieves superior efficiency, despite the ball's multiple colors.

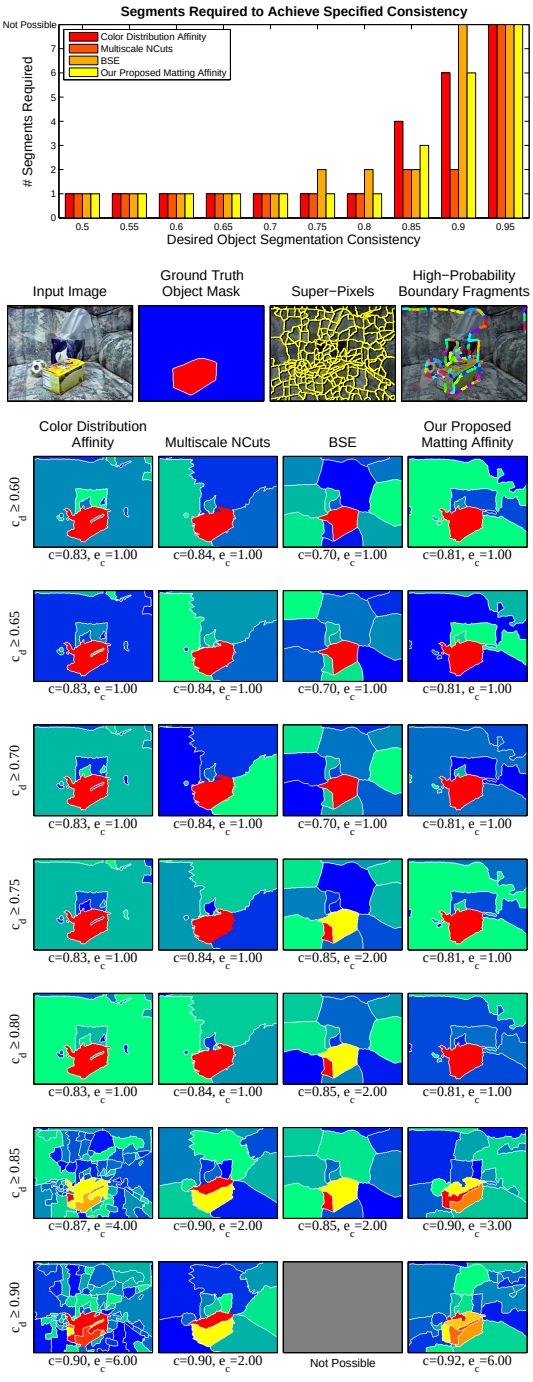


Figure C.10: Tea Box.

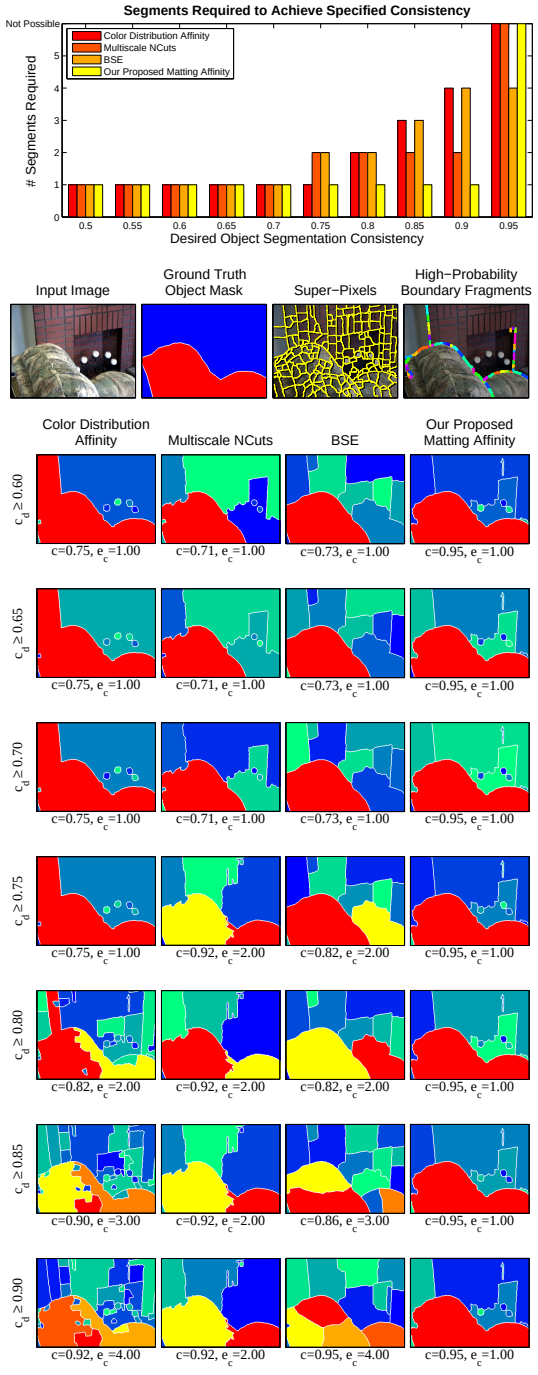


Figure C.11: Despite the hypothesized boundary fragments between the two parts of the couch, the appearance reasoning of the mattes puts the whole couch in one segment for our approach. Also, the use of super-pixels seems to help avoid a cheap cut through the object to the bottom of the image.

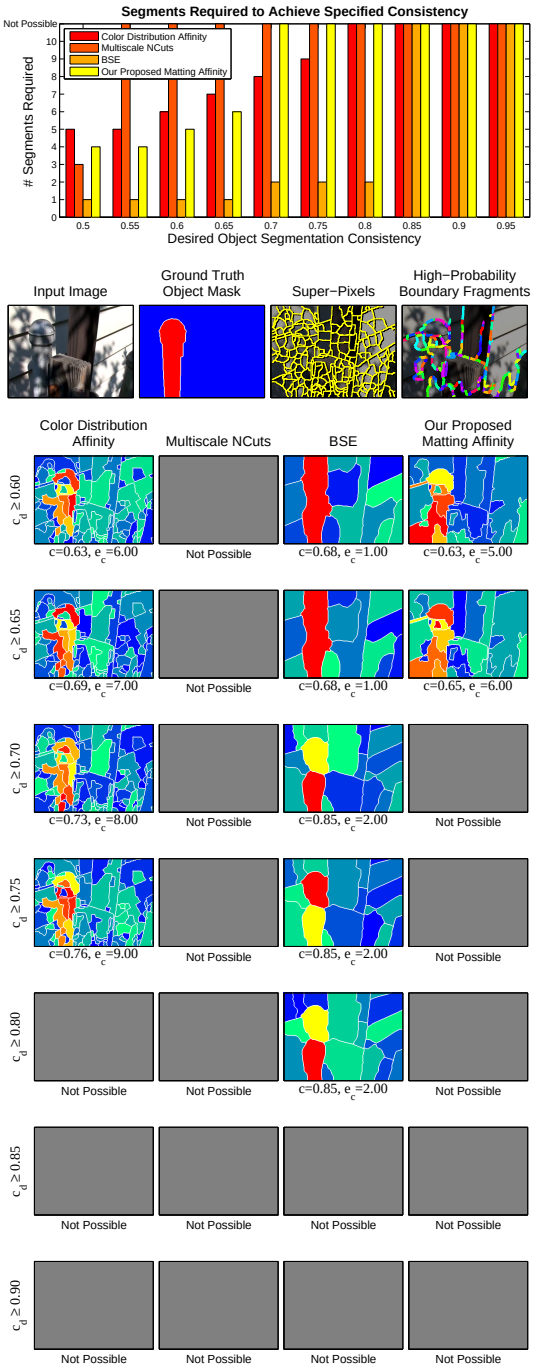


Figure C.12: The high-contrast illumination and shadows in this scene make it very difficult. They seem to confuse the boundary-hypothesis step and, in turn, our matting-based approach. (See also Figures C.13-C.14.)

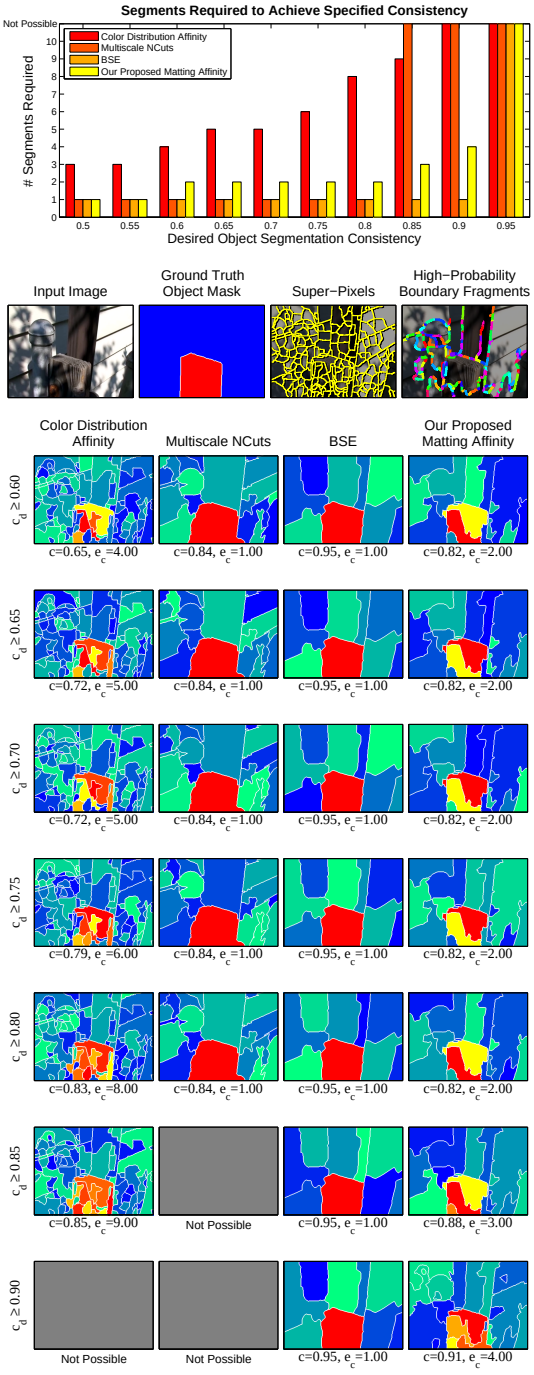


Figure C.13: Difficult fencepost scene.

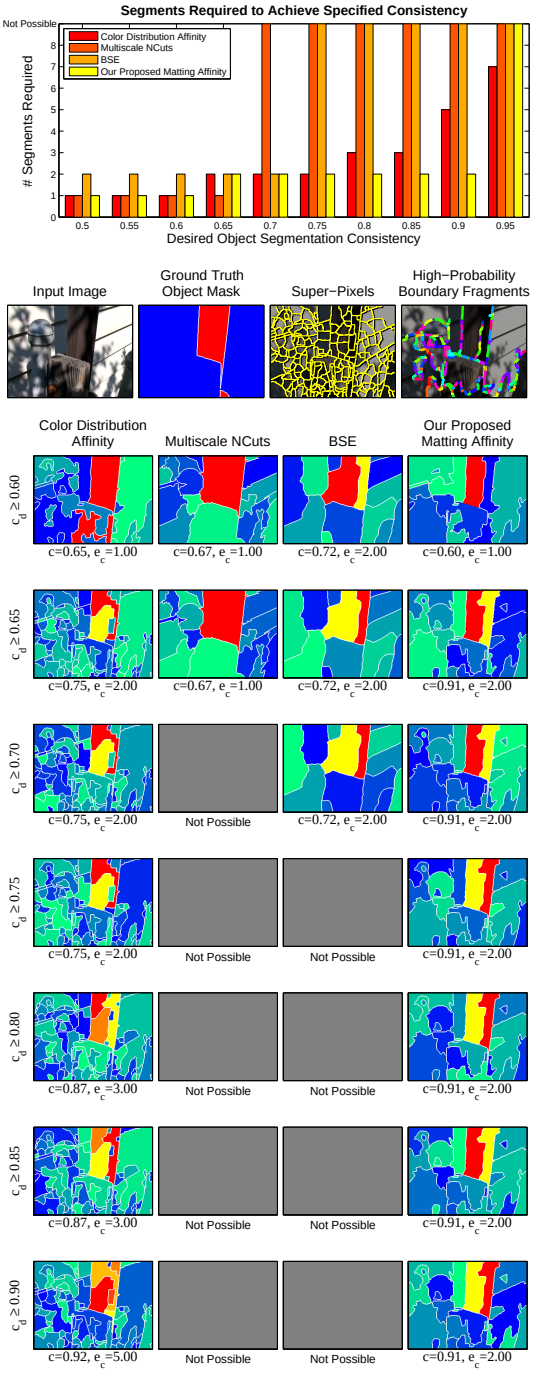


Figure C.14: Difficult fencepost scene.

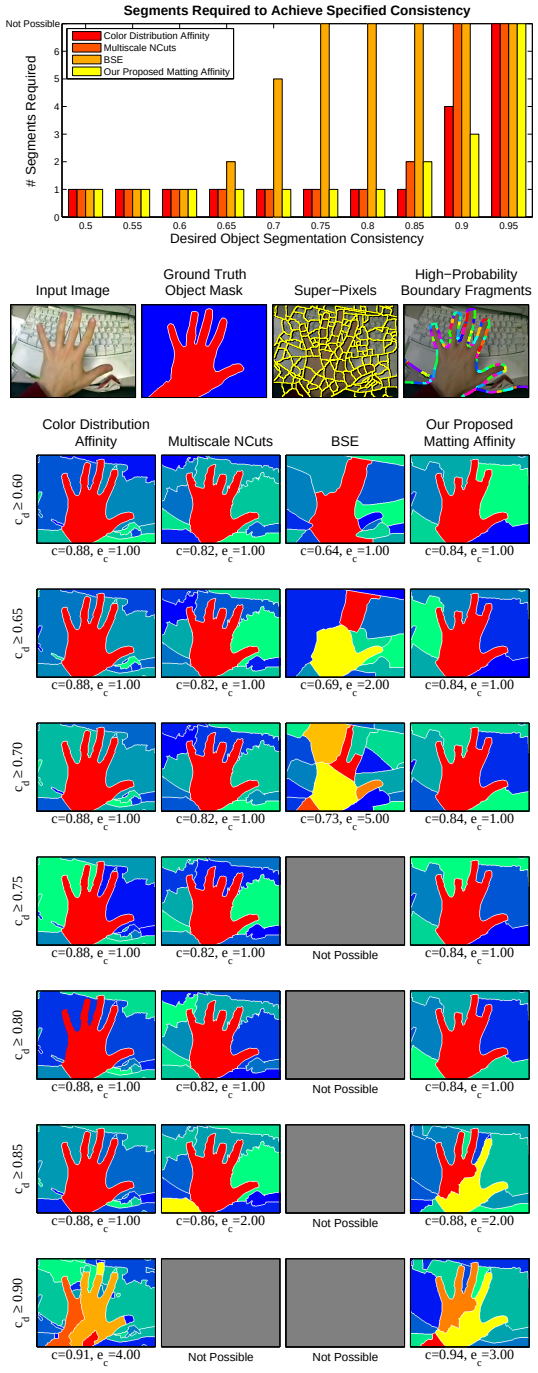


Figure C.15: Hand 1. Here, color alone is a strong cue for segmentation, but our approach also performs fairly well. All methods are reluctant to include the differently-colored bit of sleeve labeled in the ground truth.

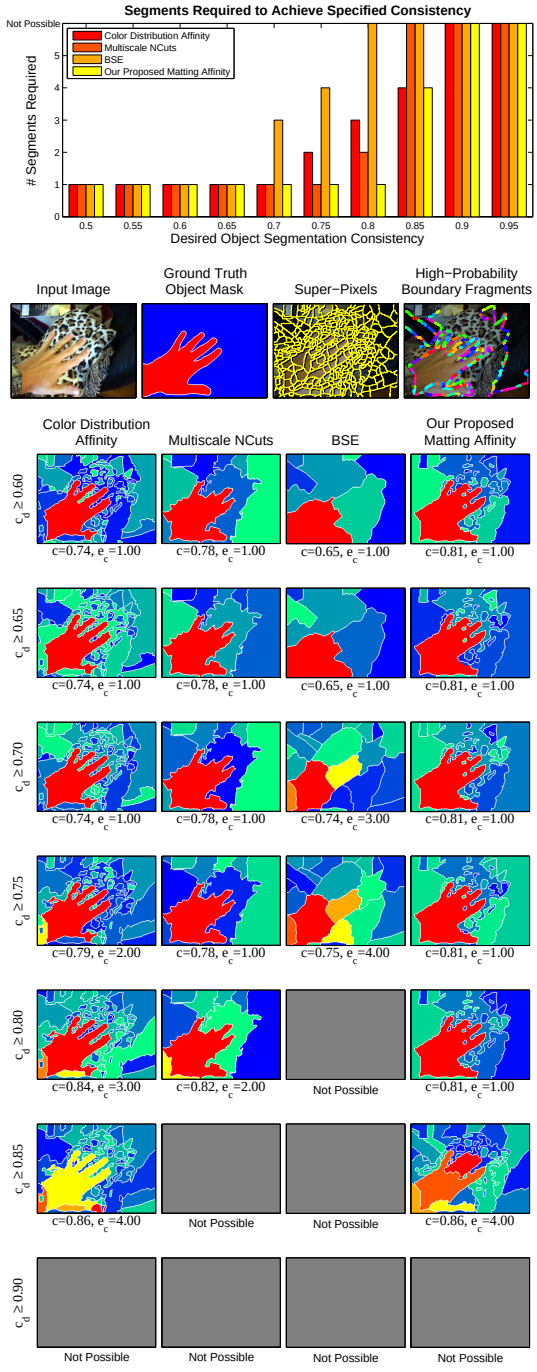


Figure C.16: Hand 2. The texture of the blanket behind the hand serves to confuse all methods, but ours does a good job of extracting the fingers. Here again, super-pixels seem to help with the extraction of narrow structures (particularly as compared to BSE).

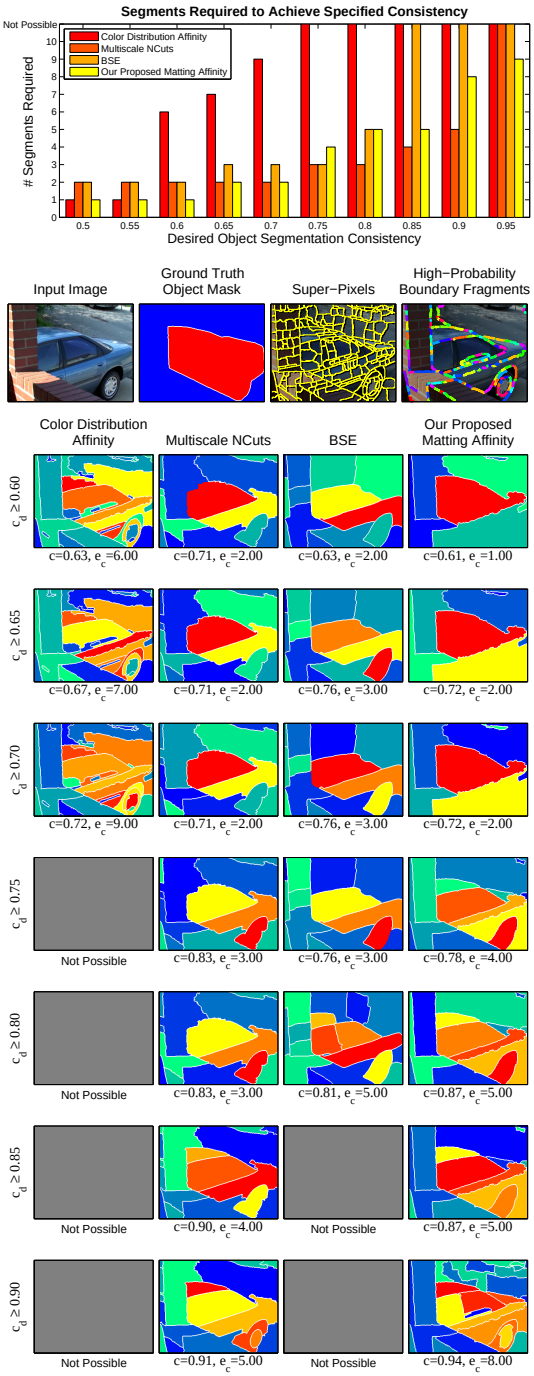


Figure C.17: Car.

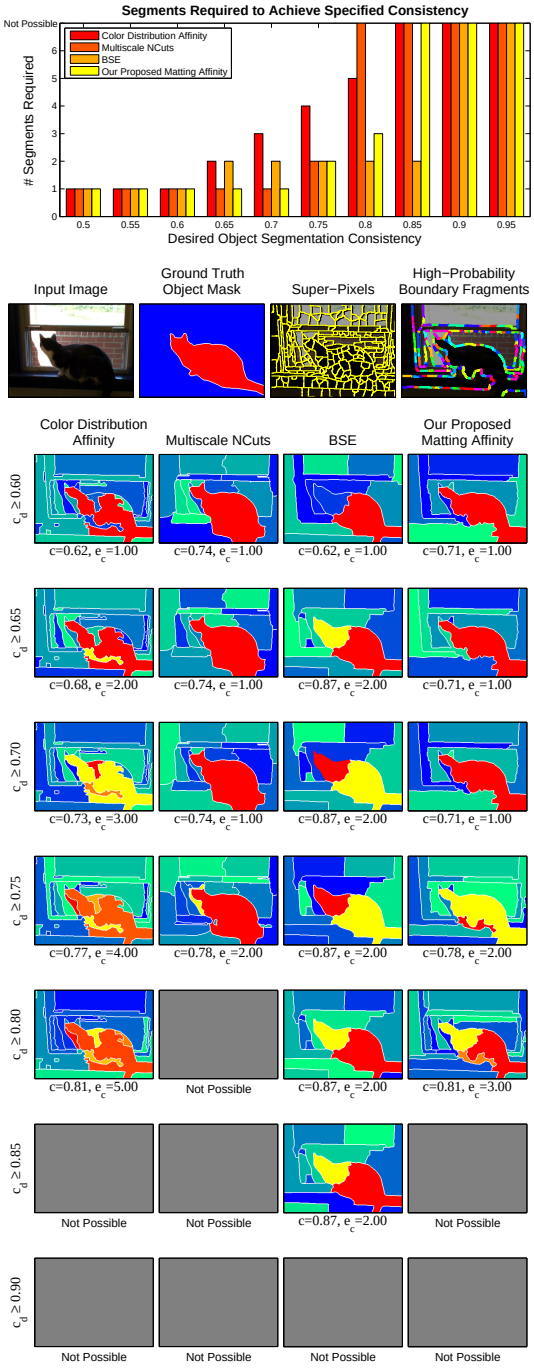


Figure C.18: Cat in window. The very low contrast between the cat and the shadowed window frame make this a difficult scene.

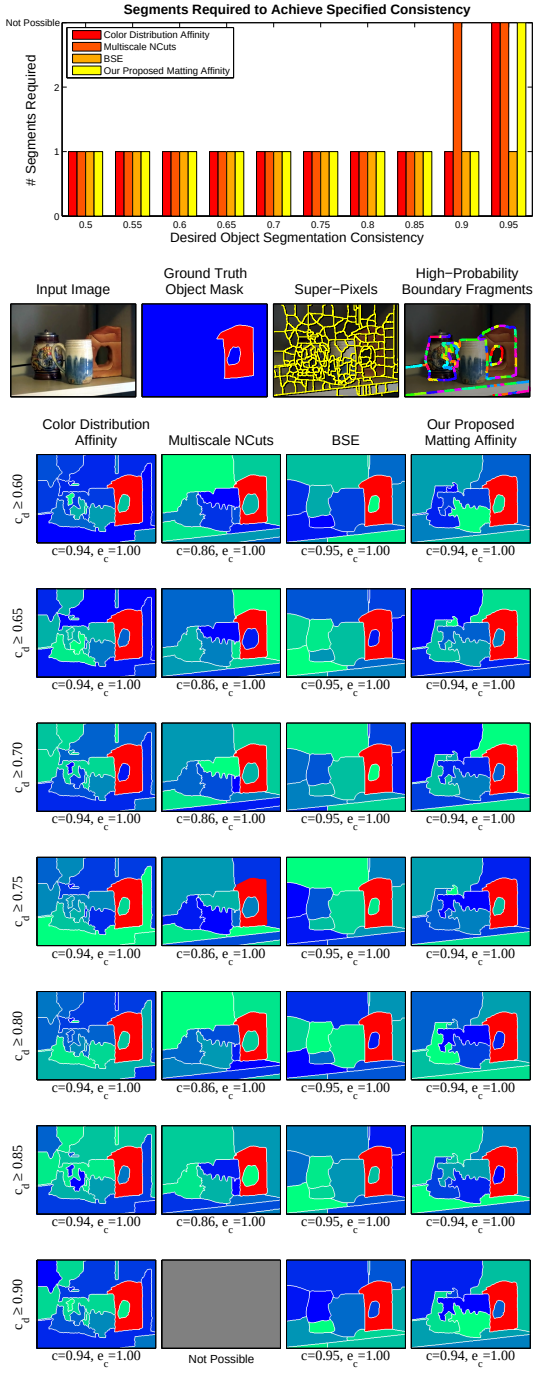


Figure C.19: Bookend. Another "easy" object for which all methods perform well.

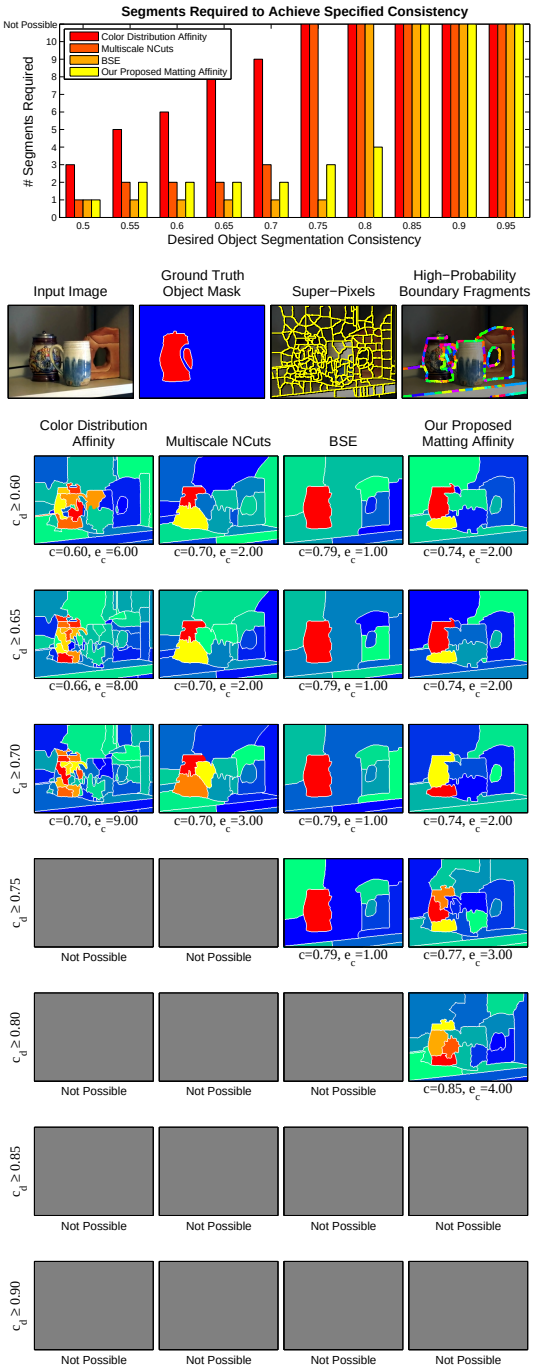


Figure C.20: Beer stein.

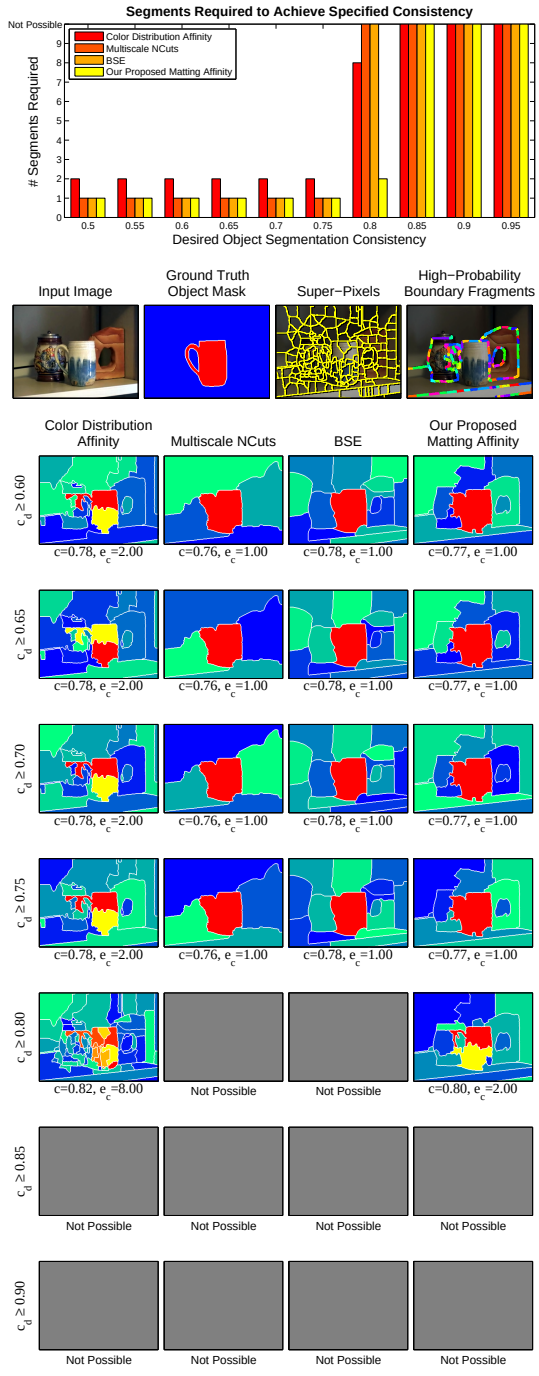


Figure C.21: Mug.

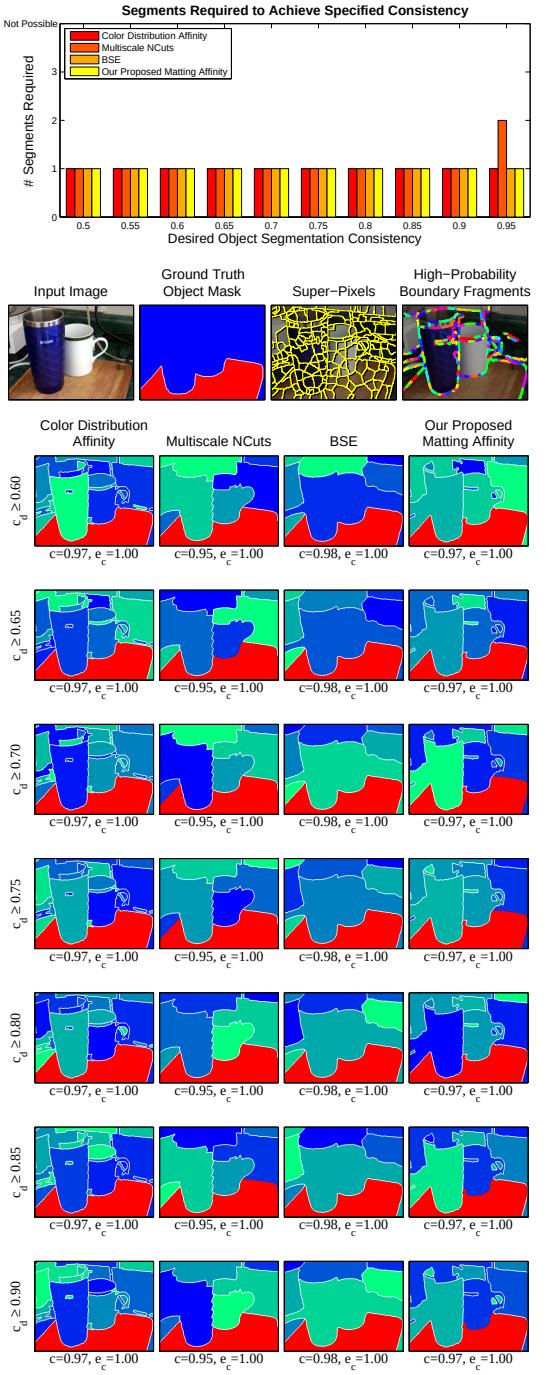


Figure C.22: Cutting board. With such uniform appearance, this is not a difficult object for any method. It is somewhat surprising, however, that neither pixelwise NCuts approach split the object with a cheap cut as in other examples, *e.g.* at the very narrow point between the bottom of the blue cup and the image border. Such unpredictable performance is a common side effect of standard, pixelwise affinities.

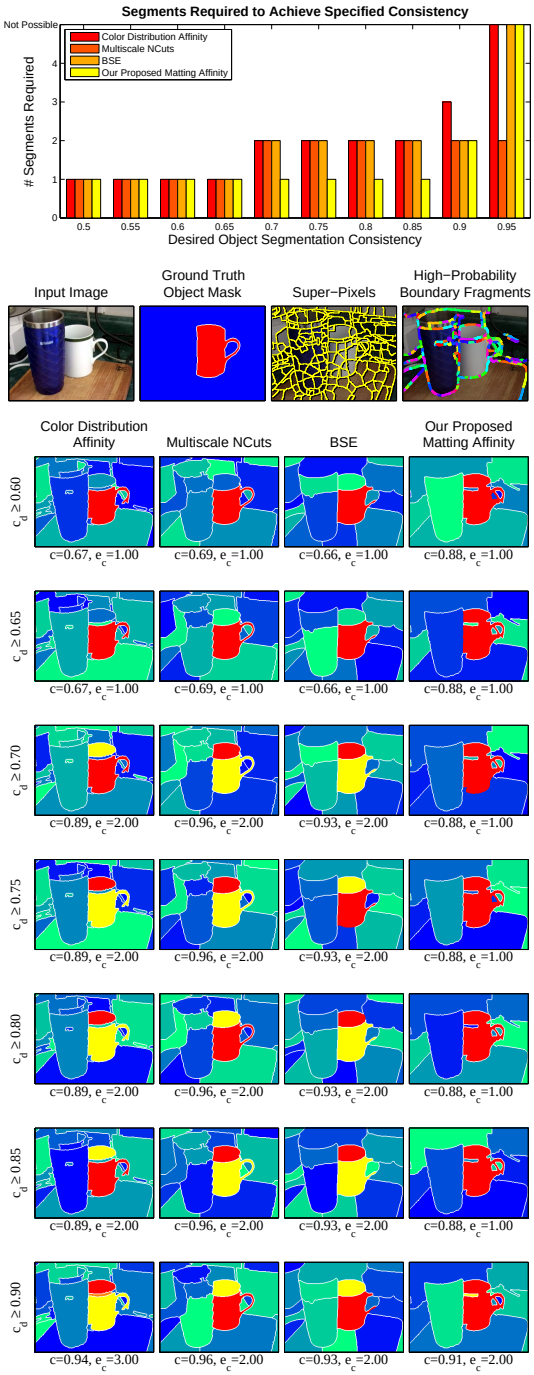


Figure C.23: Coffee mug.

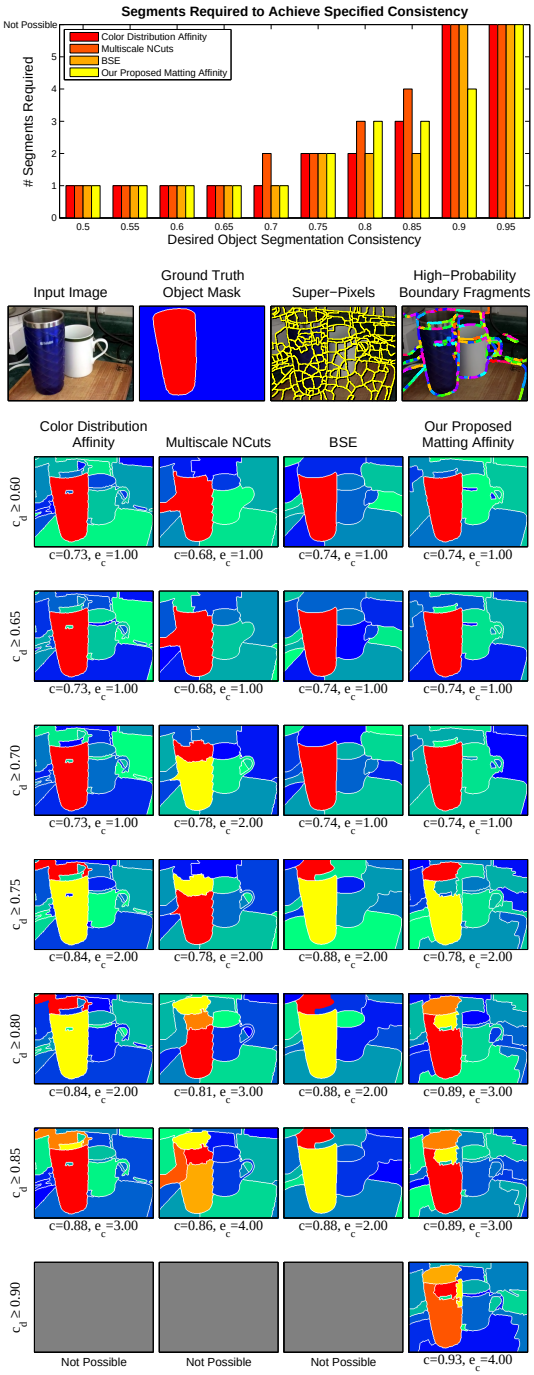


Figure C.24: Cup.

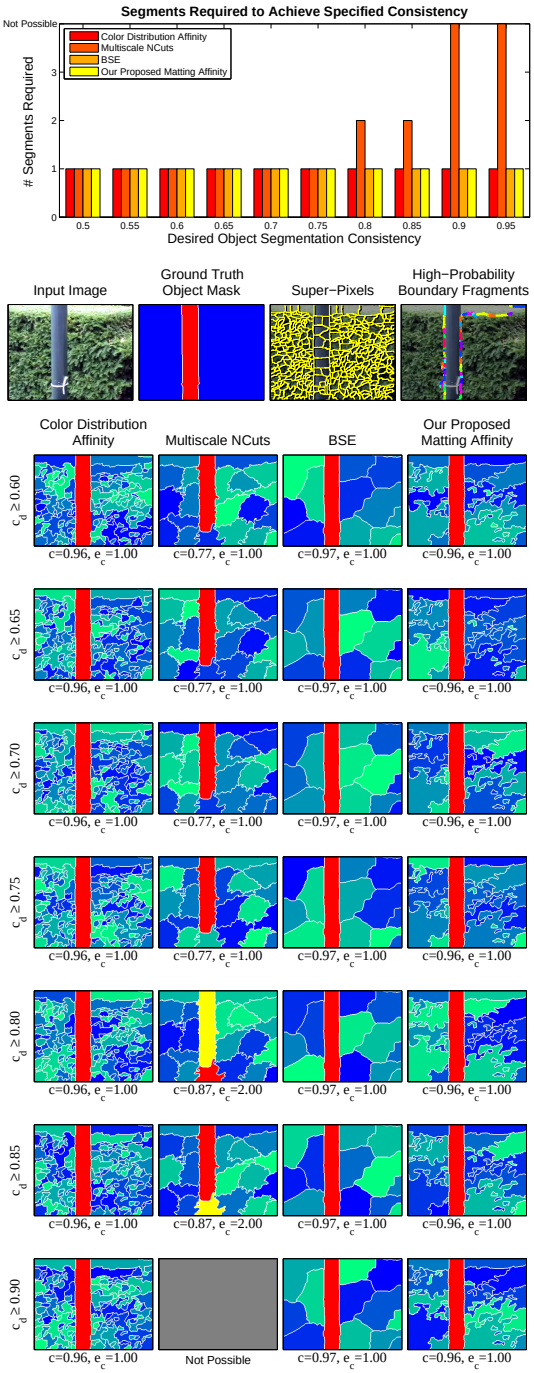


Figure C.25: Post.

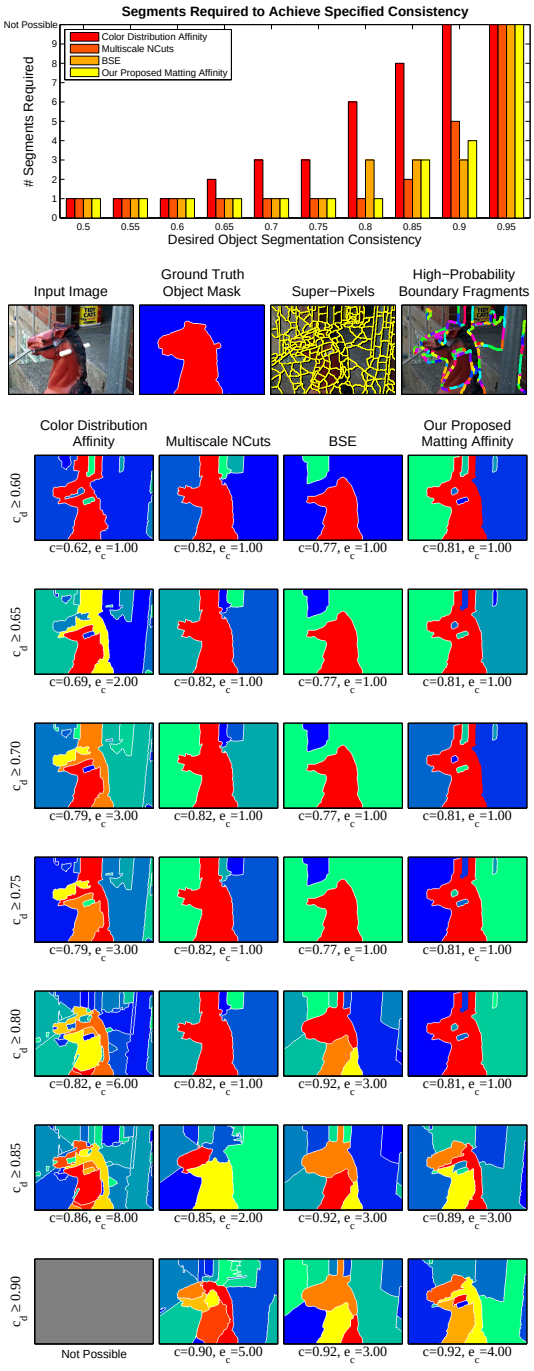


Figure C.26: Rocking horse. The low contrast between the top of the horse's head and the brick cause region "bleeding" for many results.

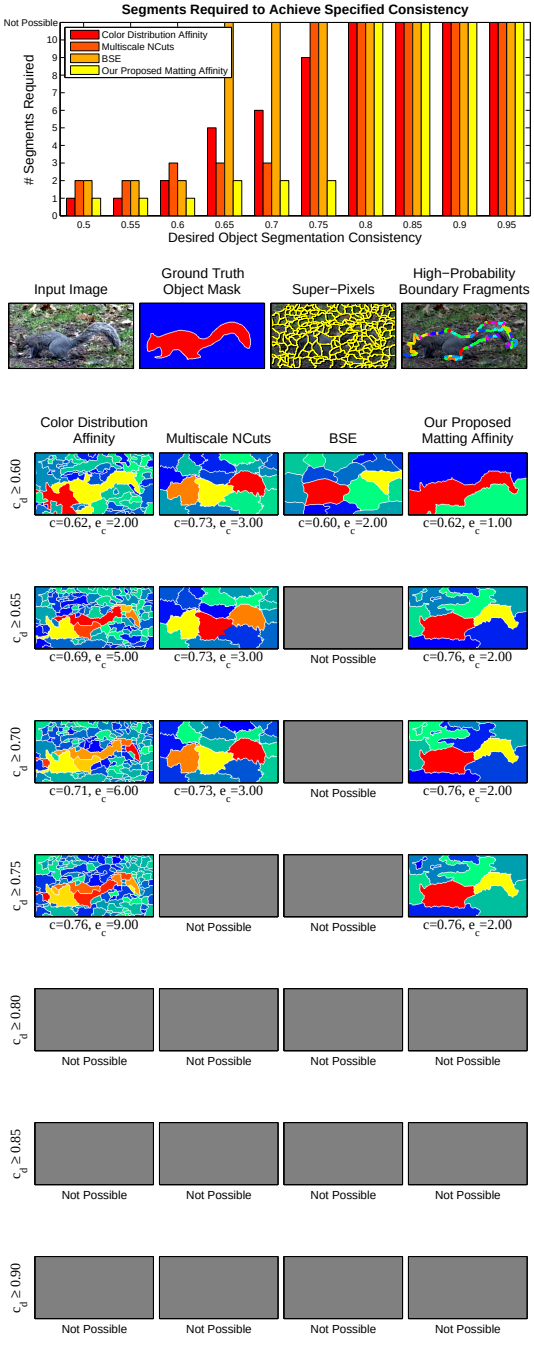


Figure C.27: Squirrel. Good boundary hypotheses along with the strong appearance-reasoning offerend by the mattes help our method outperform the the other approaches on this difficult example, similar to the one in Chapter 5.

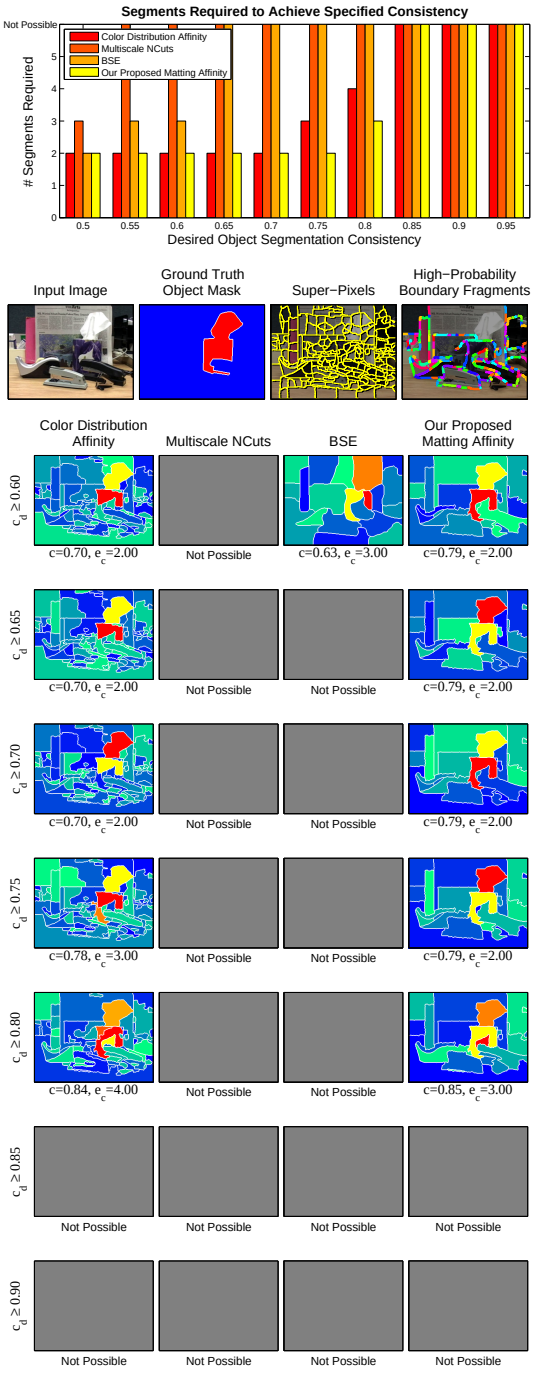


Figure C.28: Kleenex box. Again, color alone seems sufficient here, but unlike the other methods, our matting-based approach does not hurt – and even offers some improvement.

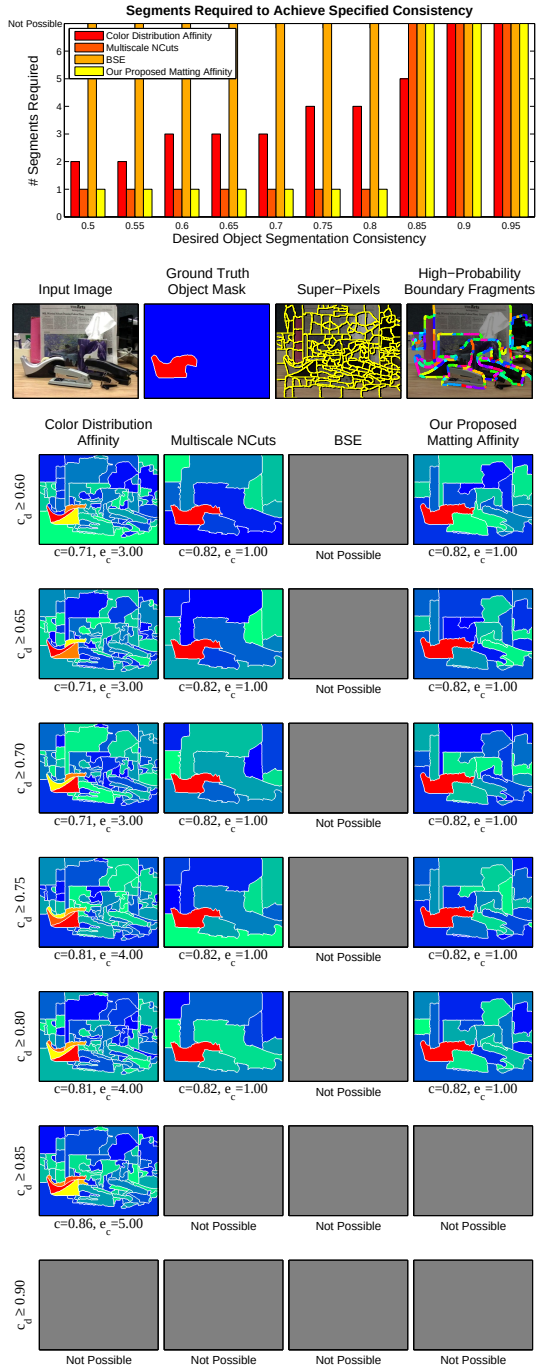


Figure C.29: Tape dispenser.

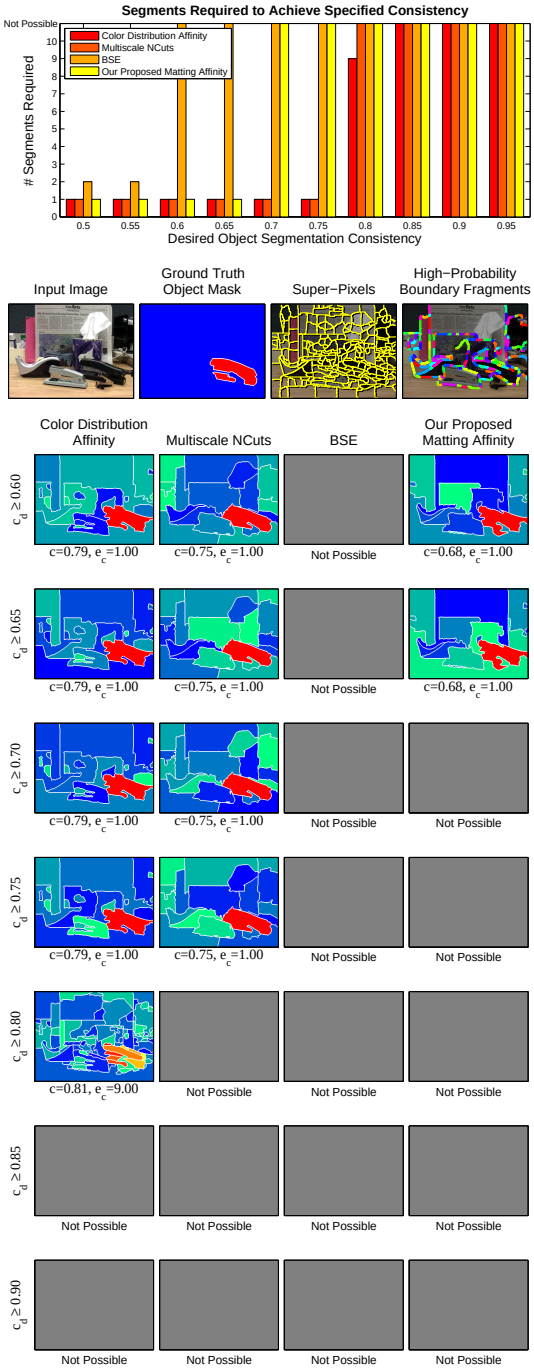


Figure C.30: Stapler 1.

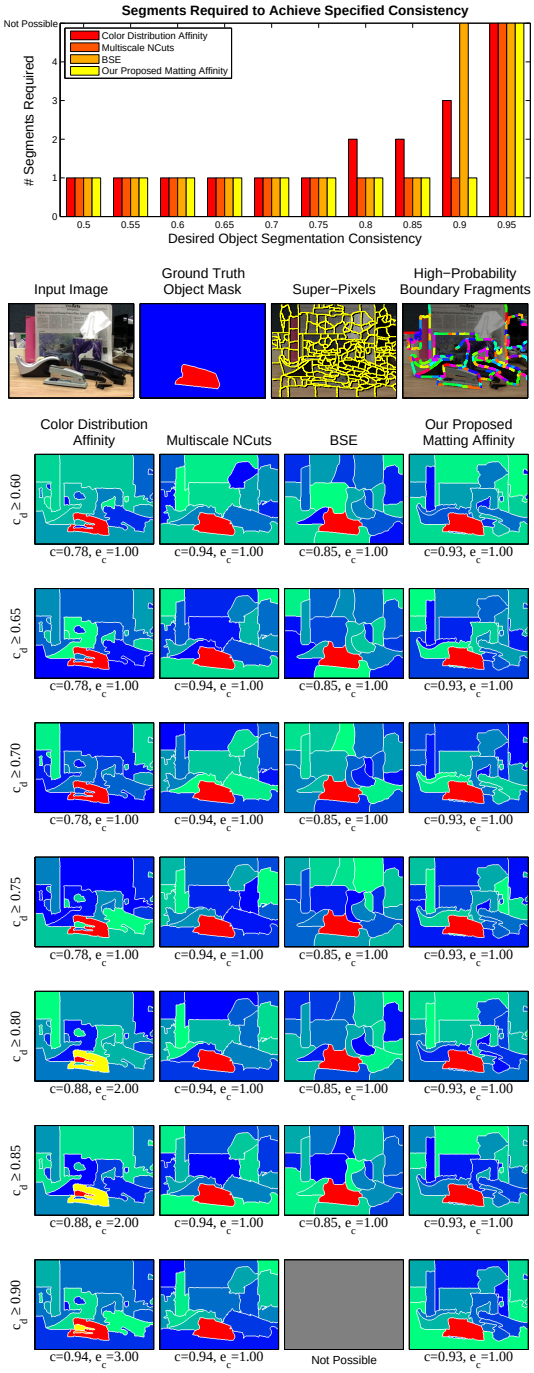


Figure C.31: Stapler 2.

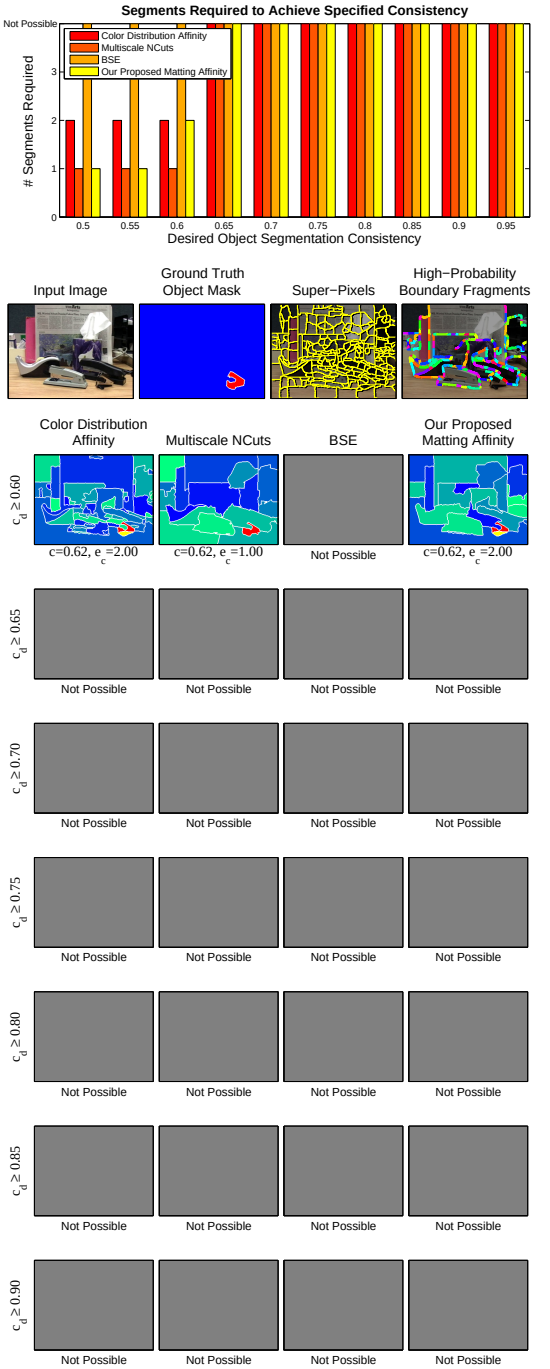


Figure C.32: Staple remover. All methods have a difficult time with this small object, particularly since the top is the same color (black) as the stapler behind it (and thus, the underlying over-segmentation is actually incorrect: the correct boundary fragment is not even hypothesized).

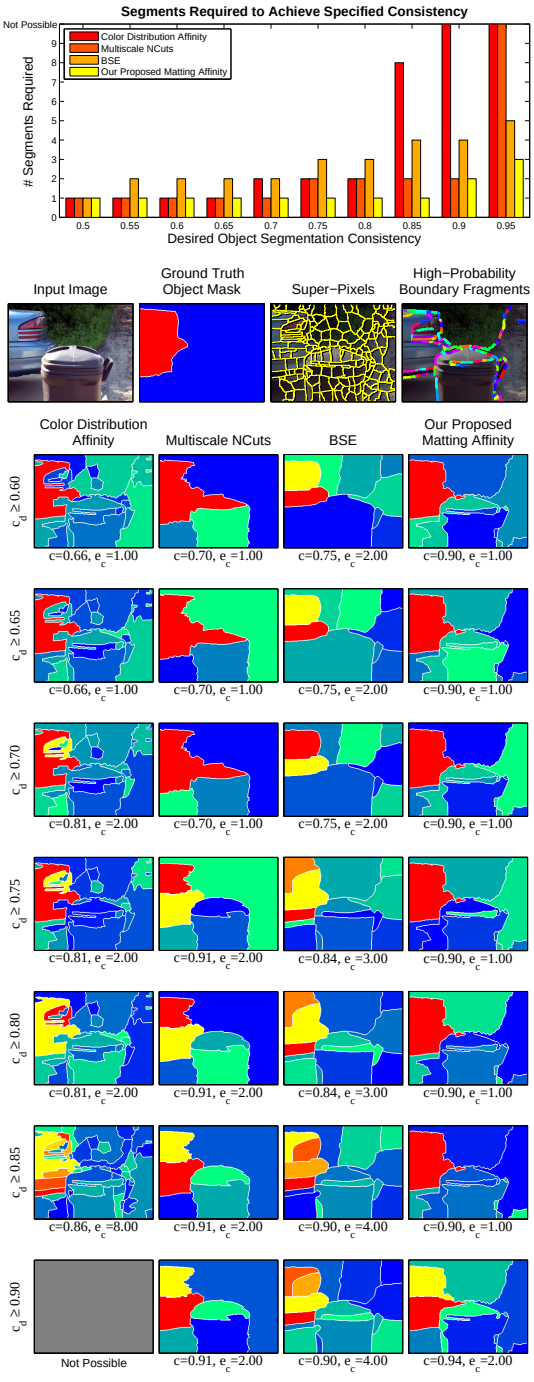


Figure C.33: Car, rear.

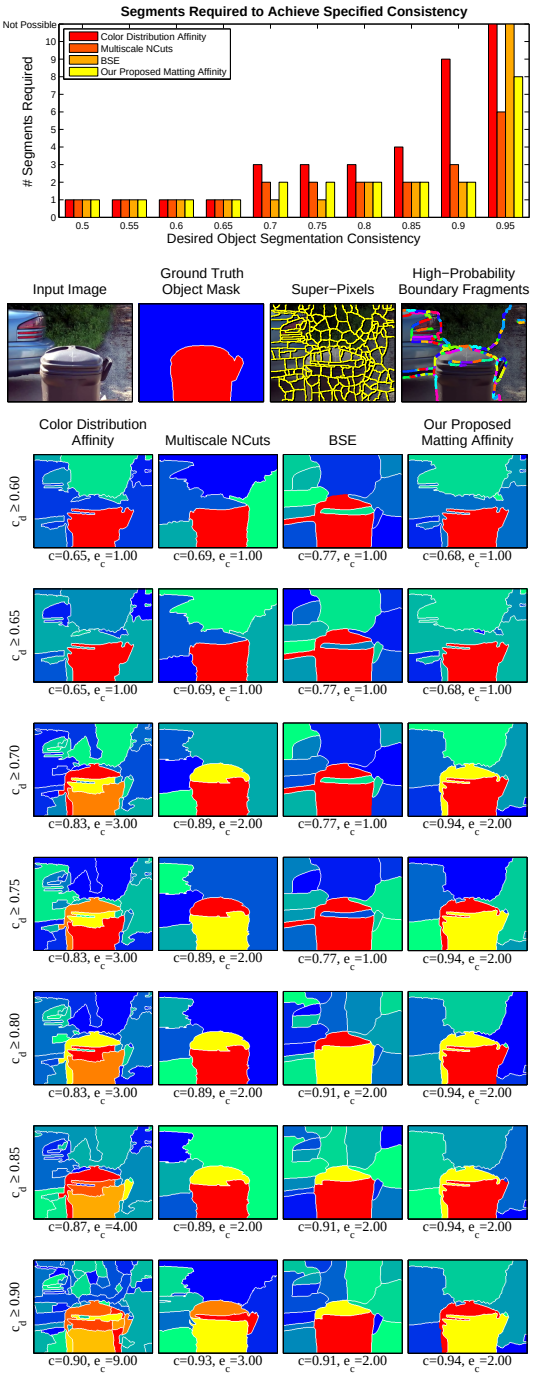


Figure C.34: Trash can.

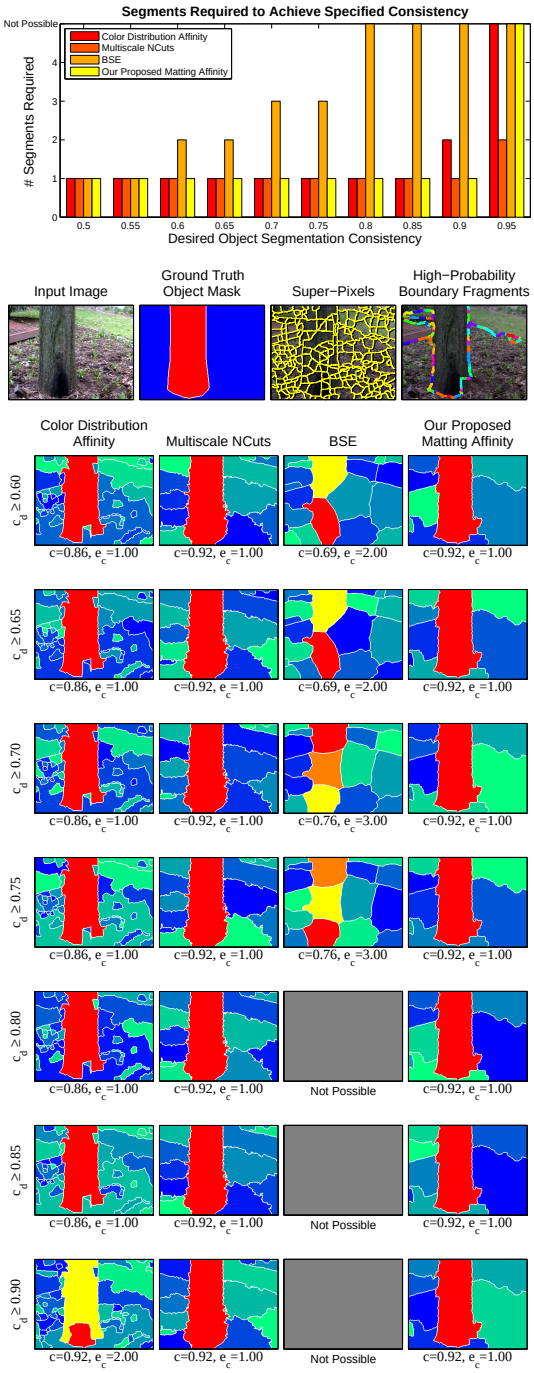


Figure C.35: Tree.

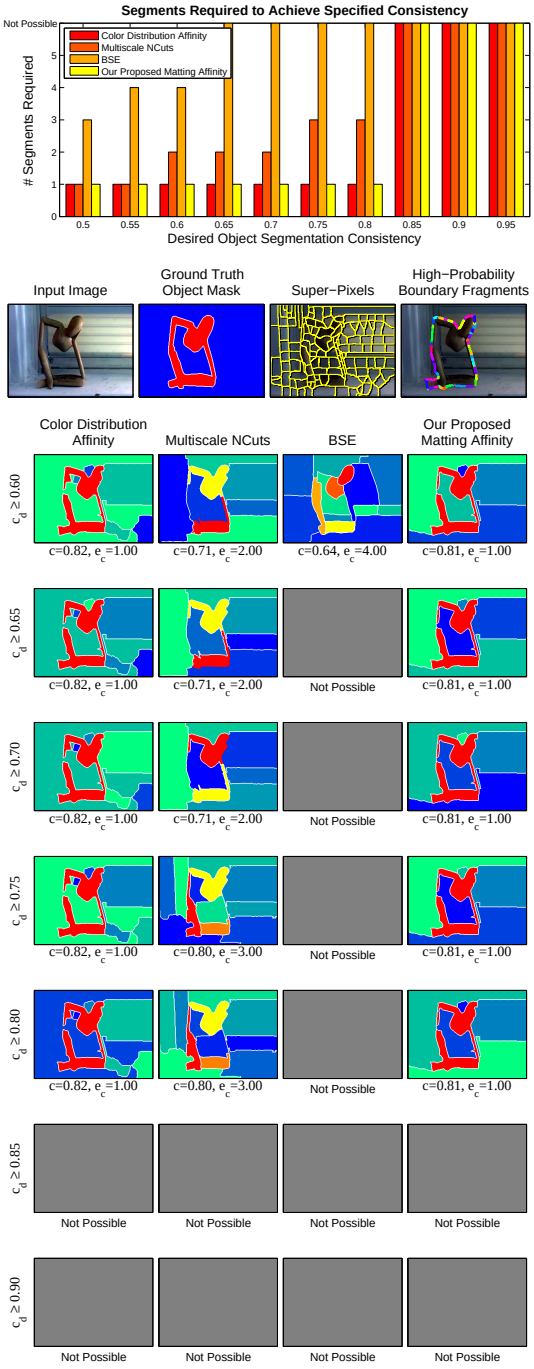


Figure C.36: Wooden statue. Again, narrow structures prove challenging for the methods relying on pixelwise affinities.

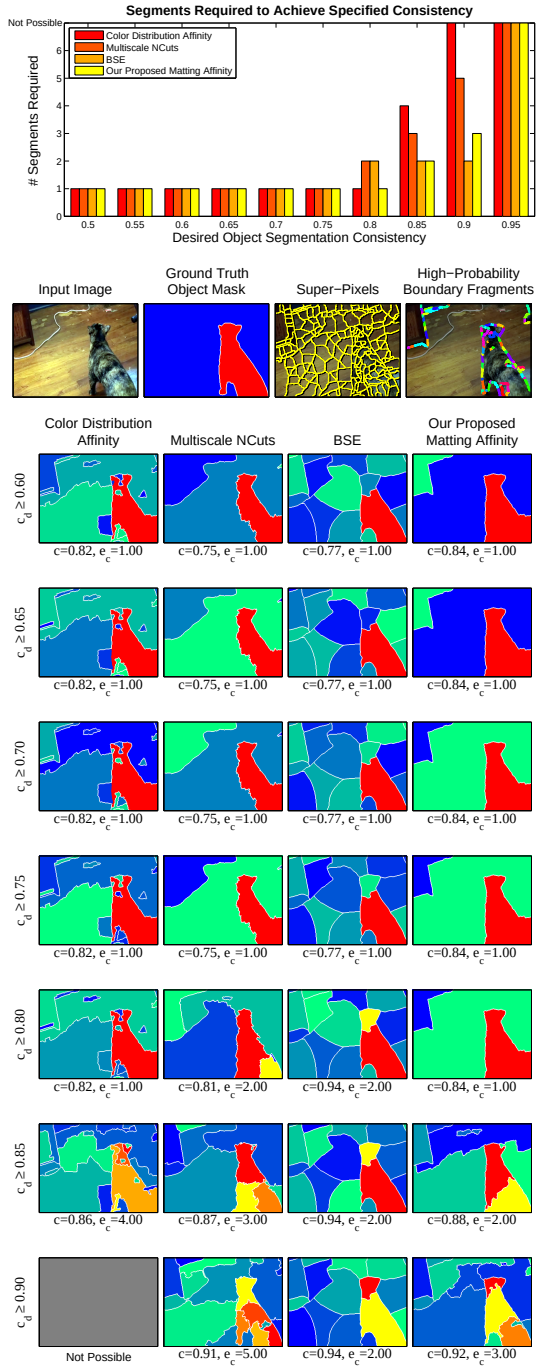


Figure C.37: Cat 1.

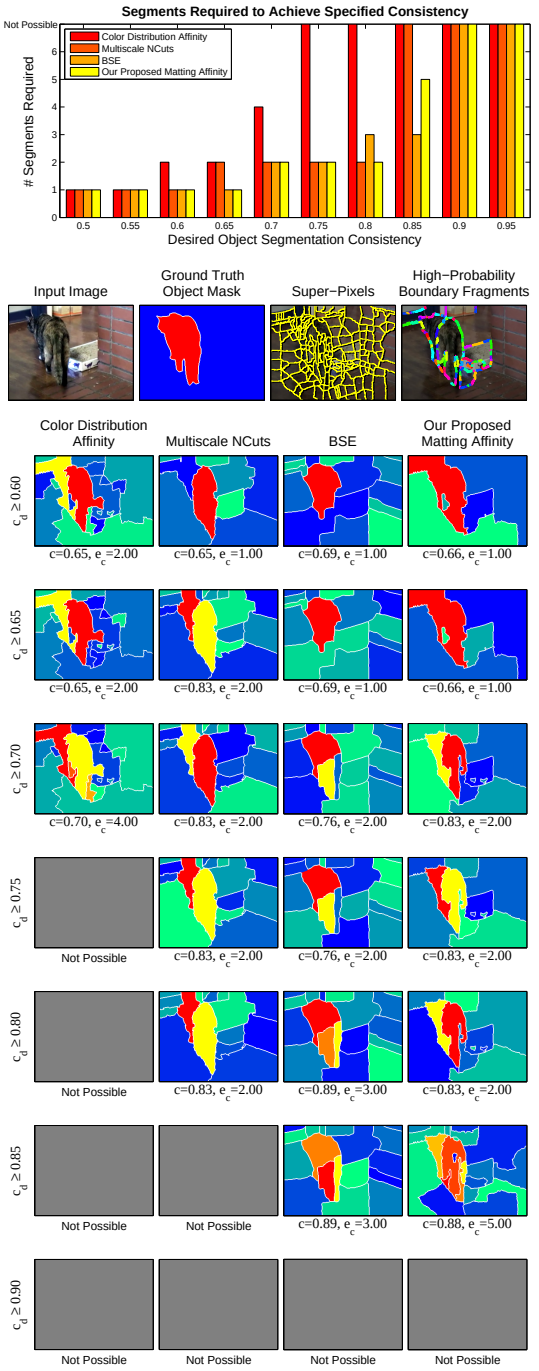


Figure C.38: Cat 2.

References

[1] E. H. Adelson and J. R. Bergen. Spatiotemporal energy models for the perception of motion. *Journal of the Optical Society of America A*, 2(2):284–299, February 1985. 20, 21, 28, 50

[2] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In M. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, chapter 1, pages 3–20. The MIT Press, 1991. 3, 75

[3] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(11):1475–1490, 2004. 25

[4] N. Apostoloff and A. Fitzgibbon. Learning spatiotemporal T-junctions for occlusion detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 553–559, 2005. 5, 10

[5] N. E. Apostoloff and A. W. Fitzgibbon. Bayesian video matting using learnt image priors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. 77, 78

[6] N. E. Apostoloff and A. W. Fitzgibbon. Automatic video segmentation using spatiotemporal T-junctions. In *British Machine Vision Conference (BMVC)*, 2006. 10, 77, 79

[7] P. Arbeláez. Boundary extraction in natural images using ultrametric contour maps. In *IEEE Computer Society Workshop on Perceptual Organization in Computer Vision (POCV)*, 2006. 55

[8] A. Azran and Z. Ghahramani. Spectral methods for automatic multiscale data clustering. In *CVPR*, pages 190–197, 2006. 83

[9] X. Bai and G. Sapiro. A geodesic framework for fast interactive image and video segmentation and matting. In *IEEE International Conference on Computer Vision (ICCV)*, 2007. 77, 78, 112

[10] H. H. Baker and R. C. Bolles. Generalizing epipolar-plane image analysis on the spatiotemporal surface. *International Journal of Computer Vision (IJCV)*, 3(1):33–49, May 1989. 10

[11] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *International Journal of Computer Vision (IJCV)*, 56(3):221–255, March 2004. 112

[12] S. Baker, S. K. Nayar, and H. Murase. Parametric feature detection. *International Journal of Computer Vision (IJCV)*, 27(1):27–50, 1998. 21

[13] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice Hall, 1982. 27

[14] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision (IJCV)*, 12(1):47–77, 1994. 34, 37

[15] A. Baumberg. Reliable feature matching across widely separated views. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 774–781, 2000. 3, 95, 96

[16] P. Bayerl and H. Neumann. Disambiguating visual motion by form-motion interaction – a computational model. *International Journal of Computer Vision (IJCV)*, 72(1):27–45, April 2007. 5

[17] A. E. Beaton and J. W. Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974. 18

[18] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. Technical Report UCB//CSD-04-1366, U.C. Berkeley, December 2004. 3

[19] A. C. Berg and J. Malik. Geometric blur for template matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 607–614, 2001. 3, 95

[20] M. J. Black and D. J. Fleet. Probabilistic detection and tracking of motion discontinuities. *International Journal of Computer Vision (IJCV)*, 38(3):231–245, 2000. 2, 5, 9, 10, 17, 35, 37, 42

[21] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heegar. Robust anisotropic diffusion. *IEEE Transactions on Image Processing*, 7:421–432, 1998. 97

[22] R. Bolles, H. H. Baker, and D. H. Marimont. Epipolar-plane image analysis: An approach to determining structure from motion. *International Journal of Computer Vision (IJCV)*, 1(1):7–56, 1987. 10

[23] P. Bouthemy. A maximum likelihood framework for determining moving edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 11(5):499–511, May 1989. 20

[24] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *IEEE International Conference on Computer Vision (ICCV)*, 2001. 78

[25] L. Breiman. Bagging predicotrs. *Machine Learning*, 24(2):123–140, 1996. 114

[26] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall, 1993. 64

[27] G. Brostow and I. Essa. Motion based decompositing of video. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 8–13, 1999. 8, 9

[28] M. Brown and D. G. Lowe. Invariant features from interest point groups. In *British Machine Vision Conference (BMVC)*, September 2002. 3, 95

[29] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision (ECCV)*, volume 4, pages 25–36, 2004. 50, 51

[30] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 8:679–698, 1986. 5, 43

[31] O. Carmichael and M. Hebert. Shape-based recognition of wiry objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Press, June 2003. 3, 95, 97, 114

[32] Y. Chuang, B. Curless, D. Salesin, and R. Szeliski. A bayesian approach to digital matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001. 77, 78

[33] C. Chubb and G. Sperling. Second-order motion perception: Space/time separable mechanisms. In *Proc. Workshop on Visual Motion*, pages 126–138, March 1989. 5, 50

[34] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Machine Learning*, 48(1-3), 2002. 61

[35] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24(5):603–614, May 2002. 54, 82

[36] T. Cour, F. Bénézit, and J. Shi. Spectral segmentation with multiscale graph decomposition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 79, 82, 85

[37] D. Crispell, D. Lanman, P. G. Sibley, Y. Zhao, and G. Taubin. Beyond silhouettes: Surface reconstruction using multi-flash photography. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, 2006. 10

[38] T. Darrell and A. P. Pentland. Cooperative robust estimation using layers of support. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(5):474–487, 1995. 8

[39] F. de la Torre and M. J. Black. A framework for robust subspace learning. *International Journal of Computer Vision (IJCV)*, 54(1–3):117–142, August–September 2003. 18

[40] K. G. Derpanis and J. M. Gryn. Three-dimensional Nth derivative of gaussian separable steerable filters. In *IEEE International Conference on Image Processing (ICIP)*, volume III, pages 553–556, 2005. 22

[41] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and objects boundaries. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 6

[42] T. Drummond and R. Cipolla. Application of lie algebras to visual servoing. *International Journal of Computer Vision (IJCV)*, 37(1):21–41, 2000. 62

[43] Y. Dufournaud, C. Schmid, and R. Horaud. Matching images with different resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 612–618. IEEE Computer Society Press, June 2000. 95

[44] F. J. Estrada, A. D. Jepson, and C. Chennubhotla. Spectral embedding and min-cut for image segmentation. In *British Machine Vision Conference (BMVC)*, 2004. 80

[45] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision (IJCV)*, 59(2):167–181, 2004. 54

[46] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 264–271, 2003. 3, 25

[47] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 3

[48] R. Feris, R. Raksar, L. Chen, K. Tan, and M. Turk. Discontinuity preserving stereo with small baseline multi-flash illumination. In *IEEE International Conference on Computer Vision (ICCV)*, 2005. 10

[49] V. Ferrari, T. Tuytelaars, and L. V. Gool. Simultaneous object recognition and segmentation by image exploration. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision (ECCV)*, volume LNCS 3021, pages 40–54. Springer-Verlag, 2004. 3, 95

[50] V. Ferrari, T. Tuytelaars, and L. V. Gool. Object detection by contour segment networks. In *European Conference on Computer Vision (ECCV)*, volume III, pages 14–28, 2006. 114

[51] D. J. Fleet, M. J. Black, and O. Nestares. Bayesian inference of visual motion boundaries. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*, pages 139–173. Morgan Kaufmann, July 2002. 5, 9, 10

[52] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision (IJCV)*, 5:77–104, 1990. 34

[53] D. J. Fleet and Y. Weiss. Optical flow estimation. In N. Paragios, Y. Chen, and O. Faugeras, editors, *Mathematical models for Computer Vision: The Handbook*. Springer, 2005. 15, 17, 34, 112

[54] L. Florack, B. ter Haar Romeny, M. Viergever, and J. Koenderink. The gaussian scale-space paradigm and the multiscale local jet. *International Journal of Computer Vision (IJCV)*, 18:61–75, 1996. 95, 96

[55] L. M. J. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, 1992. 95

[56] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: Combining patch-based and gradient-based approaches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 2, 79, 82, 85

[57] W. T. Freeman and E. H. Adelson. The design and user of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(9):891–906, September 1991. 21, 28

[58] B. J. Frey. *Graphical Models for Machine Learning and Digital Communication*. MIT Press, 1998. 60

[59] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2), 2000. 61

[60] B. Funt, K. Barnard, and L. Martin. Is color constancy good enough? In *European Conference on Computer Vision (ECCV)*, pages 445–459, 1998. 103

[61] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 858–863, June 1997. 2, 9

[62] D. Geiger, K. Kumaran, and L. Parida. Visual organization for figure/ground separation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 155–160, 1996. 10

[63] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 6(6):721–741, November 1984. 2

[64] K. Grauman, G. Shakhnarovich, and T. Darrell. A bayesian approach to image-based visual hull reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. 10, 11

[65] L. Guan, J.-S. Franco, and M. Pollefeys. 3D occlusion inference from silhouette cues. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 8, 11

[66] A. Guzman. Decomposition of a visual scene into three dimensional bodies. In *AFIPS Fall Joint Conference*, volume 33, pages 291–304, 1968. 60

[67] D. J. Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America A*, 4(8):1455–1471, August 1987. 50

[68] D. J. Heeger. Optical flow using spatiotemporal filters. *International Journal of Computer Vision (IJCV)*, 1:270–302, 1988. 20, 21, 28

[69] F. Heitger and R. von der Heydt. A computational model of neural contour processing. In *IEEE International Conference on Computer Vision (ICCV)*, pages 32–40, 1993. 10

[70] F. Heitz and P. Bouthemy. Multimodal estimation of discontinuous optical flow using markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(12):1217–1232, December 1993. 2, 9, 15, 18

[71] L. Herault and R. Horaud. Figure-ground discrimination: A combinatorial optimization approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(9):899–914, September 1993. 5

[72] T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimization. In *Uncertainty in Artificial Intelligence (UAI)*, pages 313–320, 2003. 60

[73] H. Hirschmüller, P. R. Innocent, and J. Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *International Journal of Computer Vision (IJCV)*, 47(1-3):229–246, April-June 2002. 2, 9, 16

[74] D. Hoiem, A. A. Efros, and M. Hebert. Automatic photo pop-up. *ACM Transactions on Graphics (SIGGRAPH)*, 24(3):577–584, 2005. 56

[75] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision (IJCV)*, 75(1), October 2007. 61, 75

[76] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering occlusion boundaries from a single image. In *IEEE International Conference on Computer Vision (ICCV)*, 2007. 9, 13, 56, 61, 74, 113

[77] P. W. Holland and R. E. Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics*, A6:813–827, 1977. 18

[78] M. Irani and S. Peleg. Motion analysis for image enhancement: Resolution, occlusion, and transparency. *Journal of Visual Communication and Image Representation*, 4(4):324–335, December 1993. 8

[79] A. D. Jepson, D. J. Fleet, and M. J. Black. A layered motion representation with occlusion and compact spatial support. In *European Conference on Computer Vision (ECCV)*, volume 1, pages 692–706, 2002. 8

[80] N. Jojic and B. J. Frey. Learning flexible sprites in video layers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 196–206, 2001. 8

[81] B. Julesz. Textons, the elements of texture perception and their interactions. *Nature*, 290:91–97, 1981. 43

[82] F. Jurie and C. Schmid. Scale-invariant shape features for recognition of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. 3, 97, 114

[83] T. Kadir and M. Brady. Saliency, scale and image description. *International Journal of Computer Vision (IJCV)*, 45(2):83–105, 2001. 95

[84] T. Kanade and M. Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 16(9):920–932, September 1994. 2, 9, 16

[85] G. A. Kaplan. Kinetic disruption of optical texture: The perception of depth at an edge. *Perception and Psychophysics*, 6:193–198, 1969. 5

[86] Q. Ke and T. Kanade. A robust subspace approach to layer extraction. In *IEEE Workshop on Motion and Video Computing (MOTION)*, pages 37–43, December 2002. 8

[87] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. 3, 95

[88] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *IEEE International Conference on Computer Vision (ICCV)*, October 2005. 25

[89] P. K. Kienker, T. J. Sejnowski, G. E. Hinton, and L. E. Schumacher. Separating figure from ground with a parallel network. *Perception*, 15:197–216, 1986. 10

[90] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(1):57–74, January 2003. 6, 43, 54

[91] M. P. Kumar, P. Torr, and A. Zisserman. Learning layered motion segmentations of video. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 33–40, 2005. 8

[92] S. Kumar and M. Hebert. Discriminative random fields. *International Journal of Computer Vision (IJCV)*, 68(2):179–202, 2006. 2, 114

[93] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *International Conference on Machine Learning (ICML)*, 2001. 2

[94] I. Laptev and T. Lindeberg. Space-time interest points. In *IEEE International Conference on Computer Vision (ICCV)*, 2003. 25

[95] S. Lazebnik and J. Ponce. The local projective shape of smooth surfaces and their outlines. *International Journal of Computer Vision (IJCV)*, 63(1):65–83, 2005. 9, 35, 114

[96] E. G. Learned-Miller. Data driven image models through continuous joint alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(2):236–250, February 2006. 112

[97] B. Leibe and B. Schiele. Analyzing appearance and contour based methods for object categorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2003. 103

[98] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *IEEE International Conference on Computer Vision (ICCV)*, 2005. 20

[99] T. Leung and J. Malik. Contour continuity in region based image segmentation. In *European Conference on Computer Vision (ECCV)*, June 1998. 65, 79, 82

[100] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. iii, 77, 78, 79, 112

[101] E. Levina and P. Bickel. The earth mover's distance is the Mallows distance: Some insights from statistics. In *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 251–256, 2001. 23

[102] G. Li. Robust regression. In D. C. Hoaglin, F. Mosteller, and J. W. Tukey, editors, *Exploring Data, Tables, Trends and Shapes*. John Wiley and Sons, 1985. 18

[103] T. Lindeberg and J. Garding. Shape-adapted smoothing in estimation of 3-D depth cues from affine distortions of local 2-D brightness structure. In *European Conference on Computer Vision (ECCV)*, pages 389–400, May 1994. 95

[104] C. Liu, W. T. Freeman, and E. H. Adelson. Analysis of contour motions. In *Advances in Neural Information Processing Systems (NIPS)*, 2006. 54

[105] D. G. Lowe. Local feature view clustering for 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, December 2001. Kauai, Hawaii. 3, 95

[106] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, January 2004. 3, 13, 25, 95, 97, 98, 99, 103, 104

[107] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 674–679, 1981. 15

[108] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 60

[109] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(4):433–444, April 2003. 5, 65

[110] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *British Machine Vision Conference (BMVC)*, 2007. 75

[111] C. Mallows. A note on asympototic joint normality. *Annals of Mathematical Statistics*, 43(2):508–515, April 1972. 23

[112] D. Marr. *Vision*. W. H. Freeman, 1982. 76

[113] 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 *IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 416–423, July 2001. 11, 44

[114] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(5):530–549, May 2004. 5, 6, 23, 24, 25, 28, 30, 43, 45, 54, 82, 85

[115] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. *ACM Transactions on Graphics (SIGGRAPH)*, 2000. 10, 11

[116] B. A. Maxwell and S. J. Brubaker. Texture edge detection using the compass operator. In *British Machine Vision Conference (BMVC)*, volume II, pages 549–558, September 2003. 5, 23, 43

[117] J. McDermott and E. H. Adelson. The geometry of the occluding contour and its effect on motion interpretation. *Journal of Vision*, 4(10):944–954, 2004. 5

[118] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *IEEE International Conference on Computer Vision (ICCV)*, pages 525–531, 2001. 95, 96

[119] K. Mikolajczyk and C. Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision (ECCV)*, pages 128–142. Springer, 2002. Copenhagen. 96

[120] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, June 2003. 3, 95

[121] K. Mikolajczyk, A. Zisserman, and C. Schmid. Shape recognition with edge-based features. In *British Machine Vision Conference (BMVC)*, 2003. 3, 95, 97, 114

[122] G. Mori. Guiding model search using segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, 2005. 55

[123] G. Mori, X. Ren, A. Efros, and J. Malik. Recovering human body configurations: Combining segmentation and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 3226–333, 2004. 55

[124] E. N. Mortensen, H. Deng, and L. Shapiro. A sift descriptor with global context. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 3, 95

[125] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (COIL-100). Technical report, Columbia University, February 1996. 103

[126] O. Nestares and D. J. Fleet. Probabilistic tracking of motion boundaries with spatiotemporal predictions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 358–365, 2001. 5, 9, 10

[127] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, 2001. 82

[128] A. S. Ogale, C. Fermüller, and Y. Aloimonos. Motion segmentation using occlusions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 27(6):988–992, June 2005. 8

[129] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *European Conference on Computer Vision (ECCV)*, volume II, pages 575–588, 2006. 114

[130] J. Pearl. Reverend bayes on inference engines: A distributed hierarchical approach. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 133–136, 1982. 60

[131] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988. 60

[132] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks, and roofs. In *IEEE International Conference on Computer Vision (ICCV)*, pages 52–57, 1990. 21, 28, 40, 43

[133] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(7):629–639, July 1990. 97

[134] M. A. Peterson and B. S. Gibson. Must figure-ground organization precede object recognition? An assumption in peril. *Psychological Science*, 5(5):253–259, September 1994. 90

[135] F. T. Qiu and R. von der Heydt. Figure and ground in the visual cortex: V2 combines stereoscopic cues with gestalt rules. *Neurons*, 47:155–166, July 2005. 5, 90

[136] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2004. 3

[137] A. Rabinovich, A. Vedaldi, and S. Belongie. Does image segmentation improve object categorization? Cs2007-0908, University of California San Diego, 2007. 75

[138] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics (SIGGRAPH)*, 23:679–688, 2004. 10, 11

[139] X. Ren, C. C. Fowlkes, and J. Malik. Cue integration for figure/ground labeling. In *Advances in Neural Information Processing Systems (NIPS)*, 2005. 54, 65, 114

[140] X. Ren, C. C. Fowlkes, and J. Malik. Figure/ground assignment in natural images. In *European Conference on Computer Vision (ECCV)*, 2006. 5, 7, 90, 114

[141] X. Ren and J. Malik. Learning a classification model for segmentation. In *IEEE International Conference on Computer Vision (ICCV)*, volume 1, pages 10–17, 2003. 54, 55, 79

[142] M. G. Ross and L. P. Kaelbling. Learning static object segmentation from motion segmentation. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2005. 5, 9

[143] C. Rother, V. Kolmogorov, and A. Blake. "grabCut": Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (SIGGRAPH)*, 23(3):309–314, 2004. 78

[144] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, 1987. 19

[145] E. Rubin. Visuell wahrgenommene figuren. In *Kobenhaven: Glydenalske boghandel*, 1921. 5, 76

[146] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 75, 114

[147] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. Memo AIM-2005-025, MIT AI Lab, September 2005. http://labelme.csail.mit.edu/. 1

[148] M. Ruzon and C. Tomasi. Color edge detection with the compass operator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 160–166, June 1999. 5, 23, 43

[149] M. Ruzon and C. Tomasi. Alpha estimation in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000. 77, 78

[150] J. Sato and R. Cipolla. Affine reconstruction of curved surfaces from uncalibrated views of apparent contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 21(11):1188–1197, 1999. 9, 35, 114

[151] E. Saund. Perceptual organization of occluding contours of opaque surfaces. *Computer Vision and Image Understanding (CVIU)*, Special Issue on Perceptual Organization:70–82, 1999. 10

[152] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 19(5):530–534, May 1997. 96

[153] A. Sethi, D. Renaudie, D. Kriegman, and J. Ponce. Curve and surface duals and the recognition of curved 3d objects from their silhouettes. *International Journal of Computer Vision (IJCV)*, 58(1):73–86, 2004. 3, 9, 35, 114

[154] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 2nd edition, 1999. 101, 112

[155] Y. Shan, H. S. Sawhney, B. Matei, and R. Kumar. Partial object matching with shapeme histograms. In *European Conference on Computer Vision (ECCV)*, 2004. 114

[156] E. Shechtman and M. Irani. Space-time behavior based correlation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 405–412, 2005. 15, 25, 37, 40, 43

[157] Y. A. Sheikh, E. A. Khan, and T. Kanade. Mode-seeking via medoidshifts. In *IEEE International Conference on Computer Vision (ICCV)*, 2007. 82

[158] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1154–1160, 1998. 8, 91

[159] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 22(8):888–905, August 2000. 77, 81

[160] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994. 63

[161] E. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optical flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991. 37

[162] P. Smith, T. Drummond, and R. Cipolla. Layered motion segmentation and depth ordering by tracking edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(4):479–494, April 2004. 8, 9, 10, 20, 54

[163] P. A. Smith. *Edge-based Motion Segmentation*. PhD thesis, Jesus College, University of Cambridge, August 2001. 2

[164] L. Spillmann and W. H. Ehrenstein. Gestalt factors in the visual neurosciences. In L. M. Chalupa and J. S. Werner, editors, *The Visual Neurosciences*, pages 1573–1589. MIT Press, Cambridge, MA, November 2003. 5, 90

[165] A. Stein and M. Hebert. Incorporating background invariance into feature-based object recognition. In *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 37–44, 2005. 4, 13, 97

[166] A. Stein and M. Hebert. Combining local appearance and motion cues for occlusion boundary detection. In *British Machine Vision Conference (BMVC)*, 2007. 12, 16

[167] A. Stein, D. Hoiem, and M. Hebert. Learning to find object boundaries using motion cues. In *IEEE International Conference on Computer Vision (ICCV)*, 2007. 9, 13, 61

[168] A. N. Stein and M. Hebert. Local detection of occlusion boundaries in video. In *British Machine Vision Conference (BMVC)*, pages 407–416, 2006. 9, 12, 16, 36

[169] A. N. Stein and M. Hebert. Using spatio-temporal patches for simultaneous estimation of edge strength, orientation, and motion. In *Beyond Patches Workshop at IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 19, 2006. 12

[170] A. N. Stein, T. S. Stepleton, and M. Hebert. Towards unsupervised whole-object segmentation: Combining automated matting with boundary detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. 13

[171] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum. Poisson matting. *ACM Transactions on Graphics (SIGGRAPH)*, 23(3):315–321, 2004. 77, 78

[172] J. Tang and P. Lewis. Using multiple segmentations for image auto-annotation. In *ACM International Conference on Image and Video Retrieval*, 2007. 75

[173] W. B. Thompson, K. M. Mutch, and V. A. Berzins. Dynamic occlusion analysis in optical flow fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 7:374–383, 1985. 5

[174] D. A. Tolliver and G. L. Miller. Graph partitioning by spectral rounding: Applications in image segmentation and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 82

[175] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, April 1991. 15, 20

[176] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, 5(7):1–6, 2002. 25, 95, 98

[177] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(6):929–944, June 2007. 75

[178] R. Vaillant and O. D. Faugeras. Using extremal boundaries for 3-D object modeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):157–173, 1992. 9, 35, 114

[179] T. Veit, F. Cao, and P. Bouthemy. An a contrario decision framework for region-based motion detection. *International Journal of Computer Vision (IJCV)*, 68(2):163–178, June 2006. 5, 9

[180] D. A. Waltz. Understanding line drawings of scenes with shadows. In *The Psychology of Computer Vision*, pages 19–91. McGraw-Hill, 1975. 60

[181] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994. 8

[182] A. M. Waxman, J. Wu, and F. Bergholm. Convected activation profiles and receptive fields for real time measurement of short range visual motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 717–723, 1988. 20, 62

[183] Y. Weiss. Interpreting images by propagating bayesian beliefs. In *Advances in Neural Information Processing Systems*, volume 9, page 908, 1997. 62

[184] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 520–526, 1997. 8

[185] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural Computation*, 12(1):1–41, 2000. 60

[186] L. Wolf, X. Huang, I. Martin, and D. Metaxas. Patch-based texture edges and segmentation. In *European Conference on Computer Vision (ECCV)*, pages 481–493, 2006. 5, 23, 43

[187] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi. Bilateral filtering-based optical flow estimation with occlusion detection. In *European Conference on Computer Vision (ECCV)*, volume I, pages 211–224, 2006. 17

[188] J. Xiao and M. Shah. Accurate motion layer segmentation and matting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 8, 91

[189] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, July 2005. 60

[190] P. Yin, A. Criminisi, J. Winn, and I. Essa. Tree-based classifiers for bilayer video segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007. 8

[191] A. Yonas, L. G. Craton, and W. B. Thompson. Relative motion: Kinetic information for the order of depth at an edge. *Perception and Psychophysics*, 41:53–59, 1987. 5

[192] Y. L. You, W. Xu, A. Tannenbaum, and M. Kaveh. Behavior analysis of anisotropic diffusion in image processing. *IEEE Transactions on Image Processing*, 5:1539–1553, 1996. 97

[193] S. Yu, T.-S. Lee, and T. Kanade. A hierarchical markov random field model for figure-ground segregation. In *Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR)*, 2001. 10

[194] S. Yu and J. Shi. Segmentation with pairwise attraction and repulsion. In *IEEE International Conference on Computer Vision (ICCV)*, July 2001. 5, 91

[195] S. X. Yu and J. Shi. Perceiving shapes through region and boundary interaction. Technical Report CMU-RI-TR-01-21, Robotics Institute, Carnegie Mellon University, July 2001. 9

[196] S. X. Yu and J. Shi. Multiclass spectral clustering. In *IEEE International Conference on Computer Vision (ICCV)*, 2003. 82

[197] A. L. Yuille. CCCP algorithms to minimize the Bethe and Kikuchi free energies: convergent alternatives to belief propagation. *Neural Computation*, 14(7):1691–1722, July 2002. 60

[198] Z. Zhang. Parameter estimation techniques: A tutorial with application to conic fitting. *Image and Vision Computing (IVC)*, 15(1):59–76, 1997. 19

[199] S.-C. Zhu, C. en Guo, Y. Wu, and Y. Wang. What are textons? In *European Conference on Computer Vision (ECCV)*, 2002. 43