The Dissertation Committee for Jaechul Kim
certifies that this is the approved version of the following dissertation:

Region Detection and Matching for Object Recognition

Committee:

Kristen Grauman, Supervisor

J. K. Aggarwal

Charless Fowlkes

Pradeep Ravikumar

Peter Stone

Region Detection and Matching for Object Recognition

by

Jaechul Kim, B.S.E, M.S.E.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2013

Region Detection and Matching for Object Recognition

Jaechul Kim, Ph.D.
The University of Texas at Austin, 2013

Supervisor: Kristen Grauman

In this thesis, I explore region detection and consider its impact on image matching for exemplar-based object recognition. Detecting regions is important to provide semantically meaningful spatial cues in images. Matching establishes similarity between visual entities, which is crucial for recognition. My thesis starts by detecting regions in both local and object level. Then, I leverage geometric cues of the detected regions to improve image matching for the ultimate goal of object recognition. More specifically, my thesis considers four key questions: 1) how can we extract distinctively-shaped local regions that also ensure repeatability for robust matching? 2) how can object-level shape inform bottom-up image segmentation? 3) how should the spatial layout imposed by segmented regions influence image matching for exemplar-based recognition? and 4) how can we exploit regions to improve the accuracy and speed of dense image matching? I propose novel algorithms to tackle these issues, addressing region-based visual perception from low-level local region extraction, to mid-level object segmentation, to high-level region-based matching and recognition.

First, I propose a Boundary Preserving Local Region (BPLR) detector to extract local shapes. My approach defines a novel spanning-tree based image representation whose structure reflects shape cues combined from multiple segmentations, which in turn provide multiple initial hypotheses of the object boundaries. Unlike traditional local region detectors that rely on *local* cues like color and texture, BPLRs explicitly exploit the segmentation that encodes *global*

object shape. Thus, they respect object boundaries more robustly and reduce noisy regions that straddle object boundaries. The resulting detector yields a dense set of local regions that are both distinctive in shape as well as repeatable for robust matching.

Second, building on the strength of the BPLR regions, I develop an approach for object-level segmentation. The key insight of the approach is that objects shapes are (at least partially) shared among different object categories—for example, among different animals, among different vehicles, or even among seemingly different objects. This *shape sharing* phenomenon allows us to use *partial* shape matching via BPLR-detected regions to predict *global* object shape of possibly unfamiliar objects in new images. Unlike existing top-down methods, my approach requires no category-specific knowledge on the object to be segmented. In addition, because it relies on exemplar-based matching to generate shape hypotheses, my approach overcomes the viewpoint sensitivity of existing methods by allowing shape exemplars to span arbitrary poses and classes.

For the ultimate goal of region-based recognition, not only is it important to detect good regions, but we must also be able to match them reliably. A matching establishes similarity between visual entities (images, objects or scenes), which is fundamental for visual recognition. Thus, in the third major component of this thesis, I explore how to leverage geometric cues of the segmented regions for accurate image matching. To this end, I propose a segmentation-guided local feature matching strategy, in which segmentation suggests spatial layout among the matched local features within each region. To encode such spatial structures, I devise a string representation whose 1D nature enables efficient computation to enforce geometric constraints. The method is applied for exemplar-based object classification to demonstrate the impact of my segmentation-driven matching approach.

Finally, building on the idea of regions for geometric regularization in image matching, I consider how a hierarchy of nested image regions can be used to constrain dense image feature matches at multiple scales simultaneously. Moving beyond individual regions, the last part of

my thesis studies how to exploit regions' inherent hierarchical structure to improve the image matching. To this end, I propose a deformable spatial pyramid graphical model for image matching. The proposed model considers multiple spatial extents at once—from an entire image to grid cells to every single pixel. The proposed pyramid model strikes a balance between robust regularization by larger spatial supports on the one hand and accurate localization by finer regions on the other. Further, the pyramid model is suitable for fast coarse-to-fine hierarchical optimization. I apply the method to pixel label transfer tasks for semantic image segmentation, improving upon the state-of-the-art in both accuracy and speed.

Throughout, I provide extensive evaluations on challenging benchmark datasets, validating the effectiveness of my approach. In contrast to traditional texture-based object recognition, my region-based approach enables to use strong geometric cues such as shape and spatial layout that advance the state-of-the-art of object recognition. Also, I show that regions' inherent hierarchical structure allows fast image matching for scalable recognition. The outcome realizes the promising potential of region-based visual perception. In addition, all my codes for local shape detector, object segmentation, and image matching are publicly available, which I hope will serve as useful new additions for vision researchers' toolbox.

Table of Contents

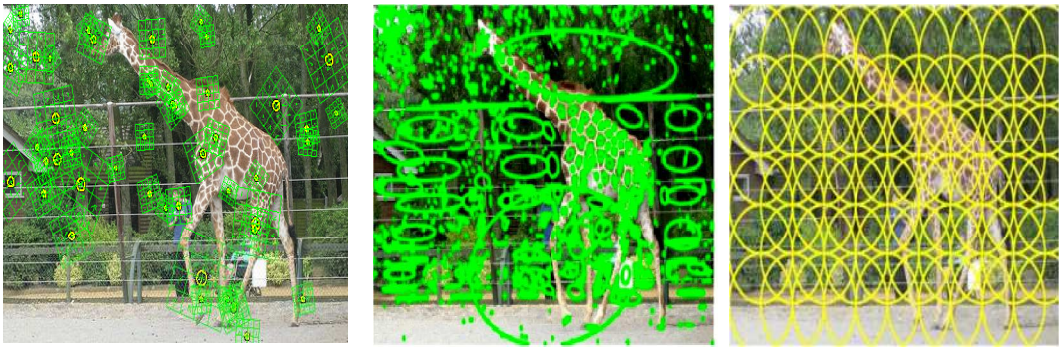
Chapter 1

Introduction

Detecting meaningful regions in an image is a long-standing research topic in computer vision. Regions of various spatial coverages, ranging from corner or junction points, to local blobs of salient texture patterns, to segments that cover entire objects, have been considered. In practice, regions serve as basic building blocks for many vision applications. For example, a local interest region detector, due to its robustness to image variations, is crucial for image matching and object recognition; an object segmentation is important for object localization and image parsing.

Local features—image regions of locally salient appearance patterns—have made a great contribution to the success of object recognition and image retrieval tasks. Their locality offers robustness to occlusions and deformation, and when extracted densely and/or at multiple scales they capture rich statistics for learning-based recognition algorithms. Local features are mostly extracted at the locations of salient local texture/intensity patterns, but their shapes are often fixed like circle, ellipse, or rectangle. Fixing features' shapes reduces the complexity of feature detection. The lack of shape, however, weakens the representational power of the extracted features and produces noisy features straddling object boundaries (see Figure 1.1(a)).

Extracting regions with varying shapes—I will call them simply "regions" unless confusion occurs—has long been studied in computer vision, mostly in the field of image segmentation. A region-based approach is appealing since it aims to capture shapes of objects' parts and whole, which provide semantically meaningful spatial support for recognition. In practice, however, the instability of segmentation with respect to image variations can make the extracted regions unreliable or sensitive to parameter settings (see Figure 1.1(b)). Multiple segmentation



(a) Local regions

(b) Segmented regions

Figure 1.1: Illustration of region detection. **(a)** We show local regions widely used in computer vision—SIFT [64], MSER [68], and dense sampling. They are extracted in fixed shapes—rectangular patches, ellipses, or circles, and often straddle object boundaries. **(b)** Segmented regions capture objects' shape; but they are unstable under image variations, making them hard to match.

strategies, by varying parameters or merging adjacent regions, enlarge the pool of segments to increase the chance of hitting a true object (e.g., [67, 34]); however, large pools of regions incur redundancy of noisy segments, and, more importantly, existing methods lack a model of true object shapes. Top-down methods instead use the prior knowledge on the shapes of objects to extract reliable segments [12, 58, 95, 20, 52]; however, they require category-specific knowledge on the object, making them less applicable.

In this thesis, I explore region detection and consider its impact on image matching for exemplar-based object recognition. I address four key questions: 1) how can we extract

distinctively-shaped local regions that also ensure repeatability for robust matching? 2) how can object-level shape inform bottom-up image segmentation? 3) how should the spatial layout imposed by segmented regions influence image matching for exemplar-based recognition? and 4) how can we exploit a hierarchical structure of different levels of local- and object-regions to improve the accuracy and speed of image matching? I propose novel algorithms to tackle these issues, addressing region-based visual perception from low-level local shape extraction, to mid-level object segmentation, to high-level region-based matching and recognition.

In particular, I first develop approaches to extract regions at both the local- and object-level, incorporating shapes in a generic way without requiring any category-specific knowledge. Those local- and object-level regions are then brought together into a novel approach to image matching for region-based object recognition in a data-driven, exemplar-based manner. To improve the scalability of exemplar-based recognition, I further explore a fast matching method that exploits the hierarchical structure of regions across various spatial extents.

In the following sections, I will overview each of four major components (local shape detection, object segmentation, segmentation-driven matching and recognition, fast image matching on region hierarchy) of my approach. Chapter 3 through 6 then give more detail on these ideas and present my results.

Local Shape Detection Researchers have developed a variety of techniques to detect local regions, ranging from sophisticated interest point operators [68, 69, 70, 43, 92] to dense sampling strategies [72]. While by design such methods provide highly repeatable detections across images, their low-level local sampling criteria generate many descriptors that straddle object boundaries and lack distinctive shapes. The first component of my work aims to create a detector for features that capture distinctive local shapes within the image and are also repeatable under image variations across images.

To extract local shapes, I propose a Boundary Preserving Local Region (BPLR) detec-

tor [47]. BPLRs are local shape features designed to preserve object boundaries derived from multiple segmentations. My approach defines a novel spanning-tree based image representation whose structure reflects shape cues combined from multiple segmentations, which in turn provide multiple initial hypotheses of the object boundaries. Unlike traditional local region detectors that rely on *local* cues such as colors and textures, the BPLR detector explicitly exploits the segmentation that encodes *global* object shape, thereby respecting object boundaries and capturing local object shapes. At the same time, it is robust to the parameter sensitivity of typical segmentation as it combines multiple segmentation hypotheses. In addition, BPLRs are densely extracted from an image, and thus retain rich statistics of visual information over the entire image that are critical for recognition and matching.

The resulting detector yields a dense set of local regions that are both distinctive in shape as well as repeatable for robust matching. Extensive evaluations on challenging benchmark datasets (Chapter 3) show the proposed BPLR detector provides significantly better repeatability and localization accuracy for matching compared to an array of existing local feature detectors.

Object Segmentation with Shapes Building on the strength of the BPLR regions, I develop an approach for object-level segmentation. Typically, a bottom-up segmentation that relies on local cues suffers from over- or under-segmentation for object-level delineation [22, 28, 3]. Pitfalls include the fact that a single object is often comprised of heterogeneous textures and colors (over-segmentation), and objects with similar appearance can appear adjacent to one another (under-segmentation). To delineate an object correctly, we need a *top-down* shape cue that can bind together an object's parts of diverse appearance or separate objects of similar appearance that would not be possible if judging color/texture/contour alone. Top-down segmentation methods [12, 58, 95, 20, 52, 56, 17, 65] elegantly integrate top-down familiar shapes with bottom-up local cues to obtain object-level segmentation. However, those top-down segmentation methods work in a category-specific manner: they assume to know what objects (or what category of the objects) are to be segmented.

I instead propose a new form of shape priors that enables *category-independent* object segmentation [48]. The key insight of the approach is that objects' shapes are (at least partially) shared among different object categories: many objects exhibit partial shape agreement—and this *shape sharing* occurs even across seemingly disparate categories.

We use this intuition to segment objects of possibly unknown categories, proposing a non-parametric, exemplar-based shape prior. Given a novel unsegmented image, we first identify any strong BPLR matches it has with local shapes in a database of segmented exemplars. Based on the scale and position of each local shape match, we project the associated exemplar shapes into the test image. This effectively maps local support into global shape hypotheses without assuming any category-specific knowledge, since the database need *not* contain exemplars of the same object class(es) as our test image. The projected shapes yield shape priors; we perform a series of figure-ground segmentations using graph-cuts, enforcing each of the shape priors in turn. This finally generates multiple hypotheses of object segments.

Experiments on challenging datasets (Chapter 4) show that 1) shape sharing improves the quality of bottom-up segmentation, and 2) my category-independent shape prior performs as well as a parallel category-specific one, demonstrating the impact of the shape sharing. As such, unlike previous top-down segmentation methods, my approach can enhance the segmentation of previously unseen objects, making it applicable to a wider range of problems.

Segmentation-Driven Image Matching For the ultimate goal of region-based recognition, not only is it important to detect good regions, but we must also be able to match them reliably and efficiently. A matching establishes similarity between visual entities (images, objects or scenes), which is fundamental for visual recognition. Thus, in the third major component of this thesis, I bring together the above ideas for local- and object-level region detection in a novel approach to region-based image matching.

Due to confusing appearance or background clutter, one often obtains noisy correspon-

dences when matching regions individually. To resolve such ambiguity, additional geometric constraints are typically considered for robust matching [64, 32, 44, 76, 57, 9]. In a constrained setting of instance-level matching such as wide-baseline stereo image matching [64, 32, 44, 76], it is straightforward to address the spatial layout in a simple parametric form. In a generic setting that match images of different scenes or objects of generic categories, on the other hand, there is no single parametric form to represent the geometric deformations across images, and thus non-parametric approaches that identify a group of matches having minimal geometric distortion are preferred [57, 9]. However, existing non-parametric methods are limited to a uniform deformation model that is not fitting to matching non-rigid objects and/or too computationally costly for matching a large number of features.

To address those challenges, I propose a segmentation-driven local feature matching strategy, in which segmented regions in the image control the spatial layout of the matched features [46]. I apply my matching method to exemplar-based object recognition using nearest-neighbor classification. One of the key aspects in my approach is that each segment in an image is represented by a 1D string that links the local features extracted within the segment. This string representation enables a dynamic programming formulation to solve the matching problem efficiently, while encoding spatial layout among features. Another key aspect is that my method uses segmented regions to allow non-parametric geometric deformations among the matched features: it imposes different penalties for the geometric distortion for matching features *within* a segment and those *across* the segments. The intuition is that feature correspondences within a segment tend to have more consistent spatial layout than those across the segments, addressing non-rigid deformations of different regions across the images.

I apply the proposed method for exemplar-based object classification. Experimental results on standard benchmark datasets (Chapter 5) show that the proposed method outperforms existing matching-based recognition methods by a significant margin with much less computation, demonstrating the impact of our segmentation-driven matching strategy coupled with an

efficient 1D string representation.

Fast Image Matching with Region Hierarchy Regions can provide geometric constraints for image matching as shown in the previous section. Building on this idea of using regions for geometric regularization, the last part of my thesis considers how to exploit regions' inherent hierarchical structure (e.g., a region of an entire object and its parts) to improve the image matching. Particularly, I address fast dense pixel matching method that builds on a hierarchy of nested regions for scalable exemplar-based object recognition.

Dense pixel matching aims to find correspondences of "every pixel" between two images. With the steady advance of dense matching quality, researchers introduce many interesting applications of dense pixel matching for vision and graphics, including semantic image segmentation [62], image completion [6], and video depth estimation [45]. As already noted, however, when matching images, we face two major challenges: image variation and computational cost. These challenges become much more severe when we address dense pixel matching, due to 1) pixel's locality that lacks the discriminating power to resolve matching ambiguity in the face of visual variations, and 2) huge problem size that we must handle each of millions of pixels between images.

To address these challenges, I propose a deformable spatial pyramid graphical model [49]. Unlike the conventional approach that builds on a flat pixel-level matching objective [62, 6, 46, 57, 23], the proposed model considers the match at multiple spatial extents in a hierarchical way—ranging from an entire image, to coarse grid cells, to every single pixel. The proposed pyramid model strikes a balance between robustness to match ambiguities on the one hand, and accurate localization of pixel correspondences on the other, leading to better matching accuracy. Larger spatial nodes offer greater regularization when appearance matches are ambiguous, while smaller spatial nodes help localize matches with fine detail. Further, the proposed pyramid model is naturally suited for fast coarse-to-fine optimization, which substantially improves

the run-time over existing methods, matching hundreds of thousands of pixels between images within a fraction of second.

I apply the method to exemplar-based semantic image segmentation. Extensive experiments on challenging benchmark datasets show that 1) various spatial supports by our spatial pyramid improve matching quality, striking a balance between geometric regularization and accurate localization, 2) our pyramid structure permits efficient hierarchical optimization, enabling fast dense matching. As such, my approach achieves substantial gains in both matching accuracy and speed over the today's most popular methods for dense matching, SIFT Flow and PatchMatch [62, 6].

Summary In this thesis, I explore key aspects of region-based objection recognition, starting from region detection at both local and object level and bringing them together into the novel framework of region-based image matching for the ultimate goal of object recognition. Throughout, I provide extensive evaluations on challenging benchmark datasets, validating the effectiveness of my approach. Figure 1.2 summarizes the organization of the thesis.

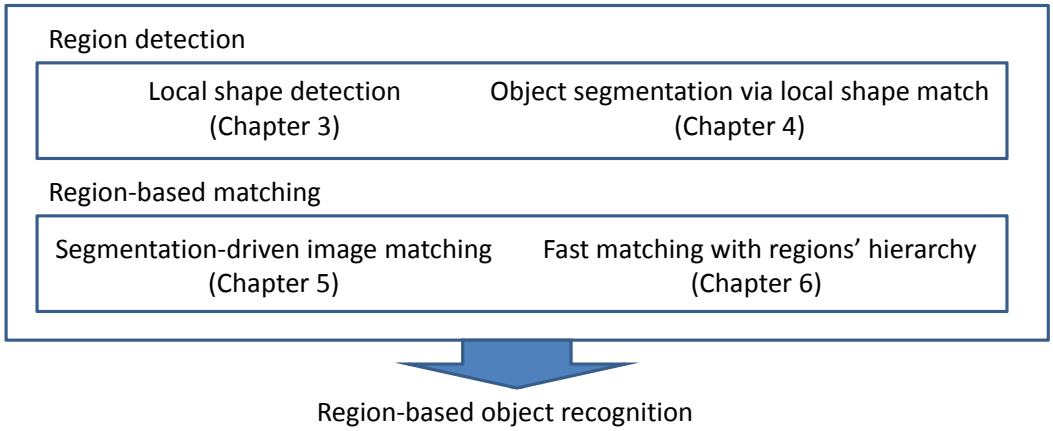


Figure 1.2: Organization of the thesis.

The main contributions of my work are:

- I propose a **Boundary Preserving Local Region detector (BPLR)** to capture local

shapes (Chapter 3). By combining multiple segmentation hypotheses via a novel spanning-tree based geometry representation, the BPLR detector respects object boundaries, robustly capturing object local shape under image variations.

- I introduce a **category-independent shape prior** for object segmentation that exploits shape sharing between the objects of different categories (Chapter 4). As such, unlike typical top-down segmentation methods, my approach can enhance the segmentation of previously unseen objects.

- I propose a **segmentation-driven image matching** that considers the spatial layout among matched features suggested by segmentation. The proposed method introduces a novel 1D image representation that effectively captures each region's geometric layout with an efficient dynamic programming solution (Chapter 5).

- I propose a **deformable spatial pyramid model** that exploits a hierarchical structure of a group of regions for a fast dense pixel matching. This improves upon the state-of-the-art in both accuracy and speed, enabling scalable exemplar-based recognition (Chapter 6).

- Putting all these components together, I achieve **region-based object recognition in an exemplar-based** way. The detected local- and object-level regions are integrated into a matching framework for efficient exemplar-based object recognition, demonstrating its impact on various vision tasks from low-level region detection, to mid-level object segmentation, to high-level object classification and semantic image parsing.

Overall, my thesis realizes the promising potential of region-based visual perception. Key issues of region-based recognition, including region detection, matching, and their applications to object recognition, are all addressed in an integrated way. I test the methods on challenging benchmark datasets, showing the proposed methods improve upon the state-of-the-art. All the codes are publicly released for future research toward region-based object recognition.

Chapter 2

Related Work and Background

In this chapter, I will review the literature according to each category of my work: local region detection, object segmentation, region-based image matching, and fast image matching for exemplar-based recognition.

2.1 Local Region Detection

Local features—image regions of locally salient appearance patterns—are a basic building block for image retrieval and recognition tasks. The general feature extraction pipeline consists of (a) a detection stage, which selects the image sites (positions, scales, shapes) where features will be extracted, and (b) a description stage, which uses the image content at each such site to form a local descriptor. The proposed Boundary Preserving Local Region detector and those that will be reviewed in this section tackle region *detection*; they use existing descriptors to capture the detected regions' shape, and standard matching techniques to demonstrate their applicability. Thus, work on shape descriptors and contour matching (e.g., [8, 31]) is complementary but separate from the focus in this section.

Local interest region detection is a long-standing research topic in computer vision, and scale or affine-invariant local regions [68, 69, 70, 43] are critically valuable for multi-view matching problems like wide-baseline stereo or instance recognition. Their invariant properties under image variations offers repeatability across images for robust matching of specific object instances. For generic object categories, on the other hand, they tend to be too sparse to capture rich statistics of visual information in an image; densely sampled local patches offer better

coverage and are regularly found to outperform interest points (e.g., see [72]), at the cost of much greater storage and computation. Recent work on dense interest points [92] shows how to merge advantages of either sampling strategy, balancing coverage with repeatability. In contrast to our region detection approach, however, prior methods for local region detection are unaware of object shapes or boundaries and thus often straddle object foreground and background, rarely firing on object parts.

Due to steady advances in bottom-up segmentation algorithms [4, 41], increasingly researchers are considering how to employ segments as base features, in place of local patches [38, 89, 75, 80]. Segments are appealing since they capture object shape and have broader spatial coverage. However, the instability of segmentation algorithms with respect to image variations can make the features' shapes unreliable or sensitive to parameter settings. Depending on illumination, viewpoints, or background clutter, segments may leak into the background or be fragmented into texture blobs, failing to capture true shapes of the object or its parts. Thus, existing work often focuses on how to select reliable segment-parts using labeled data [38, 89].

Multiple segmentations (generated by varying the segmentation parameters) are often used to expand the pool of candidates to increase the chance of hitting the true object (e.g., [67, 34]). Whereas existing methods typically try to find "good" full-object segments among this pool, my goal is to combine all segmentation hypotheses to obtain dense local regions across an image. I propose a novel spanning-tree representation to integrate shape cues from multiple segmentation hypotheses.

Much less attention has been given to the interplay between low-level local features and segmentation. The segmentation-based interest points proposed in [50] consist of ellipses fit to segment areas and corners computed on segment boundaries. However, corners may often miss shape cues of the regions, and fitting ellipses directly to segments can be susceptible to segmentation errors. In contrast, my approach captures objects' local shapes from segmented regions, while still retaining robustness to segmentation variations by combining multiple segmentation

hypotheses.

2.2 Object Segmentation with Shape Priors

Bottom-up image segmentation methods group low-level cues from color, texture, and contours to estimate the object boundaries in an image [4, 41, 22, 28]. Despite significant strides in recent years, it is widely acknowledged that a bottom-up process alone cannot reliably recover object-level segments—in particular, when objects are composed of heterogeneous colors and colors, or objects of similar appearance appear adjacent to one another.

To overcome the pitfalls of bottom-up approaches, researchers attempt a top-down strategy that uses shape models for the object to be segmented. Such top-down methods [12, 58, 95, 20, 52, 56, 17, 65] unify object shape models with bottom-up color/texture cues for object-level segmentation. Shape-based object detectors [73, 30, 86] use robust contour models to detect objects in cluttered images. These methods elegantly integrate top-down knowledge with bottom-up evidence, yet they heavily rely on a known (pre-trained) category-specific model. Category-specific shape priors in particular make strong assumptions about the viewpoint of the object to be segmented: for example, a classic test case consists of side views of horses.

Considering these limitations, I pursue a rather different approach. The main insight is that objects across different categories (at least partially) share their shapes—and this "shape sharing" enables us to transfer the shapes of a known class into unknown classes. I devise an exemplar-based partial shape match via BPLRs to transfer global shapes among different objects. The advantage of my approach is that it avoids requiring strong prior knowledge about the object(s) present; instead, it exploits shapes of one class to estimate those of possibly unknown classes, introducing a *category-independent* shape prior. Further, the exemplar-based design overcomes the viewpoint sensitivity of existing shape priors by allowing exemplars to span arbitrary classes and poses.

The notion of sharing visual properties across object categories has been pursued in var-

ious forms. In object detection, jointly training multi-class object detectors allows the reuse of common discriminative features [91, 73], and transfer learning promotes sharing by incrementally training new objects [26, 7, 87, 2, 78]. For shapes in particular, knowledge about shared properties is typically expressed in parametric forms, e.g., Gestalt cues like symmetry [59], or handcrafted geometric primitives [10]. Recent work attempts to relax such parametric restrictions by discovering prototypical local geometric features [79]. In contrast, my approach goes beyond local sharing to consider global shape projections at the object level. In addition, my exemplar-based, non-parametric approach to sharing is novel relative to all of the above, and offers greater flexibility to the rich variations of object shapes and poses.

Recent methods generating multiple figure-ground segmentations [24, 19] are closely relevant to my approach. Like my method, they also assume access to a database of segmented images, generate category-independent object segmentation hypotheses, and offer improvements over purely bottom-up image segmentation. However, the previous techniques rely on local bottom-up cues (color, texture, contour strengths). Their lack of shape priors hurts performance, particularly for cases where color consistency is insufficient to form a good segment, as I will show in the results (Chapter 4).

Multiple figure-ground segmentations provide multiple hypotheses of object regions that may possibly overlap one another. To get a more compact representation of the image with object delineation, some work aims to map multiple hypotheses to a single segmentation that provides non-conflicting regions of objects in the image. Along this line, there are recent attempts to generate an image segmentation from multiple segment hypotheses [42, 16]. They do so by selecting disjoint segments among a pool of multiple region hypotheses that best fit the bottom-up image evidence, yet lack shape priors.

2.3 Segmentation-Driven Image Matching

For region-based recognition, not only is it important to detect good regions, but it is also crucial to match them reliably and efficiently. A matching associates visual entities (images, objects, or scene) via similarity, which is vital for visual recognition. Due to their robustness to deformation and occlusion, matching local features has long served as a key component on many computer vision problems, and is especially important to today's object recognition and image retrieval tasks that use local features for image representation (e.g., [36, 53, 76, 9]).

However, the locality of such features yields noisy correspondences when used alone, and thus adding geometric constraints is crucial to select reliable matches that have consistent spatial layout. In a restricted setting like wide-baseline stereo, parametric constraints (e.g., affine transformation or epipolar geometry) are used to represent the deformation of an object under viewpoint change [64, 32, 44, 76]. For matching objects in a generic setting, however, a single parametric transformation between images is insufficient since the geometric deformation is not predictable. Instead, non-parametric approaches that identify a group of matches having minimal geometric distortion may be preferable. For example, optimization-based methods [9, 57] attempt to minimize the total pairwise distortion among all corresponding features.

Though their pairwise constraints are so powerful as to address various geometric deformations, those methods have a significant computational overhead to compute geometric distortions among all pairs of features, which in turn limits the number of features that can be handled only by tens to a few hundreds, making it hard to fully convey visual information in the image. The proposed 1D string representation significantly reduces the computational complexity by modeling distortions only between nearby features along the string, enabling a dense non-parametric matching with an efficient optimization.

Recent graph-based matching methods [63, 23] represent an image with a regular grid graph and match graphs between images. Such graphical approaches offer leading accuracy on scene matching [63] and object categorization [23]. However, their graphical model builds

on loopy graph of 2D image grid that costs expensive iterative optimization for matching. In addition, they ignore a spatial layout suggested by segmentation, possibly being vulnerable to non-rigid deformation among different regions of an image.

Going beyond local features, matching segmented regions has gained growing interest [38, 89, 90]. Due to the steady advances in segmentation methods, segmented regions have the potential to provide semantically meaningful spatial support like an object and its parts for reliable matching. In practice, successful results on object classification [89] and object detection [38] support the validity of a region-based approach. On the other hand, it is widely acknowledged that segmented regions are still less stable than local features under image variations. To address this trade-off between local features and the segmented regions, instead of matching the segmented regions directly, I propose to use the segmented regions to guide the matching of local features. This keeps the robustness of local features under image variations while still enforcing the spatial layout suggested by larger segmented regions.

A related "bundling features" algorithm [96] uses a region as a unit for which geometric constraints are independently imposed. Unlike my approach, however, that method does not match local features; instead, it augments the representation of a region by encoding the spatial configurations of local features detected within the region. Whereas I use regions for spatial layout, they use local features for it that lack global spatial support to encode the layout reliably.

2.4 Fast Image Matching with a Region Hierarchy

As shown in the previous section, regions can provide strong geometric cues for matching problem. Pursuing this idea of using regions for geometric regularization, the last part of my thesis explores regions' inherent hierarchical structure (e.g., an object and its parts) to improve the matching. Particularly, I focus on how to use such a region hierarchy for fast dense pixel matching.

Fast matching is important for scalable recognition, particularly for exemplar-based ap-

proaches that need to match a large number of images in the database. Moreover, fast matching becomes more critical for the dense pixel matching problem since it requires finding correspondences for all the millions of pixels between images, not a few hundreds or thousands of sampled local features. These challenges pose important research goal: achieve a fast matching that still provides reliable matches.

Traditional matching approaches aim to estimate very accurate pixel correspondences (e.g., sub-pixel error for stereo matching), given two images of the same scene with slight viewpoint changes. For such accurate localization, most methods define the matching cost on pixels. In particular, the pixel-level Markov random field (MRF) model, combined with powerful optimization techniques like graph-cut or belief propagation, has become the *de facto* standard [83, 15]. It casts matching as a graph optimization problem, where pixels are nodes, and edges between neighboring nodes reflect the existence of spatial constraints between them. The objective consists of a data term for each pixel's matching cost and a smoothness term for the neighbors' locations.

Unlike traditional instance matching, recent work attempts to densely match images containing different scenes [62, 45]. In this setting, the intra-class variation across images is often problematic (e.g., imagine computing dense matches between a sedan and a convertible). As introduced in the previous section, stronger geometric regularization is one way to overcome the matching ambiguity—for example, by enforcing geometric smoothness on all pairs of pixels, not just neighbors [9]. However, the increased number of pairwise connections makes such approach too costly for dense pixel-level correspondences, and it lacks multi-scale regularization.

To address such computational challenges, researchers have explored various computationally efficient solutions, including hierarchical optimization [62], randomized search [6], spectral relaxations [57], and approximate graph matching [23]. Particularly, SIFT Flow [62] and PatchMatch [6] are today's most popular and powerful methods for dense pixel matching.

The SIFT Flow algorithm pioneered the idea of dense correspondences across different scenes [62]. For efficiency, it uses a multi-resolution image pyramid together with a hierarchical optimization technique inspired by classic flow algorithms. However, its run-time is not fast enough for scalable recognition since its pyramid graph relies on a conventional pixel-based model that involves a huge number of pixels to optimize even in the coarser resolutions. Also, SIFT Flow treats graphs from different resolutions independently, which can produce gross errors once the solution in one resolution goes wrong. The PatchMatch algorithm computes fast dense correspondences using a randomized search technique [6]. For efficiency, it abandons the usual global optimization that enforces explicit smoothness on neighboring pixels. Instead, it progressively searches for correspondences; a reliable match at one pixel subsequently guides the matching locations of its nearby pixels, thus implicitly enforcing geometric smoothness. Though improving the run-time substantially, its implicit geometric smoothness often produces noisy correspondences, particularly when matching different scenes or objects with visual changes.

Despite the variations in graph connectivity, computation techniques, and/or problem domains, all of the above approaches share a common basis: a flat, pixel-level objective. The appearance matching cost is defined at each pixel, and geometric smoothness is imposed between paired pixels. In contrast, the deformable spatial pyramid model I propose in Chapter 6 considers both matching costs and geometric regularization within multiple spatial extents. I show that this substantial structure change has dramatic impact on both speed and accuracy of dense matching.

Rigid spatial pyramids are well-known in image classification, where histograms of visual words are often compared using a series of successively coarser grid cells at fixed locations in the images [53, 97]. Aside from my focus on dense matching (vs. recognition), my work differs substantially from the familiar spatial pyramid, since it models geometric distortions between and across pyramid levels in the matching objective. In that sense, my approach relates to

deformable part models in object detection [27] and scene classification [74]. Whereas all these models use a few tens of patches/parts and target object recognition, my model handles millions of pixels and targets dense pixel matching.

The use of local and global spatial support for image alignment has also been explored for mosaics [85] or layered stereo [5]. For such instance matching problems, however, it does not provide a clear win over pixel models in practice [83]. In contrast, I show it yields substantial gains when matching generic images of different scenes, and my regular pyramid structure enables an efficient solution.

Chapter 3

Detecting Local Shape Regions

In this chapter, I introduce a Boundary Preserving Local Region (BPLR) detector published in [47].[1] BPLRs are segmentation-driven local shape regions that capture objects' local shapes by respecting object boundaries.

3.1 Motivation: Boundary Preserving Local Regions (BPLRs)

Local features are a basic building block for image retrieval and recognition tasks. Their locality offers robustness to occlusions and deformation, and when extracted densely and/or at multiple scales they capture rich statistics for recognition algorithms (e.g., for a bag of words representation). The general local feature pipeline consists of (a) a *detection* stage, which selects the image sites (positions, scales, shapes) where features will be extracted, and (b) a *description* stage, which uses the image content at each such site to form a local descriptor. This part of my work is concerned with the detection stage.

As discussed in Chapter 2, researchers have developed a variety of techniques to perform detection, ranging from sophisticated interest point operators [68, 69, 70, 43, 92] to dense sampling strategies [72]. While by design such methods provide highly repeatable detections across images, their low-level local sampling criteria generate many descriptors that straddle object boundaries, and—if they are too local—may also lack distinctiveness (i.e., patches of texture vs. actual object parts). On the other hand, while segmentation algorithms can produce boundary-preserving base features and reveal object shape [80, 38, 89, 67], they tend to be

[1] Code and data are available online: http://vision.cs.utexas.edu/projects/bplr

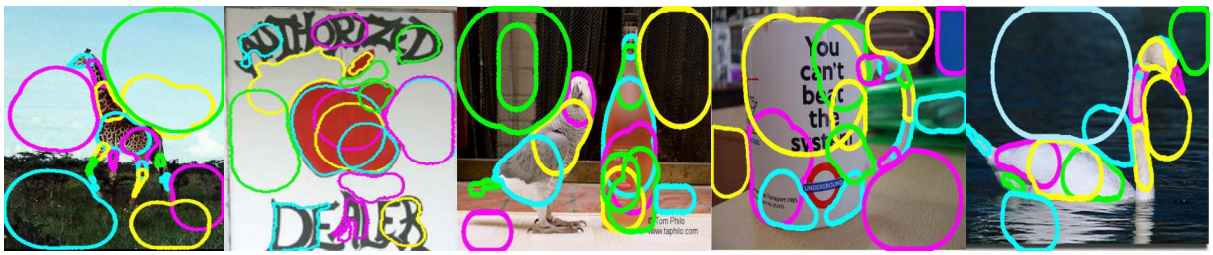


Figure 3.1: The proposed Boundary Preserving Local Regions (BPLRs) capture local object shape with dense spatial coverage. (We densely extract BPLRs across the image, but for legibility this figure displays only a few.)

sensitive to global image variations and so lack repeatability.

My goal is to address this current tradeoff, and create a detector for features that are both distinctive *within* the image as well as repeatable *across* images. To this end, I propose a novel dense local region extraction algorithm driven by segmentation, creating a Boundary Preserving Local Region (BPLR) detector. Figure 3.1 shows example detections of BPLRs in several images.

Because our extracted regions tend to preserve object boundaries, they are informative for object shape. At the same time, because they link sampled elements across multiple segmentations, they are robust to unstable segmentations and thus repeatable across images. Finally, their dense coverage of the image ensures to retain reliable feature statistics that are critical for recognition and matching. Figure 3.2 depicts the key contrasts between our approach and existing methods.

3.2 Extracting Boundary Preserving Local Regions

In this section, I will present the technical details to extract BPLRs. I first describe how to sample initial elements using the input segmentations (Sec. 3.2.1). Then I explain how to link these elements across the image (Sec. 3.2.2). Finally, I show how to use the computed structure to extract dense groups of elements, each of which is a shape-preserving region (Sec. 3.2.3).

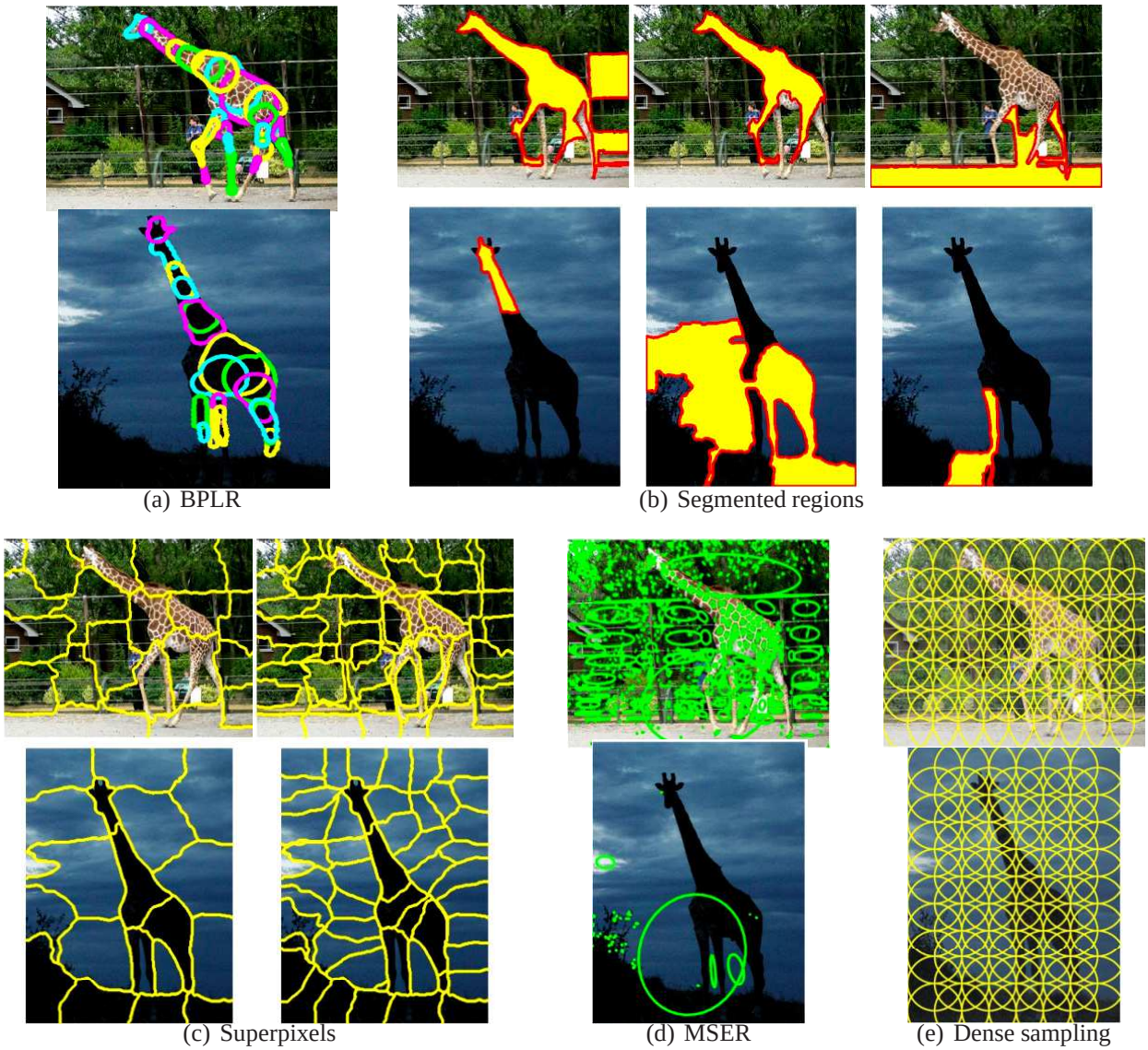


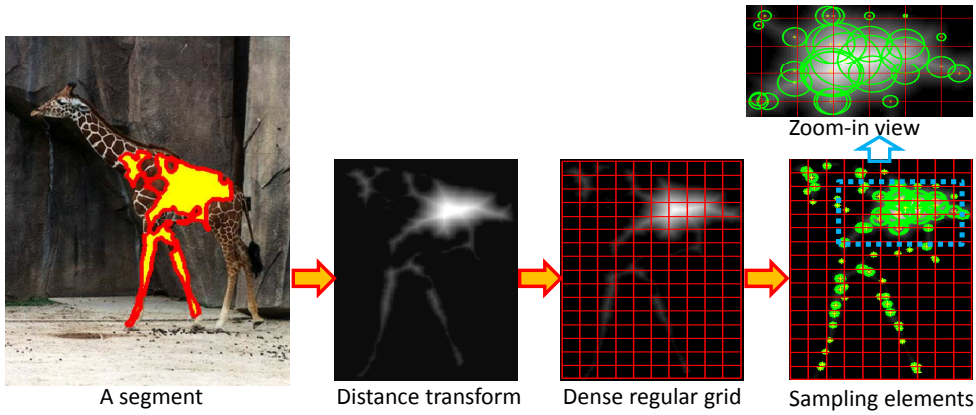
Figure 3.2: Illustration of BPLR's key contrasts with representative existing detectors. **(a) The proposed BPLR** features are reliably repeated across different object instances in spite of large intra-class variation in pose and appearance. They respect object boundaries while maintaining good spatial coverage per region. (Note, I display only a sample for different foreground object parts; our complete extraction is dense and covers entire image.) **(b) Regions** from a segmentation algorithm (here, obtained with [4], and pruned to only foreground-overlapping regions) typically produce some high quality segments, but the shape and localization often lacks repeatability across instances. Further, if a good segment encompasses the entire object, it won't match other instances with deformation. **(c) Superpixels** (obtained here with Normalized Cuts [80]) are also local and dense, but typically lose informative shape cues and lack repeatability (compare shapes of superpixels on the two giraffe instances). **(d) Local interest regions** (obtained with MSER [68]) are highly repeatable for multiple views of the same instance, but do not respect object boundaries and fire very differently across different instances of the same object class. **(e) Densely sampled regions** offer good coverage and "brute force" repeatability, but many features straddle object boundaries, and shape is mostly not preserved.

3.2.1 Sampling Initial Elements

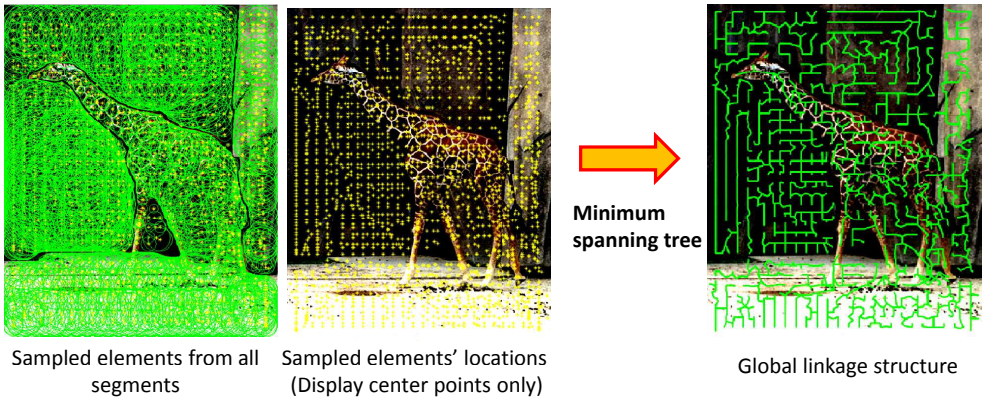
Given an image, we first obtain multiple overlapping segmentations. (Unless mentioned otherwise, we use the state-of-the-art algorithm developed in [3] to produce a high quality hierarchy of segments, though we also test a faster segmentation method [28] for comparison.) These segmentation hypotheses do not serve as detected regions; rather, we use them to guide the extraction of initial component features that we call "elements". Each element is a circle with a position (its center) and associated scale (its radius).

The goal of our novel sampling strategy is to balance both density and object boundary preservation. To that end, we compute a distance transform (DT) from the boundary edges of each segment, and then subdivide the segment into a dense grid of cells (e.g., 6×6 pixels per cell). For each cell, we sample an element at the location with the maximal distance transform value within the cell, and set the radius of the element by that maximal distance value. Figure 3.3(a) shows sampled elements from one segment.

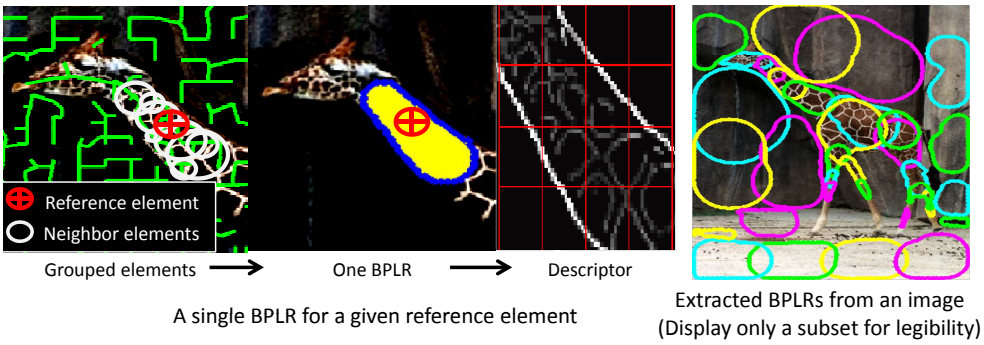
Selecting elements' scale by the DT prevents them from overlapping the originating segment's boundary. At the same time, refining the dense sampling positions by the maximal DT values pushes sampled locations to the inner part of each segment, keeping elements originating from the same segment closer to one another than those from different segments. Due to this geometric property, when we link elements across all segments in the next stage (Sec. 3.2.2), we have a soft preference to join elements originating from the same segment. In addition, the local nature of our sampling approach limits the influence of segment "errors"; that is, holes or leaks (relative to the true object boundaries) do not destroy the sampling and scale selection. Thus, we retain a large number of good elements that respect object boundaries even with partially flawed segments.



(a) Sampling elements



(b) Linking elements across image



(c) Grouping neighbor elements into BPLRs

Figure 3.3: Main components of the approach. Best viewed in color. **(a)** For each initial segment, we sample local elements densely in a grid according to its distance transform (left: segment; lower right: grid; upper right: zoom-in to show sampled elements and their scales). **(b)** Elements are linked across the image, using the overlapping multiple segmentations to create a single structure that reflects the main shapes and segment layout. **(c)** Using that structure, we extract one BPLR per element. Each BPLR is a group of neighboring elements. Finally, the BPLR is mapped to some descriptor (we use PHOG+gPb).

3.2.2 Linking Elements Throughout the Image

Next we want to take these elements and define the neighborhood structure across the entire image, which in turn will determine how we extract groups of neighboring elements to form BPLRs. A naive linking of the elements based on their spatial (image) distance would fail to capture the image-wide contours and shape revealed by the multiple segmentation hypotheses. Instead, we define a two-step linking procedure that accounts for this structure and reduces cross-object connections.

The first step computes a global linkage graph connecting all element locations via a minimum spanning tree, where each edge weight is given by the Euclidean distance between the two points it connects. By minimizing the sum of total edge weights, the resulting spanning tree removes the longer edges from the graph—most of which cross object boundaries due to the geometric property of the DT-based sampling. As a result, we have a global link structure respecting object boundaries, in which every element has at least one direct neighbor (see Figure 3.3(b), rightmost image).

Whereas the above step reduces connectivity for more distant elements, we also want to reduce connectivity for elements divided by any apparent object contours. Thus, in the second linkage step, we perform a simple post-processing of the spanning tree that removes noisy tree edges that cross strong intervening contours. We compute the contour strength at each pixel using the "globalized probability of boundary" (gPb) detector [66], and remove links crossing contours exceeding the average non-zero gPb value in the image. Nonetheless, even an erroneous pruning at this stage has limited impact, given the density of the elements and the manner in which we ultimately group them into regions, as we explain in the next section.

3.2.3 Grouping Neighboring Elements into Regions

Finally, we use the elements and the computed graph to extract a dense set of boundary-preserving local regions (BPLRs). For every element (i.e., every node in the graph), we create

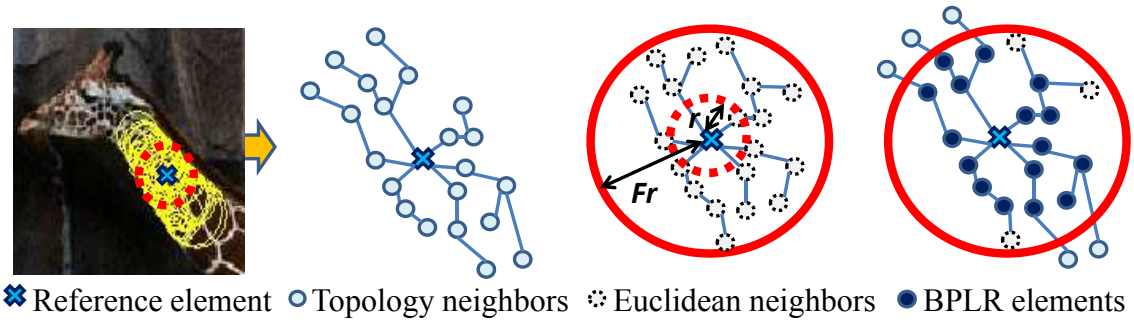


Figure 3.4: Grouping neighboring elements relative to a reference element. Topology neighbors: up to $N(= 3)$ hops for the reference; Euclidean neighbors: within F times the scale of the reference; BPLR elements: intersection of topology and Euclidean neighbors.

one BPLR. Each BPLR consists of that "reference" element, plus a group of its neighbors in the graph (see Figure 3.3(c)).

We define the neighborhood based on two measures: topological distance in the graph (how many link hops separate the elements), and Euclidean distance in the image (L_2 distance between the elements' centers). The neighbors for a reference element are those within the intersection of regions spanned by either distance. Specifically, the topological neighborhood consists of any elements within N hops along the graph relative to the reference element, while the Euclidean neighborhood consists of any elements within a radius equal to F times the reference element's scale r (see Figure 3.4). Note that the topological radius is fixed over all elements in the graph (and all images), while the Euclidean radius is proportional to each element's scale. The neighbors of each reference element within this intersected area form a BPLR.

Why do we use the two distances? Using the Euclidean distance alone would maintain scale invariance, but is blind to the graph connectivity, which intentionally accounts for estimated image boundaries. On the other hand, topological distance accounts for this connectivity, and in the face of unstable segmentations, it tends to select neighbors better than the elements' noisy scale estimates; but, if used alone, it would not be robust to significant scale changes. Thus, our design is intended to balance the good parts of both.

Since we extract the BPLRs for every densely sampled element, the resulting detections

are also dense. The exact number per image depends on the initial segmentation and sampling grid; to give a concrete sense, using the initial multiple segmentations we obtain about 150-250 segments, and then our method generates $\sim$7,000 features per image.

While earlier uses of the distance transform for shape-based representations require fairly clean segmentation (e.g., a pure silhouette for medial axis or shock graph extraction [84]), our scheme remains quite robust with challenging natural images due to its linking procedure and dense sampling. By definition our approach has some dependence on the original set of multiple segmentations; however, because our linking scheme connects elements *beyond* their originating segment, it is fairly robust to segmentation variations, recovering larger descriptive regions that partially overlap different segments. In general, we'd prefer the input err towards finer segments, since we will produce candidate regions that join them.

Our approach performs region detection. To use these regions for matching, we need to further extract a *descriptor* for every region. One could in principle employ any descriptor with our detector. In our experiments, we use Pyramids of Histograms of Oriented Gradients (PHOG) [14] computed over the gPb-edge map (see third image in Fig. 3.3(c)), which is similar to the descriptor used in [38]. It represents the outline of the shape as well as (coarsely) its inner texture, and thus is a good match for BPLR's strengths. To extract the PHOG+gPb feature, we put a bounding box around the BPLR, and nullify gPb values outside of the BPLR boundaries, excluding external edges from the histogram counts. Algorithm 1 presents pseudo-codes for the whole procedure of computing BPLRs.

3.2.4 Elaboration on Key Design Factors

In this section, I point out key technical factors in the BPLR design and explain how they overcome the weaknesses of existing methods. I also discuss about parameter choices.

For sampling elements in Sec. 3.2.1, I apply a distance transform to the initial segments. While earlier uses of the distance transform for shape-based representations require fairly clean

Algorithm 1: BPLR extraction

Data: Input multiple segmentations $\mathbf{S}$
Result: BPLRs and their PHOG descriptors

Sampling elements;
input : Multiple segmentations $\mathbf{S}$
output: Sampled elements from each segment in $\mathbf{S}$
foreach *segment* s_i *in* $\mathbf{S}$ **do**
  Distance transform on the boundary of s_i;
  Put a bounding box b_i on the segment s_i;
  Divide the b_i into grid cells (e.g., 6×6 pixels);
  foreach *grid cell* g_i **do**
    Pick a position p that has a maximum distance transform value in g_i (let denote the maximum value by m_{dt});
    Sample a circle (i.e., element) whose center is p and radius is m_{dt};
  end
end

Linking elements;
input : A set of sampled elements $\mathbf{E}$ from all segments $\mathbf{S}$
output: A graph G that links the sampled elements
Compute Euclidean minimum spanning tree graph for the 2D positions of all sampled elements in the image;
Remove graph edges that cross the strong contour whose value is above the threshold;

Grouping elements;
input : A set of sampled elements $\mathbf{E}$ and a linkage graph G
output: Boundary Preserving Local Regions (BPLRs)
foreach *element* e_i *in* $\mathbf{E}$ **do**
  Get Euclidean neighbors $\mathbf{N_{ei}}$: pick elements which are within the F times the radius of e_i;
  Get topological neighbors $\mathbf{N_{ti}}$: pick elements which are within the N hops from the e_i in the graph G;
  Compute the intersection $\mathbf{N_i}$ of $\mathbf{N_{ei}}$ and $\mathbf{N_{ti}}$;
  Form one BPLR that covers the area by the elements in $\mathbf{N_i}$;
end

Descriptor for BPLRs;
input : BPLRs and a gradient map of the image
output: PHOG descriptors for BPLRs
foreach *BPLR* b_i **do**
  (Optional) Dilate b_i by l pixels (See Section 3.3);
  Put a bounding box on b_i;
  Nullify the gradient map values outside the b_i in the bounding box;
  Compute a PHOG descriptor on the gradient map of the bounding box;
end

segmentation (e.g., a pure silhouette for medial axis or shock graph extraction [84]), my scheme remains quite robust with flawed segmentation in challenging natural images. This is largely because it combines with a dense local sampling strategy. Due to the locality, errors in a segment do not destroy the whole procedure; due to the density and regularity, intact elements tend to dominate noisy ones, considering errors in a segment often happen partially. Further, this combination is key to both BPLR's repeatability (via dense extraction) and distinctiveness (via the shape cues from DT).

Multiple segmentation approaches typically aim to find full-object segments by varying segmentation parameters [67, 34]. However, such a multi-parametric approach inherently entails noisy segments with redundancy, degrading overall feature quality. My linking scheme in Sec. 3.2.2 connects elements *beyond* their originating segment. Thus, we generate larger descriptive regions that partially overlap different segments, which not only reduces the redundancy of the initial multiple segmentations, but also adds robustness to segmentation variations.

The most important parameters in BPLR extraction are N, the number of hops to define topological neighbor, and F, the scaling factor to define Euclidean neighbor (see Figure 3.4). These two parameters define the size of the extracted BPLR: as those values increase, the BPLR's size gets larger. We find that too small or too large values of N and F hurt the performance: tiny BPLRs lack distinctive shapes, often confused by noisy texture; too big BPLRs become more sensitive to image variations like shape deformation or background clutter. In the experiments, I fix $N = 50$ and $F = 5.0$ that it consistently produces robust performance. To add more robustness to parameter choice, one could extract BPLRs using multiple N hops and F scales.

Another factor that makes an impact on BPLR's quality is the segmentation choice. By definition, my method has some dependency on the original set of multiple segmentations; however, as I will show in Sec. 3.3.3, my method consistently improves upon the initial segments of multiple different input segmentation methods.

3.3 Results

In this section, I compare BPLR to other local features and then apply it to high-level vision tasks. The main goals of the experiments are 1) to demonstrate the raw quality of my region detector, 2) to explore the impact of initial segmentation methods on BPLR's quality, and 3) to show its effectiveness when used for tasks that require reliable feature matching. For the first aspect, I analyze repeatability and localization accuracy across object categories (Sec. 3.3.1 and 3.3.2). For the second, I compare BPLR's quality from two different segmentation methods in terms of both feature quality and extraction time (Sec. 3.3.3). For the third, I apply BPLR to foreground discovery and object classification (Sec. 3.3.4 and 3.3.5).

Evaluation metric: I use three different metrics to evaluate feature quality. I test those metrics on images of different objects and scenes. Here, I give a brief summary of the metrics; formal definitions will be given in Sec. 3.3.1 and 3.3.2.

For evaluating repeatability—how regularly features are detected across different images—I use the *Bounding Box Hit Rate - False Positive Rate* (BBHR-FPR) metric defined in [77]. Simply put, it evaluates how well foreground features on an object match other foreground features in the same class. Note, I perform category-level evaluation, not an instance-level that tests repeatability by synthetically generated images by parametric transformations.

For evaluating distinctiveness—how well features are matched at the correct locations of object parts—I introduce the *Bounding Box Overlapping Score - Recall* (BBOS-Recall) and *Bounding Box Detection Rate* (BBDR) metrics. Both metrics are designed to measure how accurately feature matches can predict objects' positions and scale. However, they are complementary to each other; the former focuses on recall of matched features, while the latter focuses on precision.

Datasets: I use four public datasets: the ETHZ Shape Classes [31], the ETH-TUD set collated by [77], the Caltech-28 set collated by [18], and the Caltech-101. I choose the

ETHZ Shape Classes dataset in order to validate BPLR's advantages on shape-based classes. ETH-TUD set is chosen for comparing to existing semi-local type features [77, 54] as they tested their methods on this dataset. The Caltech-28 set is used for foreground segmentation task, while the Caltech-101 dataset is a popular benchmark for the object classification task. Figure 3.5 shows example images from each dataset.

Baselines: I compare to several state-of-the-art results in the literature ([77, 54, 1, 18] and many Caltech-101 numbers), plus three alternative extraction methods:

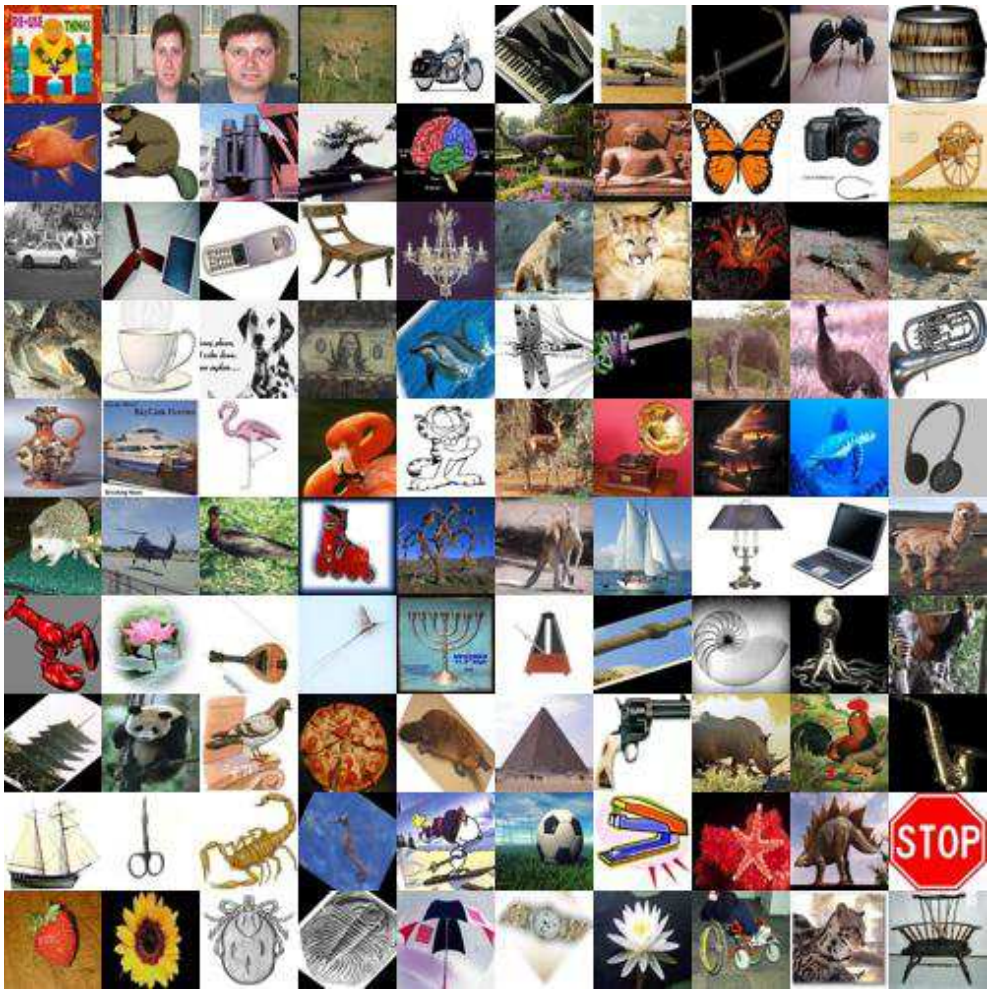
- MSER: MSER is the best local interest region in the evaluation by [70]; I use the VLFeat open source library [94] to generate 400-500 MSERs per image, where I vary its control parameter to obtain extremal regions of different stabilities.
- DENSE PATCH: This method samples rectangular patches at a regular grid every 6 pixels in the image over four different patch sizes. This approach is frequently used in current methods [53, 11, 93].
- SEGMENT: This method uses the same overlapping segments that serve as input to my algorithm. I test two different segmentation methods [3, 28].

Note, the former two baselines are widely used in the recognition literature, while the last is used in the state-of-the-art region-based approach of [38], making these strong and very informative baselines.

Implementation details: I generate multiple overlapping segmentations for each image using the algorithms of [3] and [28], with the authors' publicly available codes. I vary parameters so as to provide 5-200 segments per segmentation, pool all the segments, and use them as input to our algorithm throughout. The method in [3] provides high-quality initial segments, while the segmentation by [28] runs much faster; in Sec. 3.3.3, I compare their trade-off in BPLR extraction between run-time and feature quality. Unless otherwise mentioned, BPLR in the below refers to the BPLR derived from the segmentation of [3].

(a) ETHZ Shape Classes

(b) ETH-TUD



(c) Caltech-101

Figure 3.5: Example images from the datasets used in our experiments. For each dataset, we randomly display one or two images from each object class of the dataset. We see that each dataset shows wide range of image variations in poses, scales, shapes, and background clutter, posing substantial challenges for visual tasks. We omit Caltech-28 set in this illustration as it is a subset of Caltech-101.

I extract BPLRs from elements sampled in grid cells of 6 $\times$ 6 pixels with $F = 5.0$, $N = 50$. To link elements in the minimum spanning tree, I use code by [82]. This setting generates on average 6,000-8,000 BPLRs in a 400 $\times$ 300 image, and takes about 3-4 seconds for BPLR extraction after the initial segmentation on a machine with a 3.4GHz CPU. The initial multiple segmentations take 3-4 minutes for [3] and about 10 seconds for [28].

For all features, I use the HOG descriptor with 4 $\times$ 4 spatial bins and 8 orientation bins, for a 128-dimensional descriptor. To "match" features, I simply use nearest neighbor (NN) search with Euclidean distance on the descriptors. For BPLR, SEGMENT, and MSER, I dilate the regions by 40% over the original scale when computing descriptors, which I found provides better matching accuracy by including informative visual cues across the object boundaries while still preserving their original shapes. Also, I remove the tiny regions (less than 400 pixels) that often introduce matching ambiguity, which particulary contributes to improving the MSER's performance over my previous publication [47]. I also test the SIFT descriptor on the DENSE PATCH baseline and find it provides similar performance to HOG. Thus, I use HOG for all the methods for fairness.

3.3.1 Repeatability for Object Categories

When matching images of the *same* scene or object, one can test repeatability by synthetically warping the images with parametric transformations (e.g., see [70]). However, such measures are not applicable to images of *generic objects*, where the goal is to ensure similar object parts are detected across instances.

Thus, I quantify repeatability using the *Bounding Box Hit Rate - False Positive Rate* (BBHR-FPR) metric defined in [77]. To compute the BBHR-FPR, one selects features in the cluttered test image that have a match distance below a threshold with foreground features in the training images and declare a "hit" if at least five such features are inside the test image's bounding box. FPR counts those selected test features outside its bounding box. See Figure 3.6.

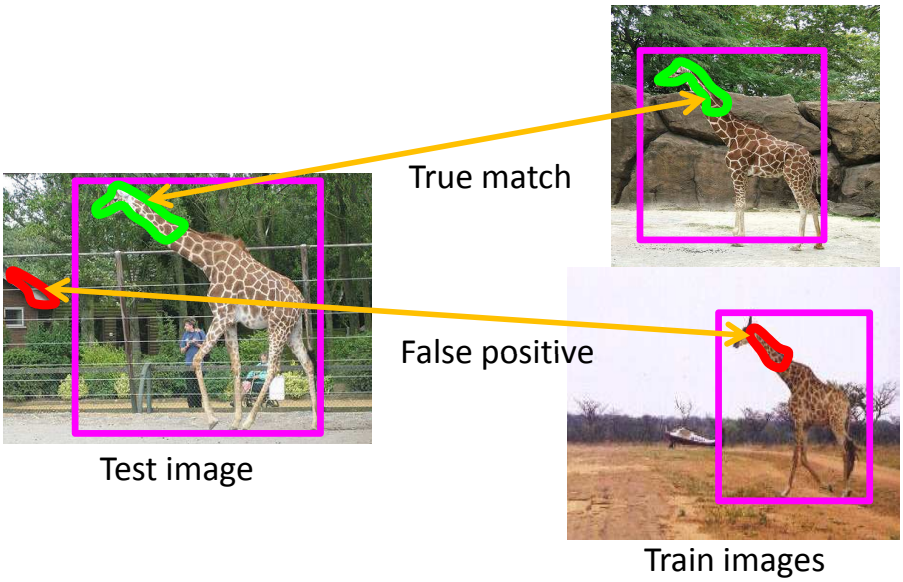


Figure 3.6: Illustration of the BBHR-FPR metric. A bounding box hit is declared when at least k true matches are found. I set $k = 5$ following the original author's choice [77]. BBHR-FPR records the average hit rate and corresponding false positive rate for all test images.

In my experiment, I compute the match distance of a test feature by the *ratio* of its best HOG distance with the foreground training features to the best one with background features, where features in the training images are labeled as foreground when they are inside the bounding box and their best match is inside another training bounding box. The second condition reduces the ambiguity of bounding box annotation, e.g., background grass in a giraffe's bounding box. Sweeping through all distance thresholds, one records this average hit rate and corresponding FPR for all test images to form a BBHR-FPR curve. In short, the metric captures to what extent the selected features are repeatedly detected on the object foregrounds.

Figure 3.7 shows the results for the ETHZ Shape Classes dataset, using a 50-50 train-test split. Our BPLR outperforms all the baselines. In particular, BPLR's gains become larger in shape-varying classes like Giraffe and Swan. The BBHR is boosted by the density of our features, and it also maintains a low false positive rate by capturing the distinctive local shapes that help reliably discern object foreground from background. In addition, we see that dense features (BPLR and DENSE PATCH) offer better repeatability than sparse features (MSER and

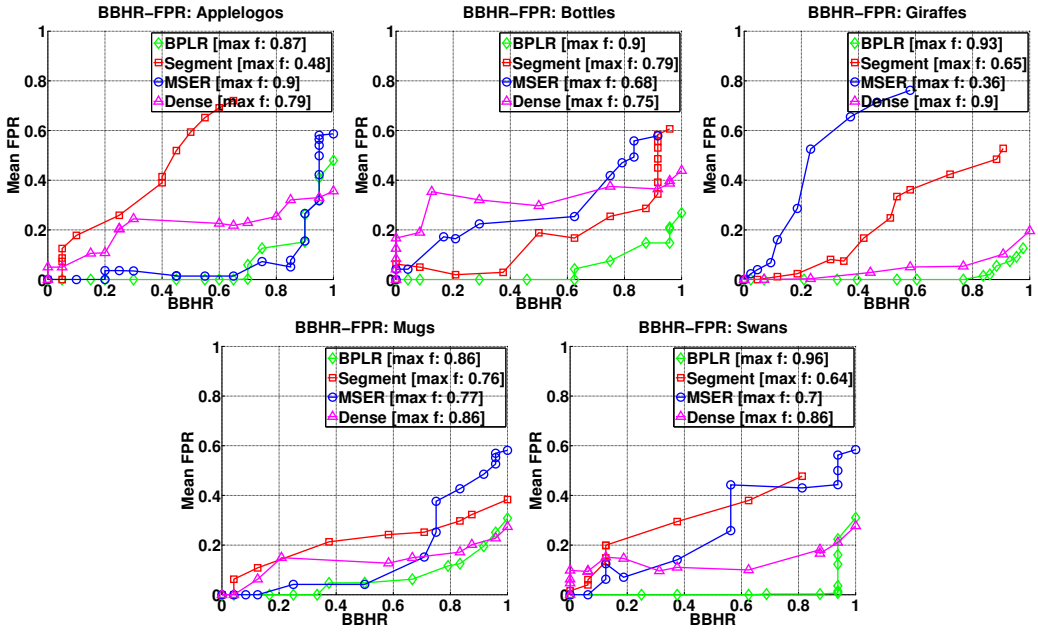


Figure 3.7: Repeatability on ETHZ objects. Plots compare my approach (BPLR) to three alternative region detectors: MSER, dense sampling, and segments. Quality is measured by the bounding box hit rate-false positive rate tradeoff (BBHR-FPR). Curves that are lower on the y-axis (fewer false positives) and longer along the x-axis (higher hit rate) are better. Maximum F-numbers in the legend are defined as the maximum harmonic mean of BBHR and 1-FPR along the curves, meaning the best combination of two scores along the curve; higher F values are better.

SEGMENT), implying the density is beneficial for repeatability.

Figure 3.8 compares to two state-of-the-art semi-local feature extraction methods [77, 54], using the ETH+TUD data and setup defined in [77].[2] Both previous methods build configurations of neighboring visual words, making them relevant to my approach to group element features. Our BPLR outperforms both—remarkably, our extraction is generic, bottom-up whereas the baselines require class-specific supervision. Also, gains on the non-rigid objects again emphasize BPLR's strength for shape-based objects.

[2]I exclude the Bike class, since it contains duplicated images in the test and training set, which inflates our results significantly.

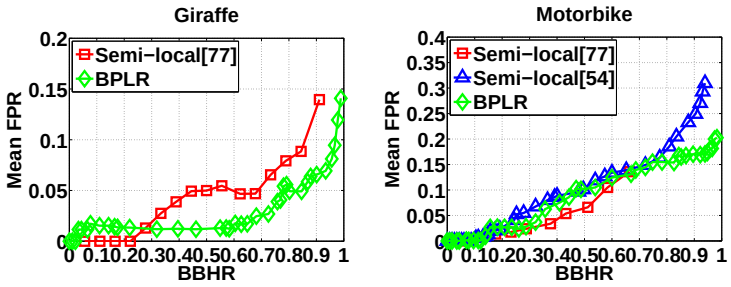


Figure 3.8: Repeatability on ETH+TUD objects. Plots compare my approach (BPLR) to two state-of-the-art semi-local feature methods [77, 54]. Lower and longer curves are better. ([54] does not report results on the Giraffe class.)

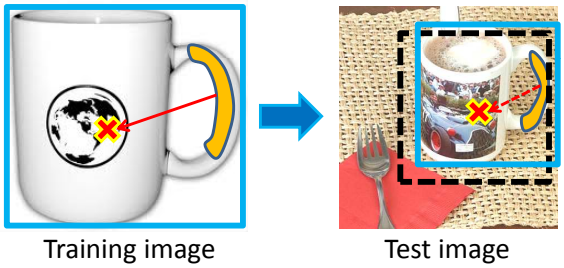


Figure 3.9: Illustration of the BBOS and BBDR metrics for localization accuracy. Given two matched regions and their relative scales, I project the training exemplar's bounding box into the test image (dotted rectangle). That match's BBOS is the overlap ratio between the projected box and the object's true bounding box. BBDR counts how many matches have more than 0.5 overlap ratio among all matches.

3.3.2 Localization Accuracy

The BBHR-FPR reveals repeatability, but not layout. Ideally, the detected regions would also match with spatial consistency; i.e., if a region is detected on the fender of the car in one image, we want the fender on a different car in another image to also be detected, with a similar shape.

To quantify this, I introduce the *Bounding Box Overlapping Score - Recall* (BBOS-Recall) metric. For each feature in a test image, I match it to the training features, and use each match's position and scale to project the training example's bounding box into the test image. The BBOS is the ratio between the intersection and union of this projected box and the test image's ground truth (see Figure 3.9). The recall is the portion of foreground test features that match a training

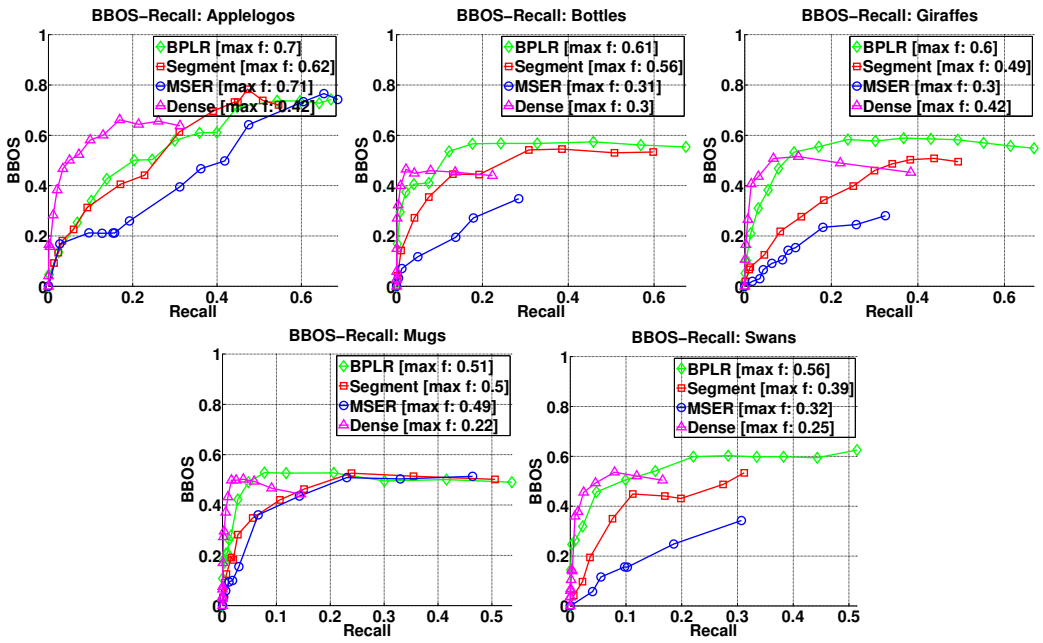


Figure 3.10: Localization accuracy on ETHZ objects. Plots compare my approach (BPLR) to three alternative region detectors: MSER, dense sampling, and segments. Quality is measured by the bounding box overlap score - recall (BBOS-Recall), which captures the layout of the feature matches. Curves that are higher in the y-axis (better object overlap) and longer along the x-axis (higher recall) are better. Maximum F-numbers in the legend are defined as the harmonic mean of BBOS and Recall along the curves.

foreground feature; false matches (to background) affect recall but not BBOS. A BBOS-Recall curve sweeps through the distance thresholds, and records the average BBOS and recall over all test images. In short, the metric captures the features' distinctiveness and localization accuracy.

Figure 3.10 shows the result for the ETHZ Shape data. In four of the five classes, my approach outperforms all the baselines, showing that its boundary-preserving property enhances localization. As in BBHR-FPR, it is particularly strong for the shape-varying classes, Giraffe and Swan. In contrast, other local-type features, MSER and DENSE PATCH, are less distinctive, and fail to localize matches reliably (e.g., a patch covering a small textured area on one giraffe's body may match anywhere in another giraffe). However, the DENSE PATCH baseline obtains better BBOS at lower recall range, e.g., Applelogo or Mug classes, likely because its rectangular shape happens to fit well to a regular shape of those classes for some scales. The shape-based

Feature	Mean BBDR
BPLR (Ours)	**0.67**
Segment	0.56
Dense	0.48
MSER	0.42

Table 3.1: Bounding Box Detection Rate (BBDR) on ETHZ objects. The score is averaged over all images. We see that BPLR improves upon its base segments and shows substantial gains over other local features in localization accuracy.

SEGMENT baseline provides better BBOS-Recall than MSER or DENSE PATCH, implying that distinct shapes improve that feature's localization power. Despite its ability to capture objects' shapes, however, it loses some points compared to our BPLR. I suspect this is due to two factors: first, the instability of segmentations across instances, and second, the segments that cover entire objects are not easily matched if there is a viewpoint change or deformation.

Though the BBOS-Recall metric captures well the localization power of the features, it is evaluated only for the foreground features in the test image, missing the errors by false positives from background features. Therefore, to complement it, I introduce the *Bounding Box Detection Rate* (BBDR) metric. Given a distance threshold, I can obtain features in a test image that are matched to the foreground training features. Then, the features whose BBOS is more than 0.5 are declared as true detections, giving the rate of true detections over all the features that fire at the given threshold (see Figure 3.9). Note that the features include both true and false positives. Finally, BBDR is defined as the *best* rate through the thresholds, indicating the features' power as a naive object detector. Table 3.1 summarizes the BBDR scores among the features. Our BPLR works best, showing BPLR achieves both localization accuracy and discriminating power from noisy backgrounds.

Figure 3.11 illustrates BPLR's localization power. For each test image on the left, I select the top five non-overlapping regions based on the foreground matching distance, and display them on the training images to the right. In addition, I show some examples of image-to-image

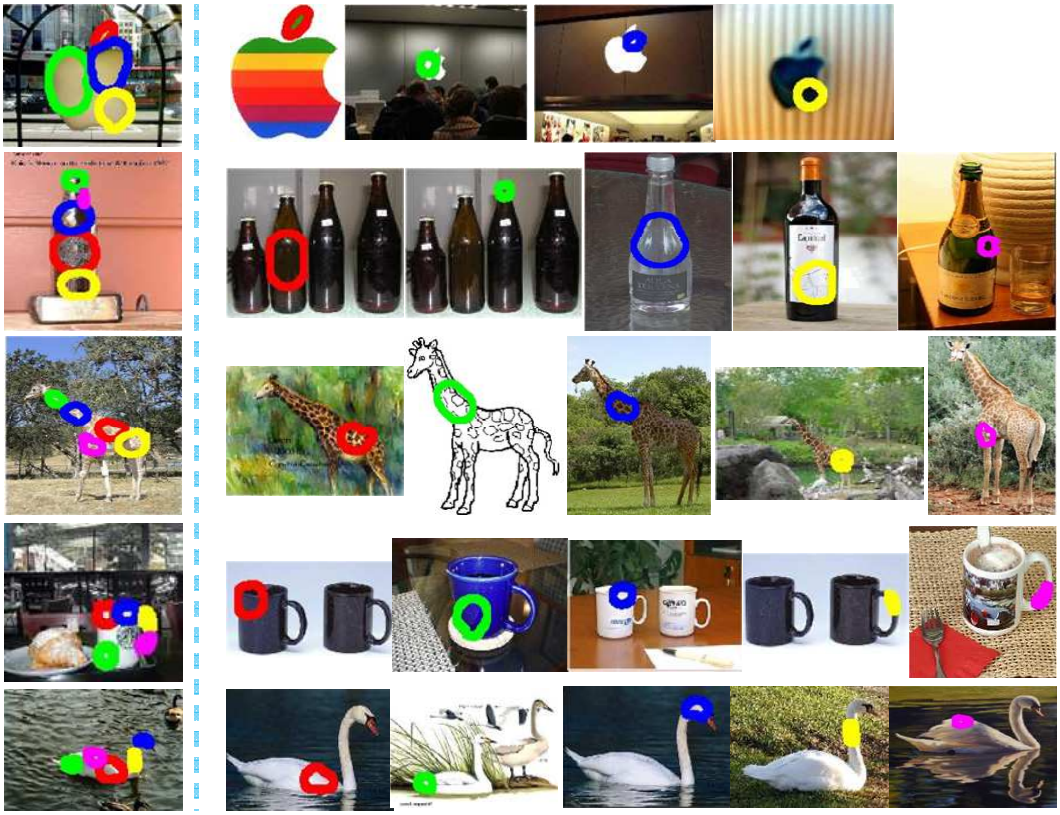


Figure 3.11: Example matches showing BPLR's localization accuracy. Colors in the same row indicate matched regions. Best viewed in color.

BPLR matches in Figure 3.12. In both cases, we see most matches are consistently localized in spite of scale changes, illumination, and background clutter. Overall, the results in this section indicate that our features' distinctiveness permits reliable localization, a strength for object detection.

3.3.3 Impact of the Initial Segmentation

For all the previous experiments, I use the Berkeley segmentation algorithm [3] to obtain the initial multiple segmentations. Although it provides sound seed segments for BPLR extraction, its run-time is somewhat costly (3-4 minutes for a 400 $\times$ 300 image on a 3.4GHz machine), limiting the scalability. Thus, I next test a more efficient segmentation method by

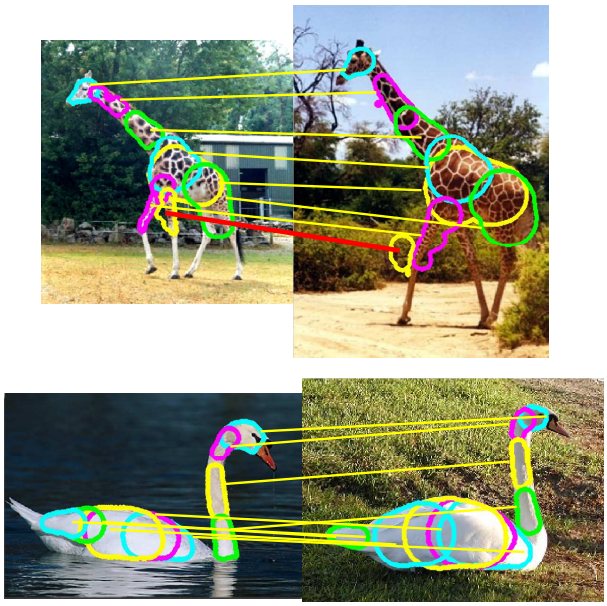


Figure 3.12: BPLR matches between two images. Given features in the left image, their best matches are shown in the right image. We can see most matches are correctly established. In the very thin region like the giraffe leg, we find some error that a BPLR leaks into the background and has a false match (marked by thicker red line). Despite such an exceptional case, most of the extracted BPLRs capture distinctive shapes of object parts, providing matches with accurate positions and scales.Best viewed in color.

Felzenswalb and Huttenlocher [28] ($\sim$10 seconds for multiple segmentations), in order to explore BPLR's trade-off between run-time and accuracy depending on the initial segmentation method. Whereas the former [3] generates segments using the high-quality gPb gradient map via a learned contour detector, the latter [28] relies on simple color similarity. The computation bottleneck in [3] actually comes from computing gPb. Thus, for descriptors, I compute HOG using the gPb map for the BPLR from [3]; for the BPLR from [28], I build HOG from the intensity gradient image. Note that I use different underlying gradient maps because each of maps is the best practical choice for the corresponding methods, respectively—one would not want to use a naive intensity gradient in [3] instead of high-quality gPb, nor would one want to sacrifice the efficiency of [28] by additionally computing the expensive gPb contour.

Fig. 3.13, Fig. 3.14, and Table 3.2 compare the quality of BPLRs from the two different segmentations. As expected, BPLR from the superior initial segmentation [3] (BPLR-UCM) provides better repeatability and localization accuracy. However, the efficient version (BPLR-EFF) carries its own value, improving over its initial segments as well as outperforming local features such as DENSE PATCH or MSER; further, it runs an order of magnitude faster than BPLR-UCM.

So far, I used different gradient maps when computing HOG for each feature; I used gPb contour map [3] for BPLR-UCM and Seg-UCM, and for all the others, I used a simple intensity gradient. Now I investigate the impact of the underlying gradient map by fixing the same map for all the features. To this end, I compare the results from the common intensity gradient image for all features. Table 3.3 summarizes the results. We see that the use of the intensity gradient loses some gains over when using gPb. However, BPLR-UCM still outperforms all the baselines, demonstrating its pure improvement upon its initial segments and the strengths over other local features due to its boundary-preserving property. Also, we see again that both BPLR-UCM and BPLR-EFF improves upon their initial segmentations over all the metrics. Segmented regions (Seg-UCM and Seg-EFF) show some advantage in localization due to their distinctive shapes, while losing points in repeatability due to unstable segmentation under image variations; DENSE PATCH (Dense) shows the opposite trend to segmented regions. In short, we can rank the feature quality in Table 3.3 as: BPLR-UCM $>$ BPLR-EFF $>$ Seg-UCM $\approx$ Seg-EFF $\approx$ Dense $>$ MSER. This reveals that Seg-UCM's gain over BPLR-EFF in Table 3.2 is attributed to its use of gPb contour map in HOG descriptor computation rather than the regions' inherent quality.

3.3.4 Foreground Discovery with BPLR

Now I examine BPLR's effectiveness for higher-level applications. My goal in the next experiment is to test whether my approach can improve *foreground discovery*, by replacing the

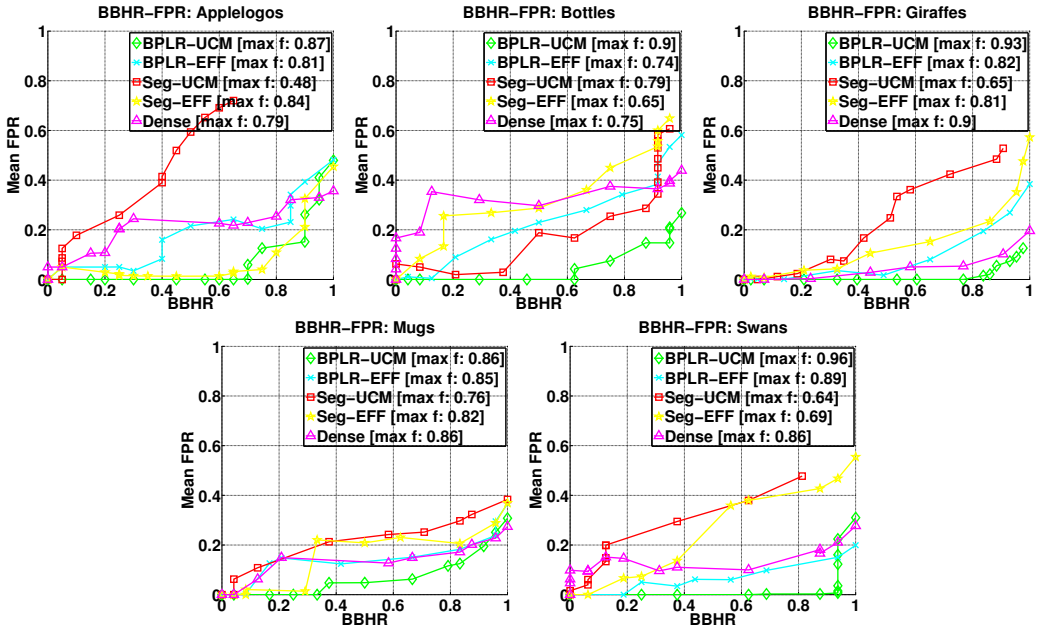


Figure 3.13: BPLR's quality in terms of BBHR-FPR from two different initial segmentations (Seg-UCM [3] and Seg-EFF [28]). Plots compare BBHR-FPR among the features. In addition to BPLRs and their initial seed segments, I show the result from dense sampling for comparison. For this BBHR-FPR metric, lower and longer curves are better.

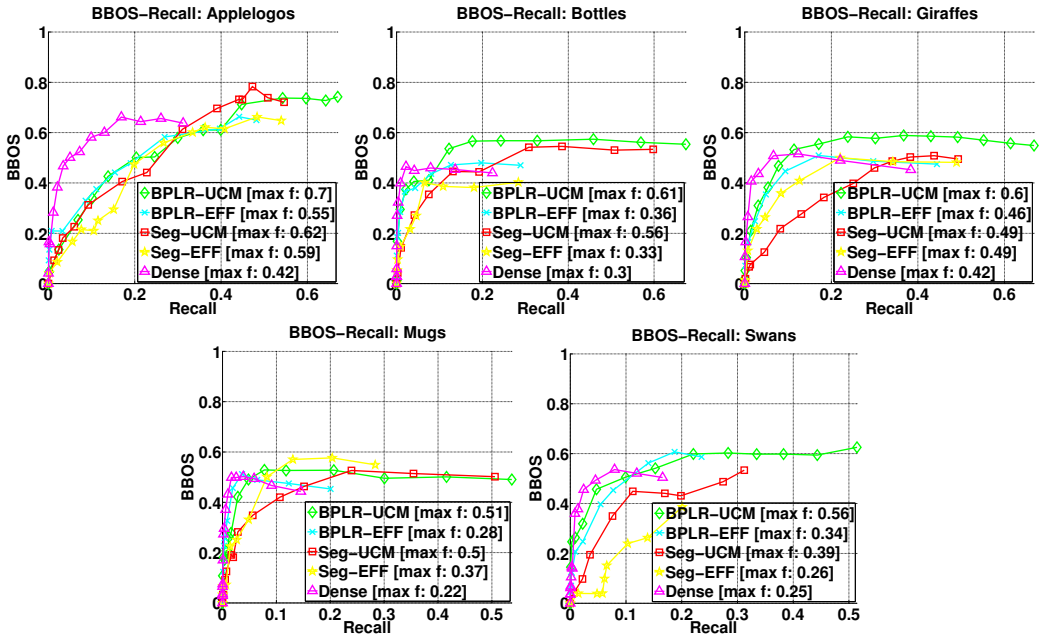


Figure 3.14: BPLR's quality in terms of BBOS-Recall from two different initial segmentations (Seg-UCM [3] and Seg-EFF [28]). Plots compare BBOS-Recall among the features. In addition to BPLRs and their initial seed segments, I put the result from dense sampling for comparison. For BBOS-Recall, higher and longer curves are better.

Feature	Mean BBDR
BPLR-UCM	**0.67**
BPLR-EFF	0.52
Seg-UCM	0.56
Seg-EFF	0.49
Dense	0.48

Table 3.2: Bounding Box Detection Rate (BBDR) from two different seed segmentations on ETHZ objects. We see that both BPLR-UCM and BPLR-EFF provide improvements over their base segments (Seg-UCM and Seg-EFF, respectively). While BPLR-UCM and Seg-UCM outperform BPLR-EFF and Seg-EFF, BPLR-EFF and Seg-EFF run an order of magnitude faster.

Feature	BBHR-FPR	BBOS-Recall	Mean BBDR
BPLR-UCM	**0.87**	**0.46**	**0.62**
BPLR-EFF	0.82	0.40	0.52
Seg-UCM	0.62	0.45	0.45
Seg-EFF	0.76	0.41	0.49
Dense	0.83	0.31	0.48
MSER	0.68	0.42	0.42

Table 3.3: Quality of features using the same underlying gradient image on ETHZ objects. For BBHR-FPR and BBOS-Recall, I show the average of maximum F-number across all the ETHZ object classes.

frequently used "superpixels" with BPLRs as base features. In the weakly-supervised foreground discovery problem [18, 1], the system is given a set of cluttered images that all contain the same object class, and must estimate which pixels are foreground.

I design a simple model for this task using BPLRs. It is much like the GrabCut [81] baseline defined in [1], in that I initialize a foreground color model from the central 25% of the images and a background color model from the rest, and then solve a standard graph-cut binary labeling problem. However, I replace the superpixel nodes used in [1] with our BPLRs, and add an additional term to the node potential based on the BPLR matches. The new term reflects that we prefer to label BPLR regions as foreground if they match well to other BPLRs in images of the same class (the assumption being that same-class backgrounds are uncorrelated). Specifically, let m_f denote the distance from a BPLR's descriptor to its nearest neighbor among the same-class images, and let m_b denote the distance to its nearest neighbor in the images from

Figure 3.15: Impact of BPLR matching on foreground likelihood. Red areas indicate where foreground likelihood exceeds that of background. The initial foreground color model is incorrect (2nd img), but BPLR matches to other bonsai images correctly predict the object location (3rd img). Combining the color model and BPLR matches (4th img), We obtain an accurate foreground estimate (last image).

Approach	Accuracy(%)
BPLR GrabCut (Ours)	**85.6**
Superpixel GrabCut [81]	81.5
Superpixel ClassCut [1]	83.6
Superpixel Spatial Topic Model [18]	67.0

Table 3.4: Foreground discovery results, compared to several state-of-the-art methods. Using BPLR regions with a GrabCut-based solution, I obtain the best accuracy to date on the Caltech-28 dataset. (See text for details.)

other classes; if $m_b - m_f$ is positive, I use it to adjust the color-based foreground likelihood (see Fig. 3.15). I average likelihoods wherever BPLRs overlap to obtain a single value per pixel. I test with the setup prescribed in previous work [1, 18], which uses 28 Caltech classes, 30 images each, and measures accuracy by the percentage of correctly classified pixels.

Table 3.4 shows the results. BPLR yields the best accuracy, showing its strength at capturing class-specific shapes in a highly repeatable manner. Our improvement over the GrabCut baseline directly isolates the contribution of BPLR matching (5% gain). Our improvements over the more elaborate models of [18, 1] suggest that even with a simpler labeling objective, BPLRs are preferable to the less-repeatable superpixel base features. Fig. 3.16 shows some example segmentations computed with my method.

Figure 3.16: Example foreground discovery results using BPLRs. Two examples per class. Ground truth is marked in red. BPLR matching cleanly separates objects from the background in most cases. In some cases, however, we see small leaks near object boundaries (e.g., see the ferry and butterfly), likely due to background regions abutting object boundaries that are confused by strong shape contours.

3.3.5 Object Classification with BPLR

Finally, I apply my features to object recognition on the Caltech-101 dataset. I again employ a relatively simple classification model on top of the BPLRs, to help isolate their impact. Specifically, I use the Naive Bayes Nearest-Neighbor (NBNN) classifier [11], which sums the NN feature match distances from a test image to those pooled among the training images of each class, and picks the class that produces the lowest matching distance. I follow standard procedures, using 15 random images per class to train and test respectively.

Table 3.5 compares our results to those using NBNN with alternative feature extractors. With the same HOG descriptor, my method outperforms the baseline (Segment) by a large margin. Furthermore, I make a 10% improvement over Dense+SIFT, the previous strongest feature choice for this task; while both extract a similar number of features, our shape-preserving features have a clear advantage over the uniform patch sampling.

Table 3.6 compares my results to existing single-feature NN-based results reported in the literature. BPLR offers noticeable gains over almost all such methods, even some that use learned metrics [38]. Overall, these results show that my shape-preserving dense features lead to more reliable matches than alternative extraction methods, and coupled with a very simple

Feature	Accuracy(%)
BPLR (Ours)	**61.1**
Dense+SIFT	55.2
Segment	37.6

Table 3.5: Direct comparison of BPLR to other feature detectors on the Caltech-101. My method provides the most accurate result.

Feature	Accuracy(%)
NBNN+BPLR (Ours)	61.1
NBNN+Dense SIFT [11]	65.0
AsymRegionMatch+Geom [46]	61.3
SVM-KNN [99]	59.1
GB+Learned distance [38]	58.4
Segment+Learned distance [38]	55.1
GB+Vote [8]	52
BergMatching [9]	48.0

Table 3.6: Comparison to existing results on the Caltech-101 that use nearest neighbor-based classifiers. Ours are among the leading results.[3]

model are quite effective for object classification.

3.4 Discussion

In this chapter, I introduced a dense local detector that produces repeatable shape-preserving regions via a novel segmentation-driven sampling strategy. As shown through extensive experiments, the key characteristics that distinguish BPLR from existing detectors are: 1) it can improve the ultimate descriptors' distinctiveness, while still retaining thorough coverage of the image, 2) it exploits segments' shape cues without relying on them directly to generate regions, thereby retaining robustness to segmentation variability, and 3) its generic bottom-up extraction makes it applicable whether or not prior class knowledge is available. As such, BPLR can serve as a useful new addition to researchers' arsenal of well-used local feature techniques.

In future work, a new descriptor for the detected BPLR can be explored to improve

[3]The authors of [11] report 65.0% when using dense SIFT with NBNN (as shown in Table 3.6); despite substantial effort, my implementation of this baseline yields only 55.2% (as shown in Table 3.5). I attribute the discrepancy to some unknown difference in the feature sampling rate or approximate neighbor search procedure parameters.

BPLR's match quality. To represent the detected BPLR regions, I currently use the existing PHOG descriptor computed from gPb contour map. While it represents the outline of the shape as well as (coarsely) its inner texture, it misses some valuable information from the BPLR extraction procedure such as topological structure of the spanning tree, and/or scale distribution of member elements within each BPLR. Alternative descriptor could fully encode the geometric layout of the extracted regions (e.g., encoding the graph topology within the BPLR), as well as incorporate invariance to rotation or scale.

Learning BPLR to detect important object parts or objects' characteristic geometry (e.g., symmetry) would also be interesting. New applications built on BPLR would be another promising venue. Some recent works [55, 21] that apply BPLR to robust shape matching for segmentation or to geometric grouping for local feature matching suggest such possibility.

So far I have shown how to extract both repeatable as well as distinctively-shaped local regions with the proposed BPLR detector. Building on the strength of the BPLRs, in the following chapter I will explore an approach for object-level segmentation, in which I employ strong local shape matches via BPLRs to estimate global object shapes.

Chapter 4

Object Segmentation with Shape Sharing

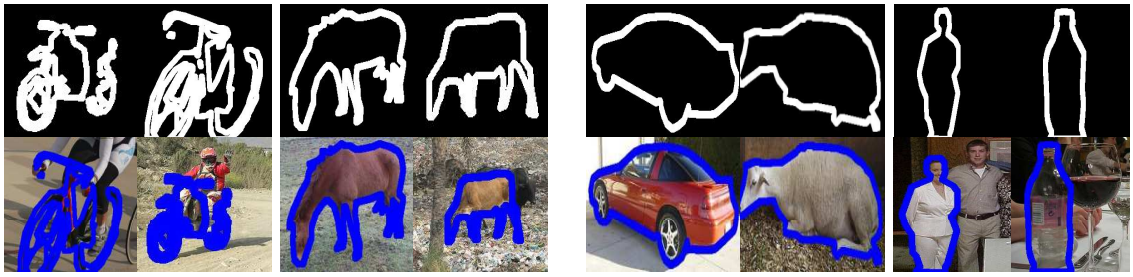
Building on the strength of the BPLR detector developed in the previous chapter, I next introduce an approach for object-level segmentation, which is published in [48].[1] For object segmentation, shape cues are critical to group diverse object parts that would not be merged if judging color or texture alone. To exploit shapes for segmentation, I propose *shape sharing*. Shape sharing is based on the intuition that object shapes are (at least partially) shared across categories. Integrating shape sharing with local shape matching via BPLRs, I devise an exemplar-based category-independent shape prior and apply it for generating object-level segmentations.

4.1 Motivation: Shape Sharing for Segmentation

Bottom-up image segmentation methods group low-level cues from color, texture, and contours to estimate the boundaries in an image. Despite significant strides in recent years, it is widely acknowledged that a bottom-up process alone cannot reliably recover object-level segments. Pitfalls include the fact that a single object is often comprised of heterogenous textures and colors, objects with similar appearance can appear adjacent to one another, and occlusions disrupt local continuity cues—all of which lead to over- or under-segmented results. This is a fatal flaw for downstream recognition processes.

As a result, researchers have explored two main strategies to move beyond low-level cues, as discussed in Chapter 2. The first strategy expands the output to produce *multiple*

[1] Code and data are available online: http://vision.cs.utexas.edu/projects/shapesharing



(a) Semantically *close* (b) Semantically *disparate*

Figure 4.1: Intuition for shape sharing. While one may expect shape sharing between objects of semantically close categories (a), we observe that even among semantically disparate objects, partial shape sharing occurs (b). This suggests exploiting partial shape matches as *category-independent* shape priors.

segmentation hypotheses, typically by using hierarchical grouping, varying hyperparameters, or merging adjacent regions (e.g., [40, 67, 19, 24]). Enlarging the set of segments increases the chance of "hitting" a true object; however, large pools of candidate regions are costly to compute and maintain, and, more importantly, existing methods lack a model of global shapes. The second strategy introduces top-down *category-specific priors*, unifying bottom-up evidence with a preference to match a particular object's layout, shape, or appearance (e.g., [12, 58, 95, 20, 52, 17, 98]). Such methods elegantly integrate segmentation and recognition, yet they rely heavily on a known (pre-trained) class model. Category-specific shape priors in particular make strong assumptions about the viewpoint of the object to be segmented.[2]

At the surface, the goals of these two existing strategies seem to be in conflict: the former maintains category-independence, while the latter enforces top-down shape knowledge. My idea is to reconcile these competing goals by developing a category-independent shape prior for segmentation. The main insight is that many objects exhibit partial shape agreement—and this "shape sharing" occurs even across seemingly disparate categories (see Figure 4.1). Thus, rather than learn a narrow prior good on only the known class of interest, we aim to derive shape priors in a *category-independent* manner.

[2] In this chapter, we use *shape* to refer to the outer contours or boundaries of objects.

Why should category-independent shape priors be feasible? Intuitively, we know that closely related objects often have global shape similarity, such as similarly structured animals or variants of some vehicle (Figure 4.1(a)). Beyond these expected cases, however, we also observe that shape sharing occurs *across* category divisions, such as a bottle and a standing person (Figure 4.1(b)). In fact, an initial study of 4,200 object instances in the PASCAL dataset reveals that 58% of the time, an object finds its best shape match to some class outside of its own! In some sense, sharing across categories should not be a surprise, as we know from the common Gestalt properties underlying mid-level grouping, or classical part-based object models theorized in psychology [10].

The key, of course, is how to effectively exploit this intuition. I propose a non-parametric shape prior that generates multiple segmentation hypotheses for an input image. Given a novel unsegmented image, we first identify any strong local shape matches it has with shapes in a database of segmented exemplars. Based on the scale and position of each local match, we project the associated exemplar shapes into the test image. This effectively maps local support into global shape hypotheses without assuming any category-specific knowledge, since the database need *not* contain exemplars of the same object class(es) as our test image. Next, we aggregate partially overlapping hypotheses, in order to allow for partial sharing with multiple exemplars (e.g., a test object that shares its left side with a car exemplar, and its right side with a bus exemplar). Each such aggregated hypothesis yields a shape prior that suggests regions that would not be considered if judging color/texture/edges alone. Finally, we perform a series of figure-ground segmentations using graph cuts, enforcing each of the shape priors in turn. Figure 4.2 summarizes my approach.

Because our approach is exemplar-based, we capture shape sharing in a data-driven manner, avoiding the need to hand-craft the geometric primitives that objects are expected to share. Furthermore, the example-based design is central to overcoming the viewpoint sensitivity of existing shape priors; by allowing our exemplars to span arbitrary object classes and arbitrary

poses, the method can pick and choose partially shared shapes freely.

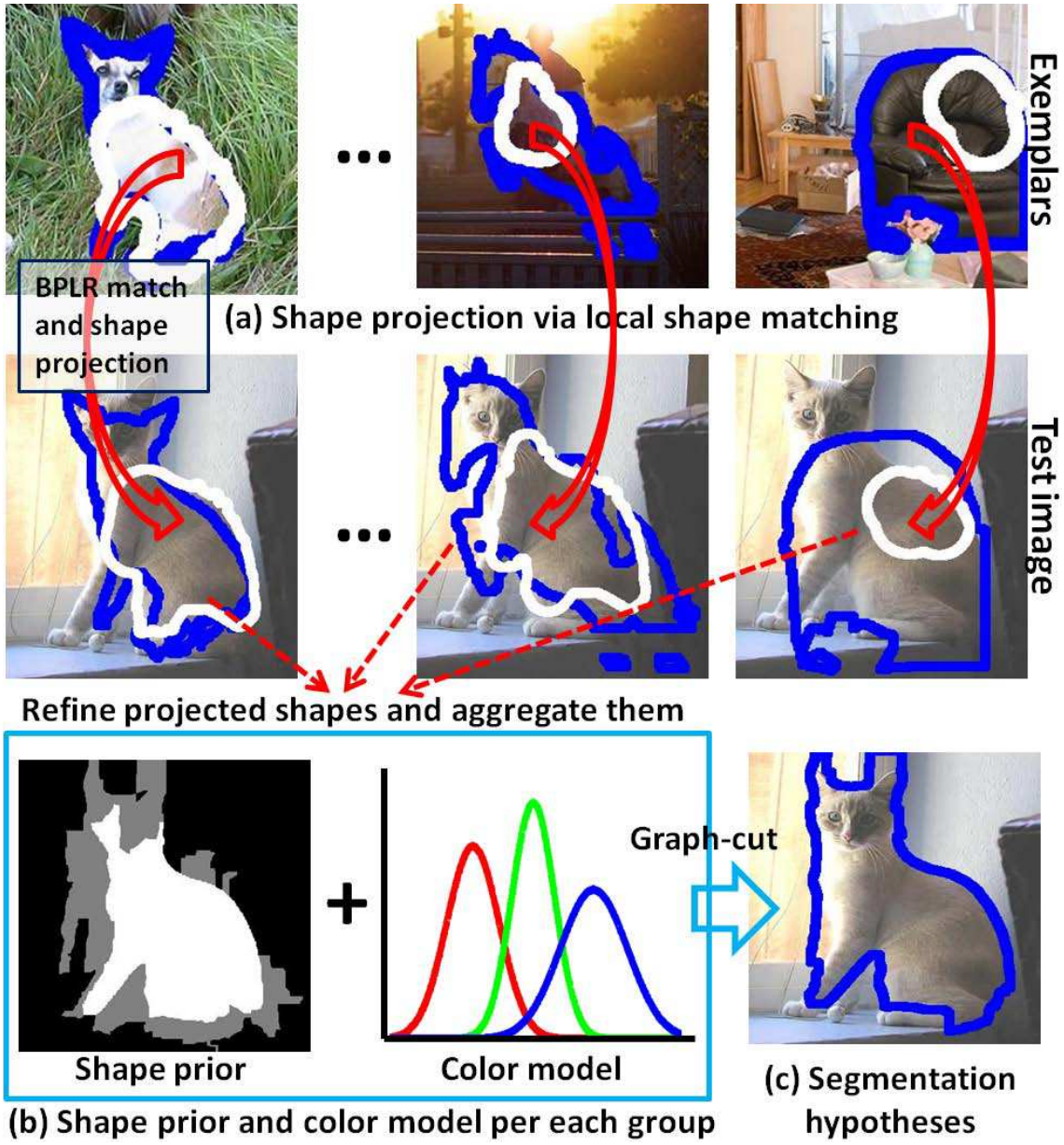


Figure 4.2: Overview of the proposed Shape Sharing segmentation. (a) Exemplars (first row) that partially share shape with the test image (second row) are projected in, no matter their category. (b) Multiple exemplars that partially agree are aggregated, to form a shape prior and color model. (c) The priors are used to compute a series of graph-cut segmentation hypotheses (only one is shown here).

4.2 Generating Object Segmentation Hypotheses

In this section, I present my approach to obtain multiple figure-ground object segmentations using category-independent shape priors derived from shape sharing. The input to the method is an unsegmented image containing unknown object categories, and the output is a set of region hypotheses (which may overlap). The method is successful to the extent that the hypotheses contain regions that highly overlap with true object boundaries.

The proposed approach consists of three main steps: 1) estimating global object shape in a test image by projecting exemplars via local shape matches (Sec. 4.2.1), 2) aggregating sets of partially aligned projected shapes to form a series of hypothesized shape priors (Sec. 4.2.2), and 3) enforcing the priors within graph-cuts to generate object segment hypotheses (Sec. 4.2.3).

4.2.1 Projecting Global Shapes from Local Matches

Suppose we have a database of manually segmented exemplars of a variety of objects.[3] For each exemplar object, we extract a set of distinctive local region features. We use the Boundary-Preserving Local Region (BPLR) to detect the local regions as described in the previous chapter.

Given a test image, the goal is to identify with which exemplars it shares shape. We first extract BPLRs throughout the test image, generating a dense set of local regions ($\sim$1,000 per image). Then, we match each BPLR in the test image to the exemplar database by finding its $k = 5$ nearest neighbor descriptors among all of the exemplars' BPLRs. For each such *local* match, we project the associated exemplar's *global* outer boundary shape into the test image based on the similarity transform computed between the two matched features (see Figure 4.2(a)). Due to the density of the BPLR detector, we will establish thousands of such initial global shape projections per test image.

[3]To be concrete, in our implementation we use the PASCAL Segmentation Taster data as the exemplar database, which contains thousands of object shape annotations from 20 different categories.

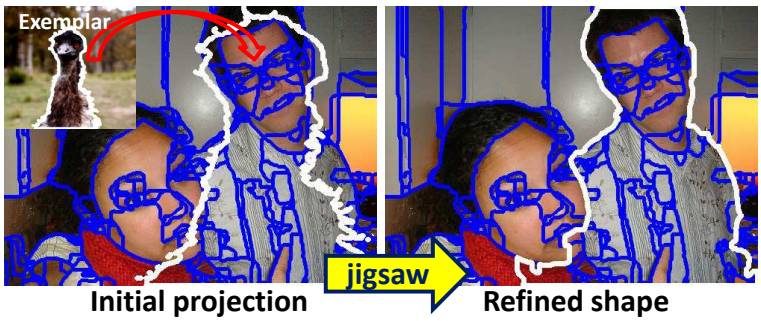
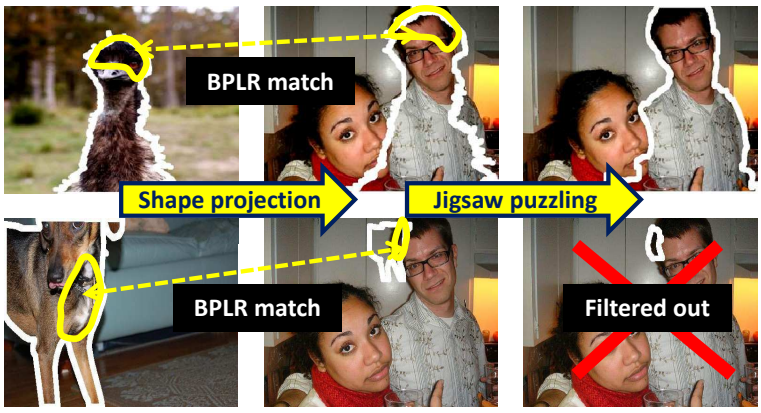


Figure 4.3: Shape refinement by jigsaw puzzling the superpixels underlying the exemplar's projection. The projected shape (marked in white) is adapted to fit the bottom-up contours suggested by underlying superpixels (marked in blue).

The projected shapes will agree at least locally with the test image; however, due to shape deformations and uncertainty in the local match, they need not be entirely aligned with the test image's contours. Therefore, we next want to snap the projected shape to align with bottom-up evidence of boundaries. To this end, we refine the initial projection boundary to span the "jigsaw" of underlying superpixels that overlap the global shape by more than half their total area. In this way, the exemplar shape is adapted to fit the observed contours. Figure 4.3 shows an example jigsaw puzzling, where we see that initial contour from an exemplar image (i.e., bird image) fits the actual object boundary in the test image (i.e., person image).

Finally, to reduce the total number of projections, we eliminate those whose shape changes substantially after the refinement process. Specifically, we rank the projections by the pixel-level overlap between the original exemplar's projection and the jigsaw refined version, and keep the top-ranked 600 projections. Essentially this weeds out unreliable projections that lack bottom-up support in the test image (see Figure 4.4 for an example). Thus, we go from about 5,000 candidate global projections to about 600.

The novel element important to my approach is the idea of generating *global* boundary hypotheses from *locally* shared shapes. The partial shape match via BPLR regions is particularly well-suited for transferring local supports to global shape hypotheses. Further, these shape predictions are made in a category-independent manner, requiring no prior knowledge of the



(a) Exemplar shape (b) Projected shape (c) Refined shape

Figure 4.4: Visualization of global shape estimation via local matches. Exemplar shapes (a) are projected into a test image (b) using the scales and positions of BPLR matches, and then refined to fit the image contours (c). Here we see one such match and projection in each row. Top row: the ostrich shares shape with the man's upper body. Bottom row: an unreliable projection that is discarded, since the dog's shape changes too dramatically during refinement (compare white outlines in (b) and (c); pixel-level overlap between the projection (b) and its refinement (c) is very low)

object present in the test image. In fact, it is irrelevant to our method whether or not the exemplar shapes have labels; their value is solely in providing a non-parametric prior on what kinds of shapes objects take on.

4.2.2 Aggregating Partially Shared Shapes

At this point, we could simply treat each of the global shape projections computed above as an individual shape prior; in fact, we find that alone they provide a reasonable prior (see Table 4.1 in results). However, doing so would not account for the fact that objects in the test image are likely to share shapes only *partially* with various exemplars—at least when we cannot assume a very large database of segmented exemplars. Therefore, we next aim to group together those global shape projections that partially agree on an object's extent in the test image. The idea is for each projection to contribute a portion of its contour to an aggregate shape prior (e.g., see the matched exemplars in Figure 4.2(a), each of which partially shares shape with the cat in the test image).

To determine which projections to aggregate, we use a simple but effective heuristic: any projections whose pixel overlap exceeds 50% are grouped. Each such group is used to construct one shape prior consisting of two parts: one that prefers including those pixels in the test shape that are shared by the contributing exemplar projections, and one that extracts a color model using their predicted shape. See Figure 4.2(b). Both parts enforce the shape prior in a graph-cut figure-ground segmentation, as we explain next.

4.2.3 Graph-Cut Segmentation with the Shape Prior

The final step is to enforce the non-parametric priors when computing the output region hypotheses. We define an energy function amenable to graph-cuts optimization [15] that reflects the quality of a given figure-ground segmentation according to its agreement with the shape prior. We optimize this function independently for each group (prior) defined above, yielding one set of region hypotheses per group.

Treating each pixel p_i in the image as a node, graph-cut optimizes their labels $y_i \in \{0 \text{ (bg)}, 1 \text{ (fg)}\}$ by minimizing an energy function of the form:

$$E(y) = \sum_{p_i \in \mathcal{P}} D_i(y_i) + \sum_{i,j \in \mathcal{N}} V_{i,j}(y_i, y_j), \tag{4.1}$$

where $\mathcal{P}$ denotes all pixels, $\mathcal{N}$ denotes pairs of adjacent pixels, $V_{i,j}$ is a smoothness function, and D_i is a data term. Note that this follows the basic graph-cut segmentation formulation [81, 15]; what is new is how we encode a non-parametric shape prior into the data term.

Data term Typically, the data term D_i is a function of the likelihood of labeling pixel p_i as foreground or background. In our formulation, it consists of two parts: a shape-prior likelihood S_i and a color likelihood C_i:

$$D_i(y_i) = S_i(y_i) + C_i(y_i). \tag{4.2}$$

The shape-prior term S_i defines the likely spatial extent of the foreground and background. Given one group from Sec. 4.2.2, we first compute the intersection $\mathcal{I}$ and union $\mathcal{U}$ of its com-

ponent shape projection regions. Then we define the cost of labeling a pixel as foreground to be:

$$S_i(y_i = 1) = \begin{cases} 0.5 - \gamma & \text{if } p_i \in \mathcal{I} \\ 0.5 + \gamma & \text{if } p_i \notin \mathcal{U} \\ 0.5 & \text{if } p_i \notin \mathcal{I} \text{ and } p_i \in \mathcal{U}, \end{cases} \tag{4.3}$$

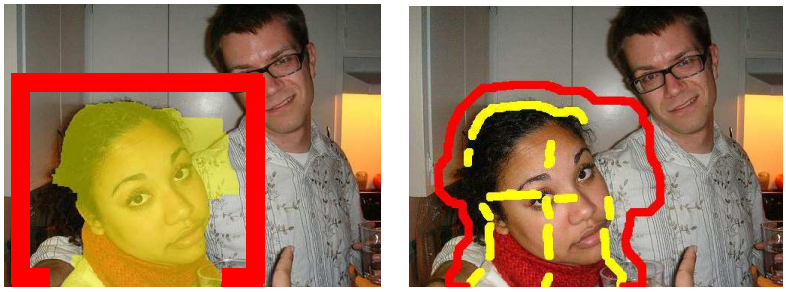
where $p_i \in \mathcal{I}$ and $p_i \in \mathcal{U}$ denote a pixel inside the intersection and union of the projections, respectively, and γ is a positive constant value used to adjust the impact of the shape prior (and will be defined below). The cost of assigning the background label is simply the inverse: $S_i(y_i = 0) = 1 - S_i(y_i = 1)$. Intuitively, this likelihood prefers a pixel inside the intersection region to be labeled as foreground, since all of the projections in the group agree that the pixel belongs in the shape. In contrast, it prefers a background label for pixels outside the union region, since none of the projections predict the pixel to belong to the shape (i.e., no sharing). Pixels in the union but outside of the intersection are treated as neutral, with no bias towards either foreground or background, as reflected by the third line in Eqn. 6.3. The white and gray pixels in Figure 4.2(b) depict these foreground biased and "don't care" regions of the shape prior, respectively.

The color likelihood term C_i also relies on the shape projections, but in a different way. Whereas S_i biases pixel memberships based on the span of the shared shapes, C_i uses the shared shape to estimate a color distribution for the hypothesized object. Let H_f and H_b denote normalized color histograms sampled from the shared shape region for the foreground and background, respectively. We define the color likelihood cost as:

$$C_i(y_i) = \frac{1}{1 + \exp(\beta W_i(y_i))}, \tag{4.4}$$

where $W_i(p_i)$ is a function of the color affinity between pixel p_i and the histograms, and β is a normalizing constant. Let $c(p_i)$ denote the histogram bin index of the RGB color value at pixel p_i. The color affinity rewards assigning the background label to pixels more likely to be generated by the background color distribution:

$$W_i(y_i = 0) = H_b(c(p_i)) - H_f(c(p_i)), \tag{4.5}$$



(a) Region-based seeds (b) Contour-based seeds

Figure 4.5: The two methods for constructing H_f and H_b histograms. Yellow: fg seeds, Red: bg seeds. Best viewed in color.

and vice versa: $W_i(y_i = 1) = -W_i(y_i = 0)$. The sigmoid in Eqn. 6.4 serves to scale the color likelihoods between 0 and 1, making them compatible with the shape-prior values S_i.

We devise two complementary ways to sample pixels from the shared shape in order to form H_f and H_b: one that uses *region-based seed pixels*, and one that uses *contour-based seed pixels*. For region-based seed pixels, H_f is computed using all pixels inside the intersection $\mathcal{I}$ of the shape projections, and H_b is computed using pixels falling within a thick rectangular border surrounding the intersection region. See Figure 4.5(a). For contour-based seed pixels, we instead form H_f using pixels just along the boundary of $\mathcal{I}$ and along its primary medial axes within the shape, and we compute H_b using pixels along the boundary of a dilated version of the same intersection region. See Figure 4.5(b).

The two seeding methods work in complementary ways. Region-based seeding provides dense coverage of pixels, and thus reflects the full color distribution of the shape prior's region. However, when the shape prior is flawed—for example, if it leaks into the background, as shown in Fig. 4.5(a)—then its estimate can be distorted. On the other hand, contour-based seeding respects the object shapes, and is motivated by how users tend to manually give seeds for interactive segmentation [81]. However, being sparser, it may lack sufficient statistics to estimate the color distribution. We use each of these seeding strategies separately when generating the pool of segmentations (see below).

Smoothness term Our smoothness function $V_{i,j}$ follows the conventional form, e.g., [81]: the cost of assigning different labels to neighboring pixels is inversely proportional to the strength of the contour at that position.

$$V_{i,j}(y_i, y_j) = \begin{cases} 0 & \text{if } y_i = y_j \\ \exp(-\sigma \max(g(y_i), g(y_j)) & \text{if } y_i \neq y_j, \end{cases} \tag{4.6}$$

where σ is a normalizing constant, and g returns contour strength by gPb contour detector [4] at a pixel.

Solving multiple graph-cut problems Having defined the complete energy function $E(y)$, we can now compute the optimal binary labeling using graph-cuts. For each group of projections resulting from Sec. 4.2.2, we solve *multiple* instances of the problem by varying the weighting constants and color histogram seeding strategies. This yields multiple segment hypotheses for a given prior

Specifically, we vary (1) the value of γ in Eqn. 6.3, which adjusts the influence of the shape prior relative to the color likelihood, (2) whether region-based or contour-based seeding is used, which adjusts the definition of C_i in Eqn. 6.4, and (3) the value of a foreground bias constant λ in the data term. For the latter, we modify the data term D_i as follows:

$$D_i(y_i, \lambda) = \begin{cases} D_i(y_i) + \lambda & \text{if } y_i = 1 \\ D_i(y_i) - \lambda & \text{if } y_i = 0. \end{cases} \tag{4.7}$$

Positive values of λ decrease the foreground bias, while negative values increase the foreground bias.

Thus, the total number of hypotheses for the given group is ($\#\gamma$ values) $\times 2 \times$ ($\#\lambda$ values); we use 2 and 8 values of γ and λ in our experiments, respectively. Note that increasing the pool of segments naturally will increase recall of true object shapes, but at the penalty of greater complexity. Algorithm 2 provides the pseudo-codes that describe the whole steps for shape sharing segmentation.

Algorithm 2: Shape sharing segmentation

Data: Superpixels by multiple segmentations in a test image $\mathbf{S_{sp}}$; BPLRs extracted from a test image $\mathbf{B_t}$; BPLRs extracted from exemplar images in database $\mathbf{B_{db}}$; Ground-truth object segment annotations in exemplar images $\mathbf{S_{gt}}$

Result: A set of object segment hypotheses in a test image $\mathbf{S_{obj}}$

`/* Shape projection */`

input : $\mathbf{S_{sp}}$, $\mathbf{B_t}$, $\mathbf{B_{db}}$, and $\mathbf{S_{gt}}$

output: A set of initial object shape projections $\mathbf{S_{init}}$ in a test image

foreach *bplr* b_i *in* $\mathbf{B_t}$ **do**

  `/* Matching BPLRs */`

  Find k nearest BPLR matches $m_j (j = 1, ..., k)$ in $\mathbf{B_{db}}$;

  `/* Shape projection based on BPLR matches */`

  for $j = 1$ **to** k **do**

    Retrieve the ground-truth object shape GT_j to which m_j belongs;

    Compute a similarity transform between b_i and m_j;

    Apply the similarity transform to GT_j;

    Project the transformed shape into the test image;

    Add the projected shape into the set of initial shape projections $\mathbf{S_{init}}$;

  end

end

`/* Shape refinement using jigsaw */`

input : $\mathbf{S_{init}}$

output: A set of refined shape projections $\mathbf{S_{refine}}$

foreach *a projected shape* $proj_i$ *in* $\mathbf{S_{init}}$ **do**

  Compute overlap between $proj_i$ and superpixels in $\mathbf{S_{sp}}$;

  Pick superpixels that overlap $proj_i$ more than 50% of their area;

  Merge the picked superpixels to form a refined shape;

  Add the refined shape into the set of refined shape projections $\mathbf{S_{refine}}$;

end

`/* Selecting reliable projections */`

input : $\mathbf{S_{refine}}$

output: A pruned $\mathbf{S_{refine}}$

foreach *a refined shape* s_i *in* $\mathbf{S_{refine}}$ **do**

  Compute overlap with its original initial projection in $\mathbf{S_{init}}$;

  if *the overlap is less than a threshold* **then**

    Remove s_i from $\mathbf{S_{refine}}$;

  end

end

Algorithm 2: Shape sharing segmentation: continued

```
/* Grouping the projections                                          */
```
input : A pruned $\mathbf{S_{refine}}$
output: A set of groups of shape projections $\mathbf{G_{proj}}$
Compute average contour strength along the boundary of shape projections in $\mathbf{S_{refine}}$;
Sort the shape projections in a decreasing order of boundary contour strength;
Remove projections of weak boundaries and retain top N projections in $\mathbf{S_{refine}}$;
Compute overlap between all pairs of projections in $\mathbf{S_{refine}}$;
foreach *a shape s_i in the sorted* $\mathbf{S_{refine}}$ **do**
  if $s_i \notin \mathbf{S_{refine}}$ **then**
    continue;
  end
  Pick a set of projections in $\mathbf{S_{refine}}$ that overlap s_i more than a threshold;
  Form a group G_i with the selected projections and s_i;
  Remove the projections in G_i from $\mathbf{S_{refine}}$;
  Add G_i into $\mathbf{G_{proj}}$;
end

```
/* Compute segmentations for each group                              */
```
input : $\mathbf{G_{proj}}$ and $\mathbf{S_{refine}}$
output: Multiple object segment hypotheses $\mathbf{S_{obj}}$
foreach *a group of shape projections G_i in* $\mathbf{G_{proj}}$ **do**
  `/* See 4.2.3 for details */`
  Build shape and color model for G_i;
  Solve multi-parametric graph-cuts for the model to obtain object segmentations;
  Add the obtained segmentations into the $\mathbf{S_{obj}}$;
end

4.3 Results

In this section, I present experimental results of my Shape Sharing object segmentation method. The main goals of the experiments are 1) to demonstrate that shape sharing improves the quality of the segmentation (Sec. 4.3.1), 2) to analyze under what conditions shapes are useful for segmentation (Sec. 4.3.2), and 3) to validate the impact of our category-independent shape priors compared to traditional category-dependent ones (Sec. 4.3.3).

Datasets and implementation details To build the exemplar database, we again use the PASCAL 2010 Segmentation training data, which has pixel-level annotations for 2,075 objects

(a) BSD300 (b) PASCAL 2010

Figure 4.6: Example images from the datasets used in the experiments. For each dataset, we randomly display one image from each object class of the dataset. The ground truth consists of the true object boundary of one or more primary object in the image.

from 20 classes. We extract 1,000-2,000 BPLRs from each exemplar, and represent them with pHOG+gPb descriptors, which capture both boundary shape and coarse inner texture. To efficiently identify nearest neighbor matches, we use FLANN [71]. For superpixels, we use the output of gPb-owt-ucm [3].

We test on two datasets: the PASCAL 2010 validation set and the Berkeley BSD300 dataset. For BSD, we use the ground truth region annotations given by [24]. Figure 4.6 shows some example images from each dataset. Note that for both test sets, we use the same PASCAL exemplars. This allows us to demonstrate cross-dataset sharing.

Evaluation metrics To evaluate segmentation quality, we use the *covering metric*, following [3, 19], which is the average best overlapping score between ground-truth and generated segments, weighted by object size. Note that due to the use of "best overlap" in the covering metric, a method that achieves higher covering for fewer segments has better focused its results on true object regions. We also report *recall as a function of overlap*, following [24], to quantify the percentage of objects recalled at a given covering score.

Approach	Covering (%)	Num segments
Exemplar-based merge (Ours)	**77.0**	607
Neighbor merge [67]	72.2	5005
Bottom-up segmentation [3]	62.8	1242

Table 4.1: Our shape-based projection and merging approach outperforms an existing merging strategy while requiring an order of magnitude fewer segments (second row). It also substantially improves the state-of-the-art bottom-up segmentation (third row).

4.3.1 Segmentation Quality

First I investigate how useful shape sharing is to improve segmentation accuracy, by comparing our results to those of several state-of-the-art techniques [3, 67, 19, 24].

Shape prediction via local matches First I evaluate the quality of our exemplar-based shape predictions via local shape matching (i.e., the first stage of our method defined in Sec. 4.2.1). I compare against two existing methods on the PASCAL data: 1) a merging method that combines pairs and triples of neighboring superpixels, without considering layout or shape [67], and 2) the state-of-the-art gPb-owt-ucm segmentation algorithm [3]. Note that the number of segments output by each method will vary, and in general, a high covering score accompanied by a low number of segments is ideal.

Table 4.1 shows the results. Our method clearly outperforms the previous methods, while also maintaining a much smaller number of segments. The results demonstrate that merging bottom-up segments increases the chance of hitting true object (compare ours and [67] to the bottom-up method [3]). Further, our shape prediction enhances the segmentation quality with a much smaller number of hypotheses, compared to a brute-force merging of nearby segments [67]; this confirms the ability of shape sharing to predict the objects' spatial extent.

Shape Sharing with graph-cuts Next we compare our full approach to existing segmentation methods, including the state-of-the-art category-independent object segmentation generators of

Approach	Covering (%)	Num segments
Shape Sharing (Ours)	**84.3**	1448
CPMC [19]	81.6	1759
Object proposals [24]	81.7	1540
gPb-owt-ucm [3]	62.8	1242

Table 4.2: Accuracy on the PASCAL 2010 dataset. Ours outperforms the competing methods that lack global shape cues; this demonstrates the impact of the proposed category-independent shape prior.

Approach	Covering (%)	Num segments
Shape Sharing (Ours)	**75.6**	1449
CPMC [19]	74.1	1677
Object proposals [24]	72.3	1275
gPb-owt-ucm [3]	61.6	1483

Table 4.3: Accuracy on the BSD300 dataset. The strength in the BSD dataset shows our exemplar-based approach is generalized among various objects of unrelated categories, since we use exemplars from the PASCAL dataset that is disjoint from the BSD dataset.

Constrained Parametric Min-Cuts (CPMC) [19] and Object proposals [24]. We use the code kindly provided by the authors.[4] To focus on raw segmentation quality, we do not consider post-processing with a learned region-ranking function (as in [19], [24]), which could equally benefit all methods, in terms of the number of segments.

Tables 4.2 and 4.3 show the results on PASCAL and BSD, respectively. Our approach outperforms the existing methods overall; it is also more accurate for 18 of the 20 PASCAL classes for the next best method (see Table 4.4). Since all three previous methods rely on only color and/or local appearance and layout cues, this result validates the impact of global shape priors for segmentation.

The strength of our method on BSD—for which we use PASCAL images as exemplars—is strong evidence that shape sharing is generalized among various objects of unrelated categories. Even the PASCAL test results illustrate category-independence, since the exemplars

[4] In order to isolate the impact of a color-based graph-cut likelihood, for [19], we select an option in the author's code to forgo graph-cut outputs with uniform foreground bias, which do not rely on image cues.

Class	Shape Sharing	CPMC [19]	Object proposals [24]
bus	**87.6**	78.4	83.7
cow	**90.4**	82.0	86.4
car	**84.3**	77.5	81.5
tv	**88.6**	82.2	83.4
train	88.3	82.4	**88.9**
boat	**84.2**	78.4	79.8
bottle	**87.2**	81.8	77.9
table	**85.0**	79.9	83.4
person	**83.2**	78.1	79.4
chair	**83.5**	79.1	79.4
motorbike	80.0	75.9	**80.7**
bird	**91.7**	87.7	87.5
sheep	**90.1**	86.7	83.9
dog	**88.0**	84.5	83.4
horse	**83.7**	81.0	81.7
sofa	**86.7**	84.6	84.5
cat	**89.3**	87.5	87.7
plant	**82.4**	81.2	81.2
plane	**88.3**	88.0	87.0
bicycle	68.6	**69.3**	69.1
mean	**85.6**	81.3	82.5

Table 4.4: Average covering score per class for the PASCAL 2010 dataset. We compare ours to two state-of-the-art category-independent region generation methods. Our method outperforms the others for 17 of the 20 classes (For legibility, we sort the classes in the order of our gains over the CPMC method). We can see that classes with regular shapes (e.g., bus, car, tv, train) obtain the largest gains over the existing methods, as do classes that share shapes (e.g., bottle and person, different animal classes). In contrast, classes with very unusual shapes (e.g., potted plant) or with thin-structured details (e.g., bicycle) yield the smallest gains.

matched to test images can and often do come from different categories. Figure 4.7 shows the sharing strength between the PASCAL object classes. We see that shape sharing often occurs among semantically close categories (e.g., among animals or vehicles). However, sharing also happens between semantically disparate classes (e.g., bottle and person). In Sec. 4.3.3 below we further explicitly isolate the category-independent exemplar matches.

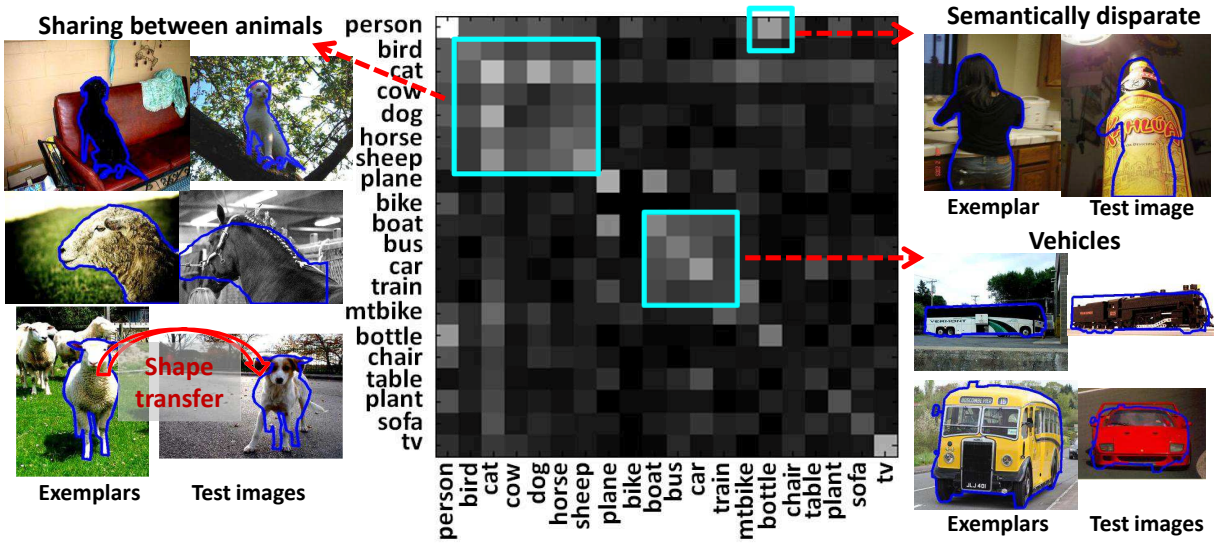
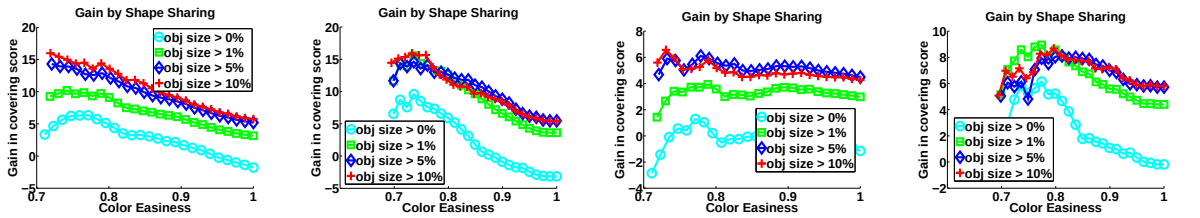


Figure 4.7: Shape sharing matrix for the 20 classes in PASCAL. We compute the strength of sharing by counting how many times shape exemplars from one class are used to generate the best segmentation hypotheses for another class. Brighter color denotes stronger sharing between categories. As one can expect, shape sharing often occurs among semantically close classes (e.g., animals or vehicles). However, sharing also happens between semantically disparate classes such as bottle and person. On the other hand, some classes (e.g., person and airplane) have very class-specific shapes, and so exhibit sharing to a lesser extent.

4.3.2 Impact of Shapes

The total gain in covering score in the previous section was about 2-3 points over the next best method, which may seem modest at a glance. However, since it is the average over all test cases and all classes, it does not fully reveal the impact of shape sharing. Therefore, we now examine in detail under what conditions our shape prior benefits segmentation. We expect shape to serve a complementary role to color, and to be most useful for objects that consist of multiple parts of diverse colors, and for objects that are similarly colored to nearby objects and the background.

To validate this hypothesis, we introduce a measure of *color easiness*, such that we can rank all test images by their expected amenability to color-based segmentation. We define color easiness by building foreground and background color histograms using pixels from inside and outside the ground truth object boundaries, respectively, and then count how many pixels in



(a) CPMC [19], PASCAL (b) CPMC [19], BSD (c) Object proposals [24], PASCAL (d) Object proposals [24], BSD

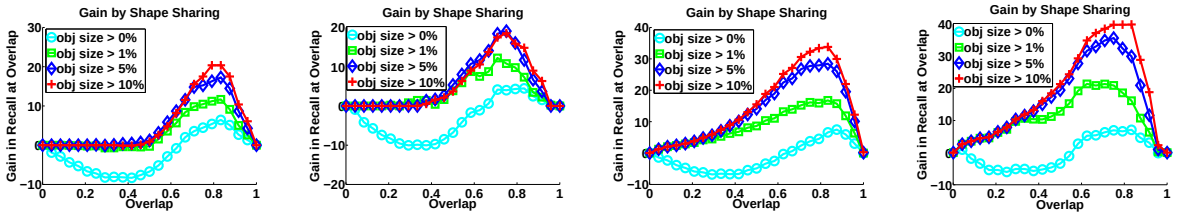
Figure 4.8: Impact of Shape Sharing as a function of "color easiness". When color alone is most confusing (lower easiness), Shape Sharing shows the greatest gains in segmentation accuracy.

the object's bounding box would be correctly labeled if using only their distance to the two histograms. The more correctly labeled pixels, the higher the color easiness for that test image.

Figure 4.8 plots Shape Sharing's accuracy gain over the baselines, as a function of color easiness (x-axis) and object size (multiple curves per plot). We see clearly that the most impressive gains—up to about 15 points in raw covering score—indeed occur when color easiness is lowest, for both datasets. The trend with color easiness is especially pronounced in the comparison to [19] (see (a) and (b)), which makes sense because its cues are strictly color-based. In contrast, compared to [24] the trend is a bit flatter, since that method uses not only color but also a local layout cue (see (c) and (d)). Still, our gains are substantial over both methods.

Figure 4.8 also reveals that Shape Sharing most benefits the segmentation of larger objects. We attribute this to a couple factors. First, shapes become more evident in the image as object size increases, since there is sufficient resolution along the boundary. Second, since larger objects tend to have various parts with diverse colors (e.g., a close-up of a person wearing differently colored pants and shirt), shape becomes more critical to combine the disparate parts. On the other hand, Shape Sharing has little impact (and can even hurt accuracy) for the smallest objects that occupy less than 1% of the image. This is because local matches are missed on the tiny objects, or the scale change computed from the local match becomes unreliable.

Figure 4.9 plots Shape Sharing's gain in recall as a function of overlap score, where recall records what percentage of objects have a best overlap score over the given threshold. Ours



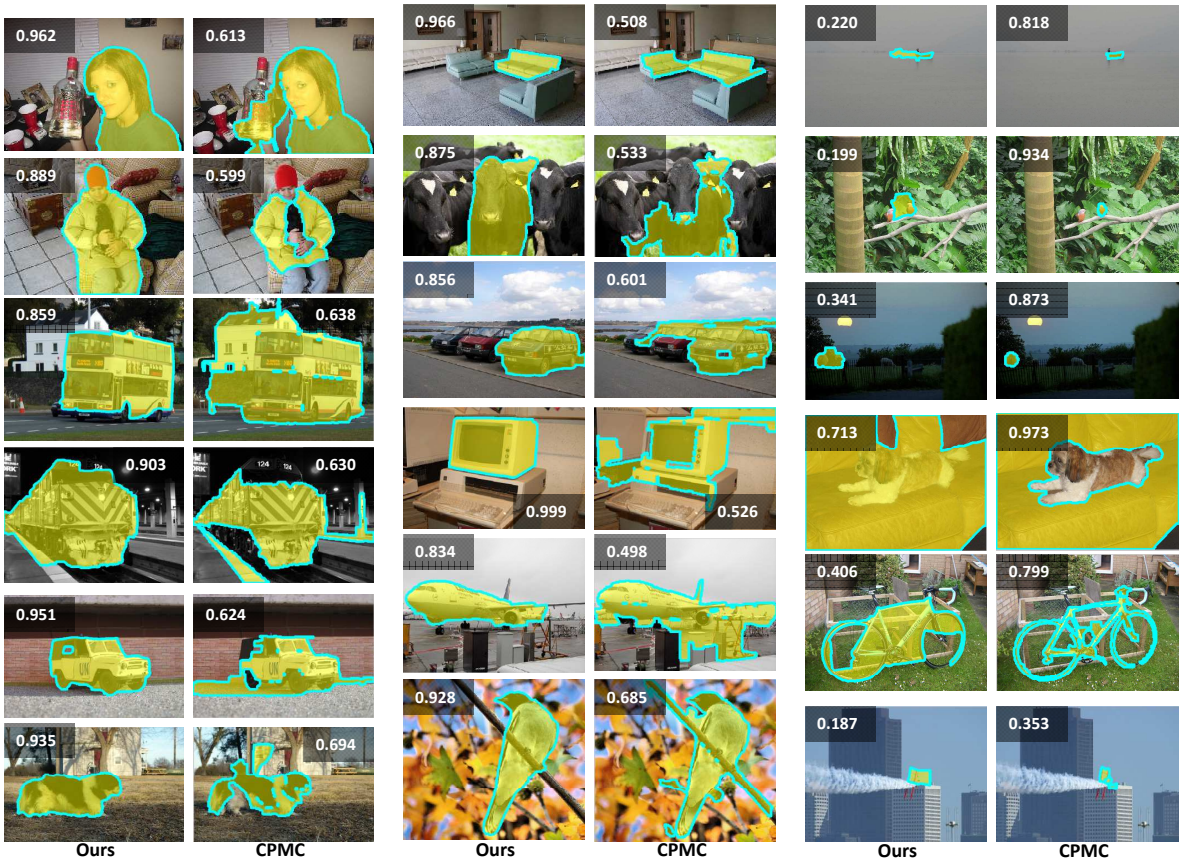
(a) CPMC [19], PASCAL (b) CPMC [19], BSD (c) Object proposals [24], PASCAL (d) Object proposals [24], BSD

Figure 4.9: Shape Sharing's gain in recall as a function of overlap. Ours outperforms the existing methods in both datasets. In particular, it provides the largest gains in the overlap scores of 0.6-0.9, in which a range of qualitative differences among segmentation methods is evidently perceived. Also, as in the color easiness test, the impact of the shapes becomes greater as the object size grows.

Figure 4.10: Example segmentations (yellow) alongside the ground truth (blue contours), for different degrees of overlap scores. Red box denotes the 0.6-0.9 overlap range, where usually perceptual quality differences are most evident. For overlaps beyond 0.9, many segmentations are so easy as to provide "equally good" results among methods. On the other hand, for low overlaps less than 0.5, segmentations are all poor, similarly making it hard to perceive the difference. However, in the range of about 0.6 to 0.9, segmentation quality is reasonable while images contain substantial challenges for segmentation, making the qualitative comparison among methods much more meaningful.

outperforms the baselines. In particular, our method provides the greatest gains in what is arguably a critical operating range for segmentation: overlaps from about 0.6-0.9. Why is this a critical range? For overlaps beyond 0.9, many segmentations are so easy as to make the perceived "winner" a toss-up. On the other hand, for low overlaps less than 0.5, segmentations are all poor, similarly making it hard to perceive the difference. However, in the range of about 0.6 to 0.9, segmentation quality is reasonable while images contain substantial challenges for segmentation, making the qualitative differences among methods much more evident. Figure 4.10 illustrates example segmentations over different overlap scores.

Figure 4.11 shows example results from our method and the best competing method, CPMC [19]. As shown in the examples, our shape priors can merge object parts of diverse appearance or separate objects of similar appearance that would not be possible if judging color/texture/contour alone.



(a) Objects with diverse colors

(b) Objects similarly colored as surroundings

(c) Failure cases

Figure 4.11: (a-c): results from Shape Sharing (left) and CPMC [19] (right). These contrasts illustrate when the shape prior is most beneficial, since CPMC uses color only. (a) Shapes pull together diversely-colored parts of an object. (b) Shapes help delineate an object from surroundings of similar colors: e.g., nearby objects from the same class (rows 1, 2, 3), or confusing backgrounds (rows 4, 5, 6). (c) Shapes do not help segment tiny objects (rows 1, 2, 3), nor objects lacking shape, e.g., the truncated sofa (4th row), or thin structured objects like the bicycle and airplane (rows 5, 6).

Approach	Covering (%)
Category-specific	**84.7**
Category-independent (Default)	84.3
Strictly category-independent	83.9
CPMC [19]	81.6
Object proposals [24]	81.7

Table 4.5: Comparison of category-independent and category-specific variants of our approach on PASCAL data. Our category-independent shape prior performs as well as a parallel category-dependent one.

4.3.3 Category-Independent vs. Category-Specific Priors

Finally, I directly study the extent to which our method's success is based on its category-independence. We compare Shape Sharing to two baselines. The first is a *category-specific* approach that operates just as our method, *except* that only exemplars of the same class as the test instance may be used (which, of course, uses information that would not be available in most realistic scenarios). The second is a *strictly category-independent* variant, where we require that the exemplar matched to a test image *must* be from another class; this too is not enforceable in realistic settings, but it verifies our gains are *not* due to having segmented exemplars of the same object class available.

Table 4.5 shows the results, with the previous baseline numbers repeated for reference in the bottom two rows. As expected, the category-specific variant performs best, and strictly-independent performs worst. However, the accuracy of all three is quite close. In addition, even our strictly independent variant outperforms the previous baselines that lack shape priors. This result demonstrates that shapes are truly shared among different categories, and one can use the proposed shape priors in a category-independent manner; hand-crafted exemplars for the object(s) in the test image are not needed in order to see gains from Shape Sharing.

4.4 Discussion

In this chapter, I introduced a category-independent shape prior for image segmentation. The main insight of my approach is that shapes are often shared between objects of different categories. To exploit this "shape sharing" phenomenon, I develop a non-parametric prior that transfers object shapes from an exemplar database to a test image based on local shape matching via BPLRs. Though not discussed in this chapter, I have also recently explored the power of shape priors initiated by local BPLR matches for video object segmentation [55]. Those results show the impact of the proposed shape priors for object segmentation in video.

Through extensive evaluations, I showed 1) shape sharing improves the quality of bottom-up segmentation, while requiring no prior knowledge of the object, and 2) the proposed category-independent prior performs as well as a parallel category-specific one, demonstrating that shapes are truly shared across categories. As such, unlike previous top-down segmentation methods, my approach can enhance the segmentation of previously unseen objects.

In this work, I used shape priors to generate multiple segmentations that may overlap each other. Those multiple object hypotheses provide a selected subset of reliable regions that higher-level tasks such as object recognition should focus on. By constraining the attention to the reliable candidates of object regions, this not only saves computation time for higher-level tasks but also enhances the robustness to noise. However, multiple overlapping segments cannot provide a coherent representation of the image, causing redundant hypotheses that may conflict with each other.

In this sense, it would be interesting future direction to find a single segmentation that provides non-conflicting regions of objects in the image. To find such a single segmentation, one could still leverage shape sharing to exploit category-independent shape priors. Unlike the multiple segmentation method that uses shape priors independently, however, one could explore a new method that considers all the priors at once to resolve the conflicts among overlapping ones. Priors that are more consistent with the bottom-up image evidence as well as less con-

flicting with other priors will be preferable to label pixels for segmentation.

Not only is it important to build a solid technique to find the solution, it is also crucial to evaluate the solution with a reasonable metric. Depending on which evaluation metric is used, the quality of the methods can be rated very differently. For segmentation, pixelwise metrics—such as pixel overlap with ground truth segment, or pixel distances to the ground truth contours—are commonly used to evaluate segmentation quality. Although they provide an objective measure for evaluation, it is not clear if those metrics are entirely consistent with human perception of quality. For example, it is acknowledged that the pixel overlap metric tends to assign higher scores to larger segments than smaller ones for a given object, preferring "blown-up" segments. In addition, the current metrics do not reflect the importance of the objects in an image; yet errors in the salient objects will be more critical to human observers. In this light, another interesting future direction would investigate how to evaluate the segmentation quality such that it is more consistent with human perception, especially focusing on segments' shapes and importance.

Thus far, I have addressed region detection—local shape detection (Chapter 3) and object segmentation (this chapter). In the next chapter, I will bring together the above ideas for local- and object-level region detection in a novel approach to efficient image matching, for the ultimate goal of region-based object recognition.

Chapter 5

Segmentation-Driven Matching for Object Recognition

Having established local- and object-level region detection in the previous chapters, I bring together those detected regions into a novel image matching strategy for object recognition. To this end, in this chapter I introduce a segmentation-driven local feature matching approach that uses segmented regions to guide local feature matching between images for exemplar-based object category recognition, which is published in [46].[1] The spatial layout among local features is represented by a 1D string that allows an efficient dynamic programming solution of matching features.

5.1 Motivation: Segmentation-Driven Local Feature Matching

Finding corresponding local features between images is a long-standing research problem in computer vision, and it is especially important to today's object recognition and image retrieval methods that use local feature representations. Thus far I have considered the matching of individual local features using a simple nearest neighbor search (e.g., feature repeatability and localization tests in Sec. 3.3.1 and 3.3.2, and naive Bayes object classification in Sec. 3.3.5).

However, the locality of such appearance-based feature matches yields some noisy correspondences when used alone, and so additional spatial layout among features is typically considered to select the geometrically consistent matching points among the initial pool of (confusing) appearance-based matches. Parameterized geometric constraints (e.g., an affine transformation between local regions) can be used for more reliable object instance matching

[1] Code and data are available online: http://vision.cs.utexas.edu/projects/asymmetricmatch

and image retrieval [64, 32, 44, 76]. For *generic* categories, however, geometric consistency is less exact and the correct transformations are non-global, making the parametric constraints less amenable to category-level object matching (e.g., matching images of articulated giraffes, or different models of boats). Instead, non-parametric approaches that identify a group of matches having minimal geometric distortion may be preferable in order to establish correspondences at the category-level [57, 9]. In addition to measuring overall image similarity, the resulting correspondences are useful to localize the object within the two views.

However, there are two key limitations to current techniques. First, pre-computing the distortion for all tentative matching pairs is computationally expensive, making very densely sampled feature points off-limits in practice. As a result, most methods restrict to a sparsely sampled set of features (such as local maxima in scale-space, edge points, etc.). While generally effective for matching object instances, sparsely sampled interest points provide a weaker representation for category-level matching; much evidence suggests that a dense coverage of features is preferable. For example, observations in [72] show dense features outperform sparse ones for object classification tasks. Similarly, in Chapter 3 I showed the density of BPLRs contributes to better repeatability of features.

A second limitation is that non-parametric methods typically identify a single group of corresponding points that undergo a common (low-distortion) transformation. Yet in typical real images, each part of a non-rigid object—or each instance of multiple objects in the image—can undergo a different transformation, suggesting that we should identify multiple groups of corresponding points, each with a different geometric configuration.

I propose a dense feature matching algorithm that exploits the grouping provided by bottom-up segmentation to compare generic objects with non-parametric geometric constraints. Our method takes two images as input and returns a score for their similarity. One input is left unsegmented, while the other is automatically segmented. The matching process begins by finding correspondences between points within each region of the segmented image and some

subset of those within the unsegmented image. In each of the region-to-image match groups, the spatial layout among local features within each region is represented by a 1D string that links nearby features, which allows us to formulate an objective solvable with dynamic programming. The union of these correspondence groups are then further evaluated for their mutual geometric consistency, at which point we favor low distortion matches *within* each segmented region, while allowing larger deformations *between* the segmented regions of the original image.

The key technical aspects that address the current limitations are 1) 1D string representation and 2) per-region matching scheme. Our 1D representation enables an efficient dynamic programming solution, allowing denser correspondences. Second, our per-region matching scheme allows different regions to move in different ways. This provides greater flexibility when matching deformable objects.

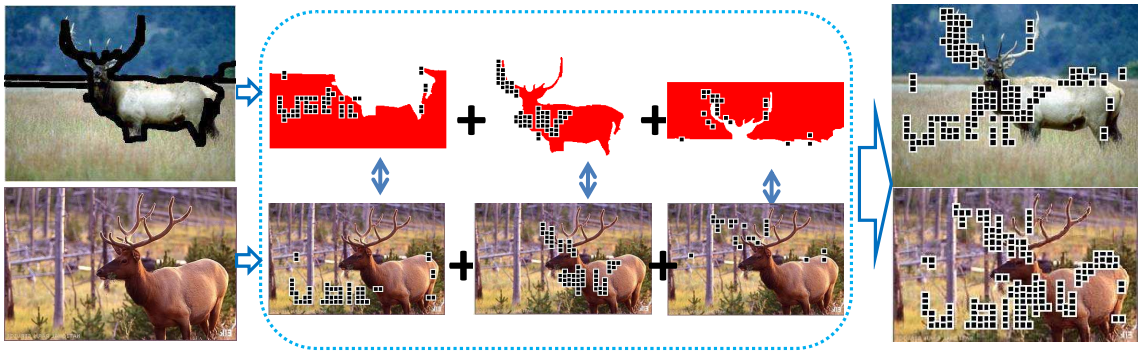
I call the proposed image matching "asymmetric" because only one of the input images is segmented into regions, and its groups of points can match within any (consistent) portion of the other image. We find this deliberate imbalance to be an advantage when matching: we get the grouping power of low-level segmentation to assemble candidate regions of points, but without suffering in the inevitable event where the bottom-up cues produce incompatible regions for two images of the same object (see Figure 5.1).

5.2 Asymmetric Region-to-Image Matching

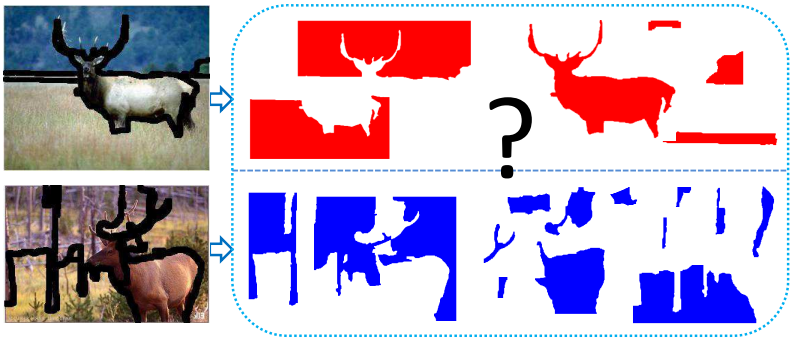
In this section, I present my approach for matching local features with segmented regions. Our method takes two images as input and returns a score for their similarity, as well as correspondences explicitly indicating their matching features.

5.2.1 Region-to-Image Point Matching

We first decompose one of the two input images into regions using bottom-up segmentation. Each region is mapped to a set of local SIFT descriptors [64] densely sampled at multiple



(a) Asymmetric region-to-image point matching (Our approach)



(b) Region-to-region matching

Figure 5.1: **(a)** The proposed asymmetric region-to-image matching method, and **(b)** the key contrast with region-to-region matching. In a *region-to-image* match, we use regions from the segmentation of one image (top row, left) to group points for which to seek matches in the second unsegmented image (bottom row, left). In our asymmetric strategy, we exploit the fact that a group of feature points (small squares, e.g., denoting dense SIFT) within the same segment often belong to the same object subpart, giving us an automatic way to generate groups which when matched to the second image, should have low total geometric distortion. For example, here, the elk horns, body, and grass are well-matched (center, larger images) even though the parts separately undergo different deformations to fit to a different category instance in the second image. In contrast, a *region-to-region* match that seeks correspondences between pairs of regions from *both* images' segmentations (two rows on right side), cannot exploit the bottom-up grouping as well, since it can be misled whenever the bottom-up segmentations for the two images lack agreement.

scales on a regular grid.[2] We denote each grid location as a "point". Then, we represent each region by strings of its constituent points. To more robustly represent the 2D layout using a 1D string, we extract two strings per region: a column-wise string and row-wise string. In the column-wise linkage of a string, we start from the top-left point in the region, and link nearest points along a column. When we arrive at the end of a column, the end-point is linked to the closest point in the next column. We repeat this process until we reach the end-point of the last column. Similarly, row-wise linkage links nearest points along a row.

Our string representation mechanically links neighbor points in a grid, and thus it is fast to build. In addition, it is robust to image variations as it is constructed from regularly sampled dense features whose locations are not affected by appearance changes. However, a more sophisticated representation such as tree that reflects the layout among features can be also considered.

When matching a region to the unsegmented image, for each point in the string, we first find a set of candidate matching points in the unsegmented image, as determined by the SIFT descriptor distances across multiple scales. Note that we only extract strings from one of the input images; the candidate matches may come from anywhere in the second (unsegmented) image. Given these candidate matches, we compute the optimal correspondence by using dynamic programming (DP) to minimize a cost function that accounts for both appearance and geometric consistency (to be defined below). We obtain the solution for both the row-wise linked string and the column-wise linked string, and take the union of the correspondence pairs to be the region-to-image matches for that particular region. See Figure 5.2 for an overview.

In the following, I define the cost function, and then explain how the resulting correspondences are scored to produce a single similarity value between the two images.

[2] At the time of this work, we did not have the BPLR detector, and so we chose the popular SIFT feature to test the matching method.

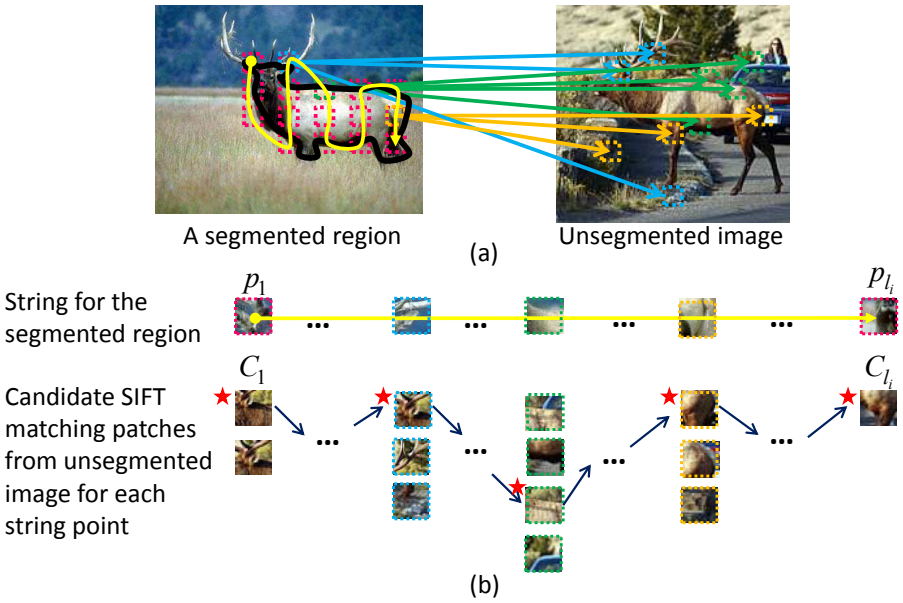


Figure 5.2: Illustration of an asymmetric string match. **(a)** A segmented region in the left image is represented by a column-wise string. For each point in the string, we identify candidate matches in the (unsegmented) right image by SIFT matching. **(b)** We use dynamic programming (DP) to solve for the assignment between the string and the candidates that minimizes the total geometry and appearance costs. Short arrows and stars denote the optimal matches selected with DP. (Note, this example does not include any null matches.) Best viewed in color.

5.2.2 Match assignment cost function

The segmented image produces a string for every region. Each region's string consists of a set of points, $P_i = \{p_1 \ldots, p_{l_i}\}$, where l_i denotes the length of the i-th region's string, each p_k records the image coordinates for that point, and any (p_k, p_{k+1}) denotes a pair of neighboring points on the string. We normalize the image coordinates by the length of the longer side of the image, making the range of coordinate values between 0 and 1. Let C_k denote the set of candidate matching points for a point p_k; each point in C_k is a feature in the unsegmented image whose SIFT descriptor is close to the one associated with p_k (e.g., C_1 is the first column of patches in Figure 5.2(b)). Among the candidate sets $[C_1, \ldots, C_{l_i}]$, we want to solve for the optimal matching $M^* = \{m_1, \ldots, m_{l_i}\}$, where each $m_k \in C_k$, such that the assignment minimizes the following cost function:

$$\mathcal{C}(P,M) = \sum_{k=1}^{l_i-1} w_g G(p_k, p_{k+1}, m_k, m_{k+1}) + \sum_{k=1}^{l_i} w_a A(p_k, m_k) + \sum_{k=1}^{l_i-1} w_o O(p_k, p_{k+1}, m_k, m_{k+1}) + \sum_{k=1}^{l_i} w_d D(p_k, m_k). \quad (5.1)$$

The cost function has two pairwise terms, $G(\cdot)$ and $O(\cdot)$, and two unary terms, $A(\cdot)$ and $D(\cdot)$. Each term has an associated weight (w_g, w_o, w_a, and w_d) that scales their relative impact. The input P is fixed; we optimize over the selections in M. We now define each component term.

The *geometric distortion* term,

$$G(p_k, p_{k+1}, m_k, m_{k+1}) = \|(p_k - p_{k+1}) - (m_k - m_{k+1})\|_2,$$

measures the pairwise geometric deformation between pairs of corresponding points. This gives preference to neighboring match pairs that have similar distortion.

The *ordering constraint* term, $O(\cdot)$, penalizes the pairs of correspondences when they violate the geometric ordering. Its value is 1 if the ordering of p_k and p_{k+1} is different from that of m_k and m_{k+1} (in either the horizontal or vertical direction), and 0 otherwise. This means, for example, that if point p_k is located left of the point p_{k+1}, its matching point m_k should be located left of m_{k+1}.

The *appearance similarity* term penalizes dissimilarity between the SIFT descriptors extracted at the two points, and is defined as:

$$A(p_k, m_k) = \frac{1}{1 + \exp\left(-\tau_a \left(\frac{1}{\mu_a}\|f_{p_k} - f_{m_k}\|_2 - 1\right)\right)}, \quad (5.2)$$

where f_{p_k} and f_{m_k} denote the SIFT descriptors at p_k and m_k, respectively, and $\|f_{p_k} - f_{m_k}\|_2$ is determined by the minimum distance among the descriptors at all scales extracted at the two points. The positive constants μ_a and τ_a simply adjust the shape of sigmoid function to make the values compatible with the other cost terms. Larger SIFT distances induce larger cost values.

Finally, the *displacement constraint* term $D(p_k, m_k)$ penalizes large displacement between the locations of corresponding points, thereby giving some preference for objects that occur within similar scene layouts:

$$D(p_k, m_k) = \begin{cases} \|p_k - m_k\|_2, & \text{if } \|p_k - m_k\|_2 > t \\ 0, & \text{otherwise}, \end{cases} \tag{5.3}$$

where t is a displacement threshold.

Our description thus far assumes that every point in the string has a corresponding point in the second image. However, some points may not have a reliable matching point. To handle this, we insert a "null match" candidate into each candidate set C_k. If a null match is selected when optimizing $\mathcal{C}(P, M)$, it means that the associated point does not have a real corresponding point. For every term where either m_k or m_{k+1} is a null match, the cost value in each sum of Eqn. 6.1 is set to a constant value χ. We set the constant χ to be larger than the typical matching costs for reliable correspondences.

For each region P_i in the segmented image, we use dynamic programming to minimize Eqn. 6.1 over the candidate selections for M, producing a set of corresponding points for each region. The union of those correspondences across all of the segmented image's regions is then the final set of matches between the two images. Let $\{(p_1, m_1), \ldots, (p_n, m_n)\}$ denote this final set of n total corresponding points.

5.2.3 Scoring the resulting correspondences

Given these correspondences, we want to assign a single match score between the two original images. Note that while each individual region's match is scored by $\mathcal{C}(P, M)$, the final match score is based on the union of the regions' matches, and must both summarize their appearance similarity as well as incorporate the geometric deformations *between* the component region matches. For two images I_1 and I_2, we define this score as a summation of match scores

between all their corresponding points:

$$S(I_1, I_2) = \frac{1}{\sqrt{N_1 N_2}} \sum_{i=1}^{n} \omega_i \; s_a(p_i, m_i) \; s_g(p_i, m_i), \tag{5.4}$$

where N_1 and N_2 are the total number of points in each image, respectively, ω_i is a weight assessing the importance of the i-th matching pair (and will be defined below), and the s_a and s_g functions score each point match's appearance and geometric similarity, respectively. High values of $S(I_1, I_2)$ mean the two images are very similar.

The appearance similarity score is determined by the SIFT distance: $s_a(p_i, m_i) = 1 - A(p_i, m_i)$, where we are simply mapping the cost from Eqn. 6.2 to a similarity value.

The inter-point geometric consistency score $s_g(p_i, m_i)$ measures the average pairwise deformation between a matching pair (p_i, m_i) and all other matching pairs in the final set:

$$s_g(p_i, m_i) = \frac{1}{n-1} \sum_{k} \frac{1}{1 + \exp\left(\tau_g \left(\frac{G(p_i, p_k, m_i, m_k)}{\alpha_{ik}} - 1\right)\right)},$$

where $G(\cdot)$ is as defined above, and $k \in \{1, \ldots, n\} \backslash i$. This entire term gives lower similarity values to larger total distortions. The constant τ_g adjusts the shape of the sigmoid function, and α_{ik} weights the pairwise deformation differently according to whether the two points are in the same region. Specifically, if p_i and p_k are in the same region, $\alpha_{ik} = \alpha$; otherwise, $\alpha_{ik} = 2\alpha$, where α is a positive constant. This doubling of the penalty for within-region matches enforces stronger consistency for pairs within the same region, while allowing more distortions for the matching pairs across different regions.

Finally, the weight ω_i in Eqn. 5.4 emphasizes the similarity of those correspondence pairs for which the point p_i has fewer initial candidates in the unsegmented image; the intuition is that those matches will be more discriminating. For example, a point in a textureless region will have a large number of SIFT candidate matches in the second image, but many will be less distinctive. Thus, we set $\omega_i = (1 + \exp(\tau_\omega(\frac{|C_i|/N_2}{\mu_\omega} - 1))^{-1}$, where $|C_i|$ denotes the number of initial matching candidate points for point p_i found in the unsegmented image, and N_2 denotes

the total number of points in the second (unsegmented) image. The remaining constants simply serve to adjust the shape of the sigmoid, and provide a weight value $\omega_i \in [0.5, 1]$. Algorithm 3 summarizes the whole steps for our matching method in pseudo-codes.

Algorithm 3: Asymmetric string match

Data: One segmented image I_1 and another unsegmented image I_2
Result: A set of matched points (2D locations) between two images and a matching score

`/* Extract features (e.g., SIFTs) on a regular grid over different sizes of patches */`
input : I_1 and I_2
output: $\mathbf{F_1}$ and $\mathbf{F_2}$, each of which is a set of features extracted from I_1 and I_2.
for $i = 1$ **to** 2 **do**
  Divide a I_i into regular grid (e.g., 6×6 pixels);
  foreach *grid location* l_j *in* I_i **do**
    for $k = 1$ **to** s **do** `/* different patch sizes */`
      Extract a feature f_{jk} from a patch p_k at l_j;
      Add f_{jk} and its location l_j into $\mathbf{F_i}$;
    end
  end
end

`/* Finding matching candidates */`
input : $\mathbf{F_1}$ and $\mathbf{F_2}$
output: Matching candidates $\mathbf{C_{1\rightarrow 2}}$ found from $\mathbf{F_2}$ w.r.t. features in $\mathbf{F_1}$
foreach *grid location* l_i *in* I_1 **do**
  for $j = 1$ **to** s **do** `/* different patch sizes */`
    Get a feature f_{ij} from $\mathbf{F_1}$;
    Find K nearest neighbors of f_{ij} from $\mathbf{F_2}$;
    Add the K nearest neighbors and their matching distances into $\mathbf{C_i}$;
  end
  `/* Pruning by matching distance */`
  foreach *feature* f *in* $\mathbf{C_i}$ **do**
    if *f's matching distance* $\geq$ *threshold* **then**
      Remove f from $\mathbf{C_i}$;
    end
  end
  `/* Pruning by ranking */`
  Sort matching distances in $\mathbf{C_i}$ and keep top N nearest neighbors;
  Add $\mathbf{C_i}$ into $\mathbf{C_{1\rightarrow 2}}$;
end

Algorithm 3: Asymmetric string match: continued

```
/* Region-to-image matching                                    */
```
input : $\mathbf{C_{1\rightarrow 2}}$
output: Optimal matching $\mathbf{M_{1\rightarrow 2}}$ between features of two images
foreach *region* r_i *in the segmented image* I_1 **do**
```
    /* Column-wise string match                                */
```
    Build a column-wise string that links grid locations in r_i;
    Build a matching cost table as seen in Figure 5.2 along the string using $\mathbf{C_{1\rightarrow 2}}$;
    Obtain optimal matches $\mathbf{M_c}$ via dynamic programming for the objective defined in Eqn. 6.1;
```
    /* Row-wise string match                                   */
```
    Build a row-wise string;
    Do the same steps as above and obtain optimal matches $\mathbf{M_r}$;
```
    /* Union of the matches                                    */
```
    Compute the union of $\mathbf{M_c}$ and $\mathbf{M_r}$ and add the union into $\mathbf{M_{1\rightarrow 2}}$;
end
```
/* Compute a match score between two images                    */
```
input : $\mathbf{M_{1\rightarrow 2}}$
output: A match score S
Compute a match score S given $\mathbf{M_{1\rightarrow 2}}$ using Eqn. 5.4;

return $\mathbf{M_{1\rightarrow 2}}$ and S

5.3 Results

I apply our matching algorithm to exemplar-based object category recognition on the Caltech-256 and 101 datasets. Both are among the largest benchmarks available for object category recognition (see Figure 5.3 for example images in the datasets). The main goals of the experiments are 1) to demonstrate the impact of the proposed matching method for exemplar-based object recognition (Sec. 5.3.1) and 2) to analyze computation cost for practical use (Sec. 5.3.2).

Implementation details For both datasets, we resize the images such that their longer side is 320 pixels, and then densely sample SIFT descriptors at every eight pixels over four scales, generating about 1200 points per scale for the typical image size (320 x 240). The descriptors

Figure 5.3: Example images from 256 object categories in Caltech-256 dataset. Caltech-101 is a subset of Caltech-256.

are sampled in square patches of sizes 16, 24, 32, and 40 pixels. [3] We use the segmentation method of [4], and the authors' code. We use approximate nearest neighbor search to quickly find close SIFT descriptors, with code by [71].

We set the weight parameters in the cost function by visually inspecting the matches for ten same-class image pairs, mainly to understand the trade-off between the geometric and appearance terms. All were fixed after this initial inspection; we did not attempt to validate them with recognition accuracy. We lock the values for all experiments and all $\sim$15,000 images.

We use a simple near-neighbor classifier to perform exemplar-based category recognition with our matching. To avoid exhaustively matching against each of the exemplars, for each query, we first prune the pool of classes according to SIFT descriptor distances (we use the top

[3] The initial candidate matching points are those with a SIFT descriptor distance within $1.25\mu_a$ for each scale, among all scales.

Method	Accuracy (%)
SPM [37]	42.1 ± 0.81
GBDist [93]	45.2 ± 0.96
BergMatching [9]	48.0
GBVote [8]	52.0
Ours	**61.3**

Table 5.1: Comparison of our method to other image matching algorithms on the Caltech-101, for 15 training images per category. All methods listed here use nearest-neighbor classification based on an image matching score. Ours outperform all of the most relevant nearest neighbor matching methods by 10-20% gains, demonstrating the impact of the proposed matching strategy.

25 and 30 classes for the 256 and 101, respectively). Then, we apply our matching method to search for the top matched exemplars in only those classes. To categorize a query image, we take the sum of our method's matching scores for the k top-scoring exemplars in each category; the top-scoring category is the predicted label. All of the results use $k = 2$, which produced the best performance (for $k = 1, 2, 3$, accuracy varies only by 0.5-1%).

5.3.1 Object Category Recognition

In this section, I apply our matching method to exemplar-based object recognition, and compare its results to existing techniques.

Caltech-101 Dataset We randomly pick 15 exemplar and test images per category. Table 5.1 compares our algorithm to other image matching algorithms. Here we focus on those methods that are most closely related: all results listed use a nearest-neighbor scheme based on image-to-image matching scores. Our method clearly outperforms the existing methods by a large margin. In particular, it gives far stronger recognition accuracy than the method of [9], though both algorithms include related pairwise geometric constraints. This suggests that our asymmetric region-to-image matching is more effective for imposing geometric constraints than an image-to-image matching approach when dealing with intra-class variations, and also supports using more densely sampled descriptors (which is more efficiently done in our method).

Feature	Method	Accuracy (%)
Single	PMK+SVM [36]	50.0
	SVM+kNN [99]	59.1 ± 0.6
	SPM+SVM [37]	59.3
	GBDist+SVM [93]	59.3 ± 1.0
	NBNN (1 desc.) [11]	65.0 ± 1.14
	Ours	61.3
Multiple	SKM [51]	57.3
	KTA [60]	59.8
	LearnDist [33]	63.2
	MKL [35]	70.0
	BoschTree [13]	70.4 ± 0.7
	NBNN (5 desc.) [11]	72.8 ± 0.39

Table 5.2: Comparison of our method to best existing recognition algorithms on the Caltech-101, for 15 exemplar images per class. The table divides the methods into two groups, depending on whether they use a **single** descriptor type or combine **multiple**. Our result competes strong learning-based methods, demonstrating our raw matching accuracy is quite strong.

Table 5.2 compares existing state-of-the-art algorithms, divided into two groups based on whether single (top) or multiple (bottom) descriptor types are used. When comparing to methods using a single local feature type as we do, our method is better than all previous methods, except for the method of [11], which measures an image-to-class distance without explicitly computing individual image-to-image distances. In contrast, our method computes both the similarity between images as well as the matching feature assignment. This can be seen as an advantage, since the localization functionality is potentially useful for both detection in new images as well as for feature selection among training exemplars. Interestingly, the authors report that their accuracy decreases by 17% when they attempt an explicit image-to-image matching using the same protocol as the one used in their image-to-class matching [11].

When compared to the methods using multiple types of local features (Table 5.2, bottom), our accuracy using only a single feature is a bit behind the state-of-the-art, which is perhaps expected given the known value of complementary feature types. At the same time, however, we do outperform some multi-feature learning methods [51, 60], and have comparable numbers to the method of [33], which includes a sophisticated learning stage to identify discriminative

Method	Accuracy (%)
Todorovic-GenLearn [88]	54.0
Todorovic-DiscLearn [89]	72.0
Gu et al. [38]	65.0
Ours	61.3

Table 5.3: Comparison of our method to region-to-region matching methods on the Caltech-101, for 15 training images per class. Ours outperforms region-to-region matching method [88], showing the effectiveness of our asymmetric region-to-image matching strategy. Compared to learning-based methods [89, 38], our method is behind by 4-10 points, but we should note that our result comes from raw matching scores; in fact, the best result [89] is obtained by adding a learning component to the raw matching result of [88].

features. Thus, overall, these results demonstrate that our raw matching accuracy is quite strong.

Table 5.3 compares our method to those based on region-to-region matching. Numbers reported for other algorithms in Table 5.3 do not give their "raw" region-to-region matching accuracy, since all incorporate learning algorithms on top of the matching. Without learning, our algorithm outperforms that of [88], suggesting that the image-to-region match can indeed be more robust when matching with imperfect bottom-up segmentations. We obtain close accuracy to [38], though in our case without any discriminative region selection. Compared to [89], our algorithm is about 10 points less accurate. One interesting thing here is that both [88] and [89] rely on the same region-to-region matching algorithm, but the more recent [89] improves over [88] by about 20 points, apparently due to the switch from generative to discriminative learning. Since the matching algorithm is the same in both cases, this suggests that the significant jump may be largely attributable to the powerful learning algorithm, not the matching itself.

Caltech-256 Dataset For the Caltech-256, we randomly pick 30 exemplar images and 25 test images per category. Table 5.4 compares our algorithm to existing methods. To our knowledge, we are the first to attempt to use raw image matching to perform recognition on this challenging dataset.

Method	Accuracy (%)
Todorovic-GenLearn [88]	31.5
SPM+SVM [37]	34.1
NBNN (1 desc.) [11]	37.0
NBNN (5 desc.) [11]	42
BoschTree (No ROI) [13]	38.7 ± 1.3
BoschTree (ROI) [13]	43.5 ± 1.1
MKL [35]	45.8
Torodivic-DiscLearn [89]	49.5
Ours	36.3

Table 5.4: Comparison of our method to existing results on the Caltech-256, for 30 training images per category. Ours provides comparable results to the state-of-the-art learning-based methods. We should note that ours is the first image matching method tested on Caltech 256 that achieves a raw matching accuracy comparable to some classifier-based algorithms.

Compared to methods using a single type of local feature, our nearest-neighbor accuracy improves over the SVM-based method used in [37], and gives very similar results to that of [11]. Compared to region-to-region matching-based methods, we achieve better accuracy than the method of [88], which learns generative models for each category based on the matching. Compared to methods using multiple feature types [13, 35], our method lags by only about 2.4-9.5%. Given that our recognition accuracy is based solely on raw image matching with a single feature type, these are very encouraging results.

Qualitative Results For qualitative evaluation I show the nearest neighbors for some image queries (see Figure 5.4 and 5.5). We selected query images from some of the hard categories for which previous recognition methods produce below average accuracy [37]. Our algorithm shows powerful matching results for those difficult object categories under notable intra-class variations or background clutter. Further, even for the failure cases (as judged by the true category label of the nearest exemplar), we find that the top matched images often show very similar geometric configurations and appearance, meaning the mistakes made are somewhat intuitive (see last two rows in Figure 5.4 and the last row in Figure 5.5).

In addition, I demonstrate example matches that reveal some interesting aspects of my

Figure 5.4: Examples of matching results on the Caltech-256. Leftmost image in each row is a query, following six are nearest exemplars found by our method among 7680 total images. Last two rows show queries whose nearest exemplar has the wrong label, and yet seem to be fairly intuitive errors.

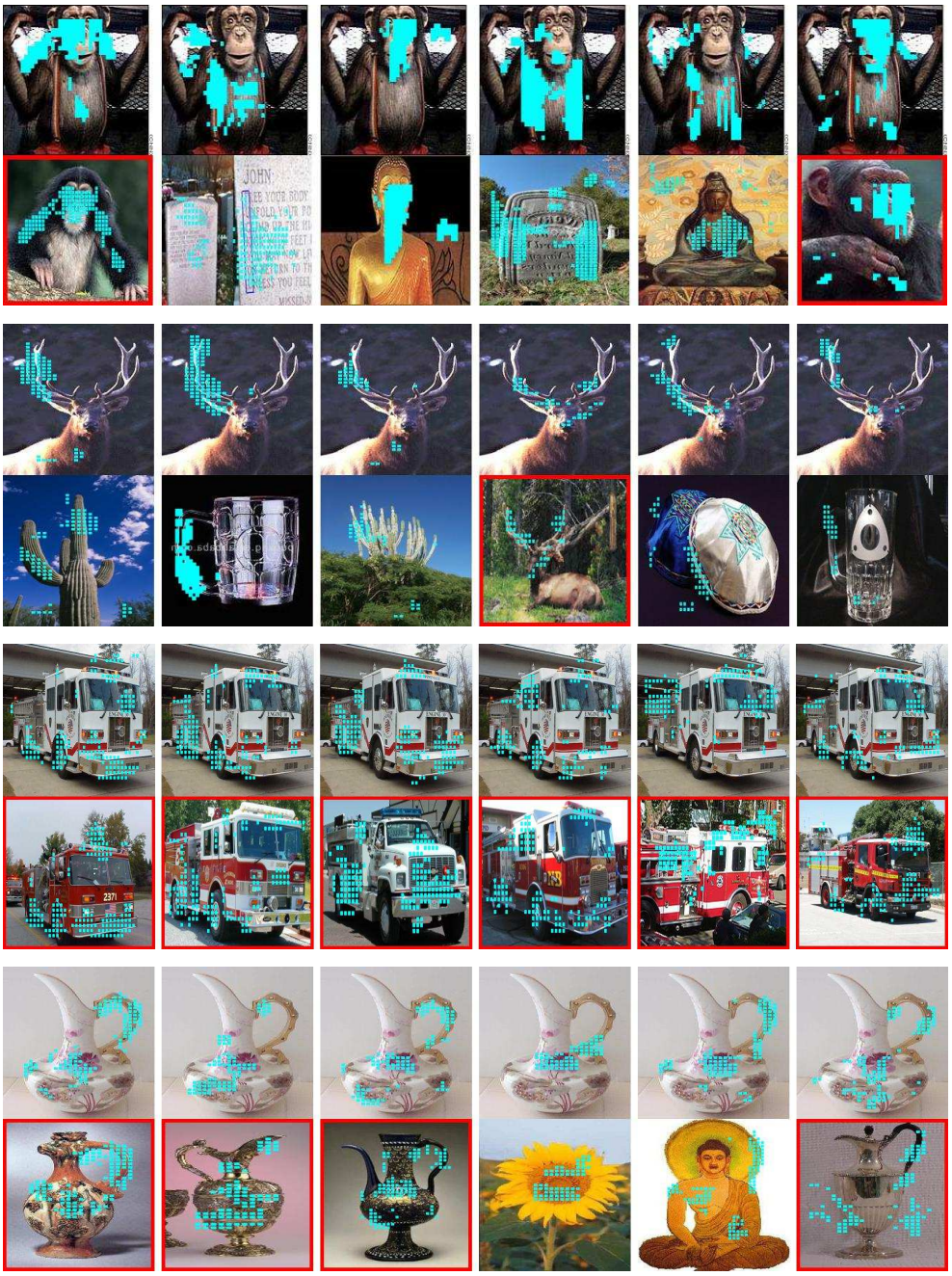
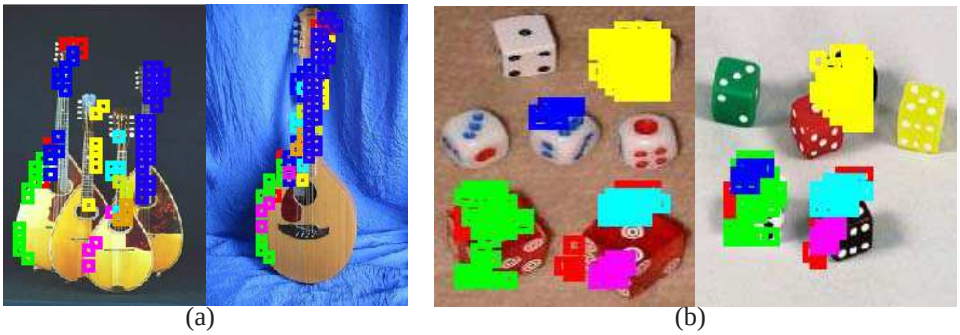


Figure 5.5: Examples of matching results with corresponding points visualized. For each query image (repeated six times in the top row of each group), we show its six nearest neighbor exemplars (the six images in the bottom row of each group), along with corresponding points between them. The small cyan squares denote the subset of densely sampled SIFT points that formed the final correspondence between the query and matched image, according to our algorithm. Note that different points are marked on each instance of the query, since different points contribute to each of the six individual matches. An exemplar image surrounded by a red square denotes that it belongs to the same category of the query. Last example show a query whose nearest exemplar has the wrong label, and yet seems to be intuitive.



(a) (b)

Figure 5.6: Example many-to-one and many-to-many matches. In both examples, the left image is segmented, and the right remains unsegmented. Points are color-coded according to their correspondence. **(a)** Different parts of an object (Mandolin's neck and base) are matched to a single instance of the object in the second image. **(b)** Multiple instances of the same category (dice) are matched.

method. Figure 5.6 shows examples of many-to-one or many-to-many object matching with my method, which illustrates how my asymmetric approach can successfully match images that lack a strict global geometric consistency. This flexibility is useful for matching object categories whose instances have noticeable geometric variation.

A possible concern might be that because of the asymmetry, our matching results may vary depending on which image's segmentation is used for matching. I find in practice it typically produces similar final matching scores, which makes sense given that we only use a segment as a bounding area for imposing geometric constraints, not as a matching primitive. Figure 5.7 shows two examples illustrating this point. The matching points are not identical when we swap which image is segmented, yet we see that the quality of the assignments is quite similar.

5.3.2 Computational Cost

Using the dense sampling described above, we get about 4800 features per image. It takes about 5-15 seconds to match two such images using our MATLAB implementation on a 2.5GHz CPU. Computation time varies depending on how many initial matching candidates are obtained via SIFT search. Segmentation using [4] required about 3-5 minutes per image,

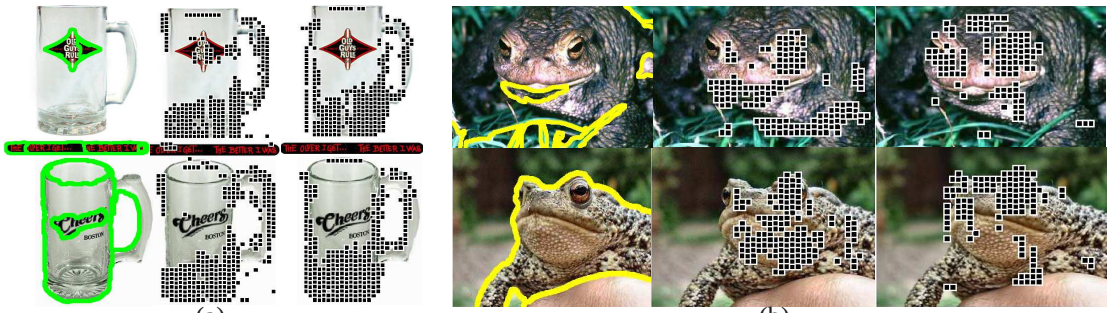


Figure 5.7: Examples showing robustness to the choice of which image is segmented. First columns in (a) and (b) show the segmentation of each image, and the remaining two columns show matched points (black dots) when we swap the segmentation used.

though the authors report that much faster parallel implementations are possible ($\sim$2 *s*). When using our matching to compare to exemplars for recognition, note that the segmentation cost is a one-time thing; due to the asymmetry, we only need to have segmented exemplars, and can leave all novel queries unsegmented.

To compare the cost of our method to other matching algorithms, we tested the computation time of the spectral matching algorithm [57], which is known as the most efficient matching algorithm among ones using a pairwise geometric constraint. In our MATLAB implementation, it takes more than 10 minutes to match images of 320 by 240 pixels with dense features. Most of that time was consumed by the pairwise distortion computation; once those are computed, the spectral matching itself runs fast. Another popular and effective matching algorithm ([9]) has been reported in [57] to be limited in practice to running with under 50-100 features, due to the expense of the linear programming step. Thus, the existing most related matching algorithms that enforce geometric constraints do not seem amenable for dense point matching.

5.4 Discussion

The proposed matching method in this chapter showed that a dense feature matching provides much better discriminating power for recognizing a large number of different object categories. Also, it is critical to consider spatial layout among features for reliable matching. Our 1D string representation and segmentation-driven matching cost are combined to provide an efficient solution for matching features with geometric constraints enforced.

I would like to stress two important aspects of my approach. First, rather than consider only region-to-region matches between the two images' segmentations, we take the feature grouping given by each region in one image to find a geometrically consistent match in *any part* of the second image. This way, we can find the best matches whether or not the region in the second image would have happened to appear in a segmentation (see Figure 5.1).

Second, this very idea is what lets us efficiently solve for matching regions using dynamic programming (DP), without being susceptible to traditional string matching's sensitivity. Our matches are not string-to-string. Rather, we match a string from one region to a set of candidate points identified by local appearance similarity (see Figure 5.2). This means we avoid the pitfalls of traditional DP-based matching, namely, sensitivity to the strings' start and end points, and overly strict geometric consistency requirements. The upshot, as I demonstrated in the experiments of the previous section, is that ours is the first image matching method to realize a dense matching with non-parametric geometric constraints—both aspects vital to matching images with generic object categories.

Compared to existing approaches that compute pairwise relations among all points, my approach substantially reduces the computational complexity of enforcing pairwise geometric constraints on the candidate match pairs. This complexity advantage does come at the price of considering geometric relations only between adjacent points on the string when solving for the optimal match. Nonetheless, we purposely mitigate the impact of this simplification by both 1) using two strings per region (column- and row-wise linked), as well as 2) using densely sampled

local features and cross-scale matches.

One promising extension would be to add scale and/or rotation invariance on the method. My current implementation is not fully scale or rotation invariant. While we do employ multiple scales of SIFT descriptors for matching, we sample feature points on a single grid. One could easily achieve scale invariance by matching across an image pyramid, or adding a multi-scale grid. We retain orientation information (which can be useful for object matching) since the geometric distortion term is computed with respect to 2D coordinates. When using our measure for example-based recognition, we assume a class's pose and scale variation will be represented via the exemplars. In Chapter 6, I will explore a multi-scale matching that exploits regions' hierarchical structure.

In this chapter, I leverage the spatial grouping cue suggested by each individual segment to enforce geometric consistency on the matched local features. Such a geometric constraint is treated independently for each region in the matching objective, which allows greater flexibility when matching deformable regions. In next chapter, I consider another spatial cue derived from regions, "region hierarchy". In contrast to the grouping cue suggested by each individual region, region hierarchy comes when we consider regions as a whole. This region hierarchy links different regions to be matched simultaneously. To exploit the region hierarchy, I introduce a novel graphical model, particularly focusing on fast image matching.

Chapter 6

Fast Image Matching with a Region Hierarchy

In the previous chapter, I introduced a segmentation-driven local feature matching, where each segmented region provides a geometric constraint that encourages a group of local features within it to find their matches with stricter spatial consistency. In this chapter, I consider another strong geometric cue derived from regions, "region hierarchy", to improve the matching. Whereas the grouping cue used in the previous chapter is suggested by each *individual* region, a hierarchical structure comes when considering these regions as a *whole*: all the regions cover various spatial extents from an entire image to its objects to their parts. Particularly, I focus on fast image matching that exploits such a region hierarchy. To this end, I introduce a deformable spatial pyramid (DSP) matching for fast dense pixel correspondences, which is published in [49].[1] The resulting DSP method can match hundreds of thousands of pixels in a faction of a second and is applied for exemplar-based semantic image segmentation.

6.1 Motivation: Region Hierarchies For Fast Pixel Matching

Thus far, I addressed matching problem that considers up to $\sim$5000 (sampled) local features to be matched. As already seen, $\sim$5000 features are dense enough for object classification task, where the segmentation-driven matching approach efficiently recognizes hundreds of object categories with challenging intra-class variations (Chapter 5). More recently, however, researchers have pushed the boundaries of dense matching to estimate correspondences of *millions of pixels* between images of different scenes or objects. This leads to many interesting new applications, such as semantic image segmentation [62], image completion [6], and video

[1] Code and data are available online: http://vision.cs.utexas.edu/projects/dsp

depth estimation [45].

There are two major challenges when matching pixels between generic images: computational cost and image variation. Different scenes and objects undergo severe variations in appearance, shape, and background clutter. These variations can easily confuse low-level pixel matching functions. At the same time, the search space is much larger, since generic image matching permits no clean geometric constraints. Without any prior knowledge on the images' spatial layout, in principle we must search every pixel to find the correct match.

To address these challenges, as discussed in Chapter 2, existing methods have largely focused on imposing geometric regularization on the matching problem. Typically, this entails a smoothness constraint preferring that nearby pixels in one image get matched to nearby locations in the second image; such constraints help resolve ambiguities that are common if matching with pixel appearance alone. If enforced in a naive way, however, they become overly costly to compute. Thus, researchers have explored various computationally efficient solutions, including hierarchical optimization [62], randomized search [6], 1D approximations of 2D layout [46], spectral relaxations [57], and approximate graph matching [23].

Despite the variety in the details of prior dense matching methods, their underlying models are surprisingly similar: minimize the appearance matching cost of individual pixels while imposing geometric smoothness between paired pixels. That is, existing matching objectives center around *pixels*. While sufficient for instances (e.g., MRF stereo matching [83]), the locality of pixels is problematic for generic image matching; pixels simply lack the discriminating power to resolve matching ambiguity in the face of visual variations. Further, the computational cost for dense pixels still remains a bottleneck for scalability.

To address these limitations, I introduce a *deformable spatial pyramid* (DSP) model for fast dense matching. Rather than reason with pixels alone, the proposed model exploit regions' hierarchy so that it regularizes match consistency at multiple spatial extents—ranging from an entire image, to coarse grid cells, to every single pixel. A key idea behind my approach is to

strike a balance between robustness to image variations on the one hand, and accurate localization of pixel correspondences on the other. I achieve this balance through a pyramid graph: larger spatial nodes offer greater regularization when appearance matches are ambiguous, while smaller spatial nodes help localize matches with fine detail. At the same time, the DSP model naturally leads to a fast hierarchical optimization procedure, producing noticeably faster matching than today's popular matching methods.

Figure 6.1 contrasts our DSP model with existing ones. In particular, I would like to point out the difference between the proposed DSP model (Figure 6.1 (a)) and SIFT Flow model (Figure 6.1 (b)), since SIFT Flow also takes a hierarchical model for matching. Key contrast is that our model is based on multi-level regions of various spatial extents across the hierarchy, whereas SIFT Flow relies on only pixels that repeats the identical pixel-grid graph structure across the image pyramid. In the results (Sec. 6.3), I will show our region-based approach makes substantial gains over such pixel-based models for dense matching.

6.2 Deformable Spatial Pyramid Matching

In this section, I present my approach for fast dense pixel correspondences. My method takes two images as input and matches *every* pixel between them. I first define my deformable spatial pyramid (DSP) graph for dense pixel matching (Sec. 6.2.1). Then, I define the matching objective I will optimize on that pyramid (Sec. 6.2.2). Finally, I discuss technical issues, focusing on efficient computation (Sec. 6.2.3).

6.2.1 Pyramid Graph Model

To build the spatial pyramid, we start from the entire image and divide it into different sub-regions and keep dividing until we reach the predefined number of pyramid levels. In my implementation, I explore both grid-based and region-based hierarchies. For grid-based one, I use four rectangular grid cells to divide a region at each level (I use 3 levels). For region-based

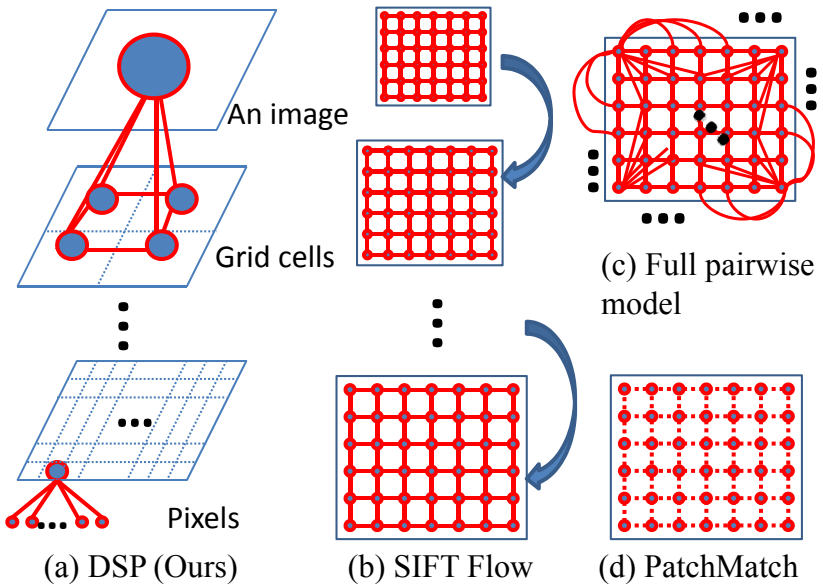
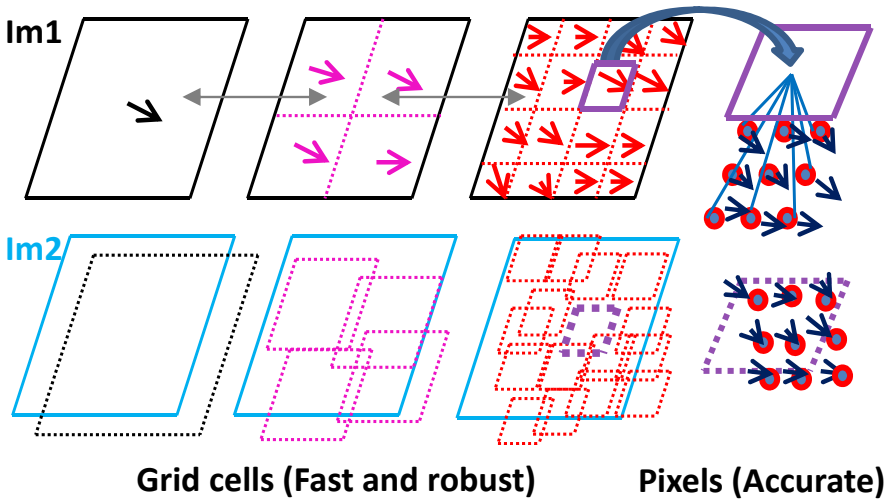


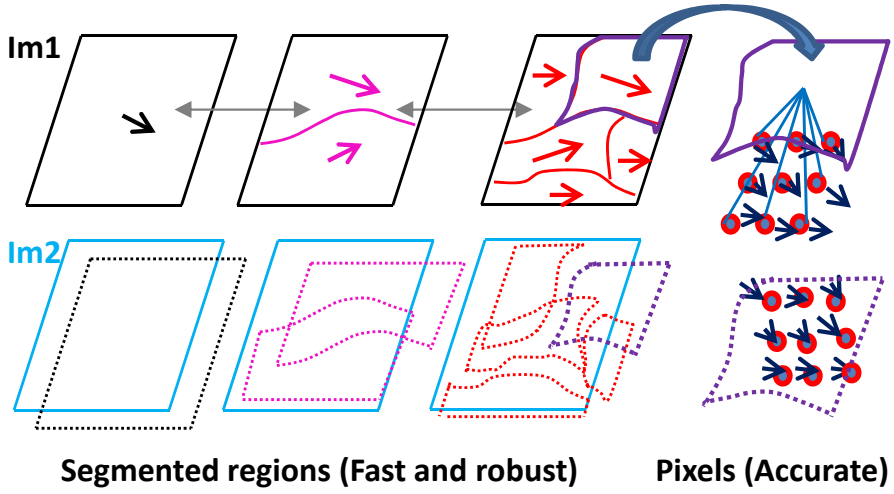
Figure 6.1: Graph representations of different matching models. A circle denotes a graph node and its size represents its spatial extent. Edges denote geometric distortion terms. **(a)** *Deformable spatial pyramid* (proposed): uses spatial support at various extents. **(b)** *Hierarchical pixel model* [62]: the matching result from a lower resolution image guides the matching in the next resolution. **(c)** *Full pairwise model* [9, 57]: every pair of nodes is linked for strong geometric regularization (though limited to sparse nodes). **(d)** *Pixel model with implicit smoothness* [6]: geometric smoothness is enforced in an indirect manner via a spatially-constrained correspondence search (dotted lines denote no explicit links). Aside from the proposed model (a), all graphs are defined on a pixel grid.

one, I use bottom-up segments from a hierarchical segmentation method [3] as region input. The grid pyramid has the advantage of efficient computation due to its regular structure, while the region pyramid can explicitly consider the visual cues in images when building the graphical model. In Sec. 6.3.4, I compare two models in terms of both computational cost and matching accuracy. In addition to those hierarchical pyramids of grid cells or segmented regions, I further add one more layer, a pixel-level layer, such that the finest cells are one pixel in width.

Then, I represent the pyramid with a graph. See Figures 6.1 (a) and 6.2. Each grid cell (or segmented regions) and pixel is a node, and edges link all neighboring nodes within the same level, as well as parent-child nodes across adjacent levels. For the pixel level, however, I do not link neighboring pixels; each pixel is linked only to its parent node. This saves us a lot of edge connections that would otherwise dominate run-time during optimization.



(a) DSP with regular grid regions



(b) DSP with segmented regions

Figure 6.2: Sketch of the proposed DSP matching method. I show two models: (a) DSP with regular grid cells (default implementation), and (b) DSP with generic hierarchical segmentation. I focus most results on a DSP model with regular grid regions since I can exploit its regularity for efficient computation. However, the DSP model can take any hierarchical regions as input, as shown in (b). In both (a) and (b), first row shows image 1's pyramid graph; second row shows the match solution on image 2. Single-sided arrow in a node denotes its flow vector $\mathbf{t}_i$; double-sided arrows between pyramid levels imply parent-child connections between them (intra-level edges are also used but not displayed). We solve the matching problem at different sizes of spatial nodes in two layers. Cells in the grid-layer (left three images) provide reliable (yet fast) initial correspondences that are robust to image variations due to their larger spatial support. Guided by the grid-layer initial solution, we efficiently find accurate pixel-level correspondences (rightmost image). Best viewed in color.

6.2.2 Matching Objective

Now, I define my matching objective for the proposed pyramid graph. I start with a basic formulation for matching images at a single fixed scale, and then extend it to multi-scale matching.

Fixed-Scale Matching Objective Let $\mathbf{p}_i = (x_i, y_i)$ denote the location of node i in the pyramid graph, which is given by the node's center coordinate. Let $\mathbf{t}_i = (u_i, v_i)$ be the translation of node i from the first to the second image. We want to find the optimal translations of each node in the first image to match it to the second image, by minimizing the energy function:

$$E(\mathbf{t}) = \sum_i D_i(\mathbf{t}_i) + \alpha \sum_{i,j \in \mathcal{N}} V_{ij}(\mathbf{t}_i, \mathbf{t}_j), \tag{6.1}$$

where D_i is a data term, V_{ij} is a smoothness term, α is a constant weight, and $\mathcal{N}$ denotes pairs of nodes linked by graph edges. Recall that edges span across pyramid levels, as well as within pyramid levels.

The data term D_i measures the appearance matching cost of node i at translation $\mathbf{t}_i$. It is defined as the average distance between local descriptors (e.g, SIFT) within node i in the first image to those located within a region of the same scale in the second image after shifting by $\mathbf{t}_i$:

$$D_i(\mathbf{t}_i) = \frac{1}{z} \sum_{\mathbf{q}} \min(\|d_1(\mathbf{q}) - d_2(\mathbf{q} + \mathbf{t}_i)\|_1, \lambda), \tag{6.2}$$

where $\mathbf{q}$ denotes pixel coordinates within a node i from which local descriptors were extracted, z is the total number of descriptors, and d_1 and d_2 are descriptors extracted at the locations $\mathbf{q}$ and $\mathbf{q} + \mathbf{t}_i$ in the first and second image, respectively. For robustness to outliers, we use a truncated L1 norm for descriptor distance with a threshold λ. Note that $z = 1$ at the pixel layer, where $\mathbf{q}$ contains a single point.

The smoothness term V_{ij} regularizes the solution by penalizing large discrepancies in the

matching locations of neighboring nodes: $V_{ij} = \min(\|\mathbf{t}_i - \mathbf{t}_j\|_1, \gamma)$. I again use a truncated L1 norm with a threshold γ.

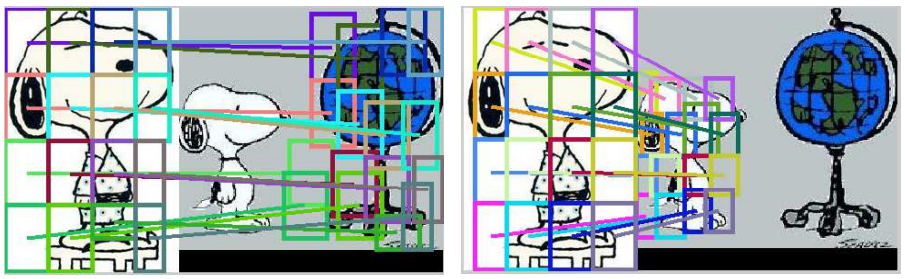
How does this objective differ from the conventional pixel-wise model? There are three main factors. First of all, graph nodes in my model are defined by cells of varying spatial extents, whereas in prior models they are restricted to pixels. This allows us to overcome appearance match ambiguities without committing to a single spatial scale. Second, the data term aggregates many local SIFT matches within each node, as opposed to using a single match at each individual pixel. This greatly enhances robustness to image variations. Third, I explicitly link the nodes of different spatial extents to impose smoothness, striking a balance between strong regularization by the larger nodes and accurate localization by the finer nodes.

To minimize the main objective function (Eq. 6.1), we use loopy belief propagation to find the optimal correspondence of each node (see Sec. 6.2.3 for details). Note that the resulting matching is asymmetric, mapping all of the nodes in the first image to some (possibly subset of) positions in the second image. Furthermore, while my method returns matches for all nodes in all levels of the pyramid, we are generally interested in the final dense matches at the pixel level. As mentioned in the previous section (Sec. 6.1), these dense pixel matches have many new applications for vision and graphics. In this work, I use dense pixel matches for predicting class label of every pixel for semantic image segmentation (Sec. 6.3).

Multi-Scale Extension Thus far, I assume the matching is done at a fixed scale: each grid cell is matched to another region of the same size. Now, I extend the objective to allow nodes to be matched across different scales:

$$E(\mathbf{t}, \mathbf{s}) = \sum_i D_i(\mathbf{t}_i, \mathbf{s}_i) + \alpha \sum_{i,j \in \mathcal{N}} V_{ij}(\mathbf{t}_i, \mathbf{t}_j) + \beta \sum_{i,j \in \mathcal{N}} W_{ij}(\mathbf{s}_i, \mathbf{s}_j). \tag{6.3}$$

Eq. 6.3 is a multi-scale extension of Eq. 6.1. We add a scale variable $\mathbf{s}_i$ for each node and introduce a scale smoothness term $W_{ij} = \|\mathbf{s}_i - \mathbf{s}_j\|_1$ with an associated weight constant β. The



(a) Fixed-scale match (b) Multi-scale match

Figure 6.3: Comparing my fixed- and multi-scale matches. For visibility, I show matches only at a single level in the pyramid. In (a), the match for a node in the first image remains at the same fixed scale in the second image. In (b), the multi-scale objective allows the size of each node to optimally adjust when matched.

scale variable is allowed to take discrete values from a specified range of scale variations (to be defined below). The data term is also transformed into a multi-variate function defined as:

$$D_i(\mathbf{t}_i, \mathbf{s}_i) = \frac{1}{z} \sum_{\mathbf{q}} \min(\|d_1(\mathbf{q}) - d_2(\mathbf{s}_i(\mathbf{q} + \mathbf{t}_i))\|_1, \lambda), \tag{6.4}$$

where we see the corresponding location of descriptor d_2 for a descriptor d_1 is now determined by a translation $\mathbf{t}_i$ followed by a scaling $\mathbf{s}_i$.

Note that this formulation allows each node to take its own optimal scale, rather than determine the best global scale between two images. This is beneficial when an image includes both foreground and background objects of different scales, or when individual objects have different sizes. See Figure 6.3.

Dense correspondence for generic image matching is often treated at a fixed scale, though there are some multi-scale implementations in related work. PatchMatch has a multi-scale extension that expands the correspondence search range according to the scale of the previously found match [6]. As in the fixed-scale case, my method has the advantage of modeling geometric distortion and match consistency across multiple spatial extents. While I handle scale adaptation through the matching objective, one can alternatively consider representing each pixel with a set of SIFTs at multiple scales [39]; that feature could potentially be plugged into any matching method, including ours, though its extraction time is far higher than typical fixed-

scale features. The proposed multi-scale matching is efficient and works even with fixed-scale features.

6.2.3 Efficient Computation

For dense matching, computation time is naturally a big concern for scalability. Here I explain how I maintain efficiency both through the problem design and some technical implementation details.

There are two major components that take most of the time: (1) computing descriptor distances at every possible translation and (2) optimization via belief propagation (BP). For the descriptor distances, the complexity is $O(mlk)$, where m is the number of descriptors extracted in the first image, l is the number of possible translations, and k is the descriptor dimension. For BP, we use a generalized distance transform technique, which reduces the cost of message passing between nodes from $O(l^2)$ to $O(l)$ [29]. Even so, BP's overall run-time is $O(nl)$, where n is the number of nodes in the graph. Thus, the total cost of my method is $O(mlk + nl)$ time. Note that n, m, and l are all on the order of the number of pixels (i.e., $\sim 10^5 - 10^6$); if solving the problem at once, it is far from efficient.

Therefore, I use a hierarchical approach to improve efficiency. We initialize the solution by running BP for a graph built on all the nodes except the pixel-level ones (which I will call first-layer), and then refine it at the pixel nodes (which I will call second-layer). In Figure 6.2, the first three images on the left comprise the first layer, and the fourth depicts the second (pixel) layer.

Compared to SIFT Flow's hierarchical variant [62], ours runs an order of magnitude faster, as I will show in the results. The key reason is the two methods' differing matching objectives: ours is on a pyramid, theirs is a pixel model. Hierarchical SIFT Flow solves BP on the *pixel grids* in the image pyramid; starting from a downsampled image, it progressively narrows down the possible solution space as it moves to the finer images, reducing the number of possible

translations l. However, n and m are still on the order of the number of pixels. In contrast, the number of nodes in my first-layer BP is just tens. Moreover, I observe that sparse descriptor sampling is enough for the first-layer BP: as long as a grid cell includes $\sim$100s of local descriptors within it, its average descriptor distance for the data term (Eq. 6.2) provides a reliable matching cost. Thus, I do not need dense descriptors in the first-layer BP, substantially reducing m.

In addition, my decision not to link edges between pixels (i.e., no loopy graph at the pixel layer) means the second-layer solution can be computed very efficiently in a non-iterative manner. Once I run the first-layer BP, the optimal translation $\mathbf{t}_i$ at a pixel-level node i is simply determined by: $\mathbf{t}_i = \arg\min_{\mathbf{t}}(D_i(\mathbf{t}) + \alpha V_{ij}(\mathbf{t}, \mathbf{t}_j))$, where a node j is a parent grid cell of a pixel node i, and $\mathbf{t}_j$ is a fixed value obtained from the first-layer BP.

The proposed multi-scale extension incurs additional cost due to the scale smoothness and multi-variate data terms. The former affects message passing; the latter affects the descriptor distance computation. In a naive implementation, both linearly increase the cost in terms of the number of the scales considered. For the data term, however, we can avoid repeating computation per scale. Once we obtain $D_i(\mathbf{t}_i, \mathbf{s}_i = 1.0)$ by computing the pairwise descriptor distance at $\mathbf{s}_i = 1.0$, it can be re-used for all other scales; the data term $D_i(\mathbf{t}_i, \mathbf{s}_i)$ at scale $\mathbf{s}_i$ maps to $D_i((\mathbf{s}_i - 1)\mathbf{q} + \mathbf{s}_i\mathbf{t}_i, \mathbf{s}_i = 1.0)$ of the reference scale (see the next paragraph for details). This significantly reduces computation time, in that SIFT distances dominate the BP optimization since m is much higher than the number of nodes in the first-layer BP.

Mapping data terms across different scales: To re-use descriptor distances for the multi-scale matching, I define a mapping between data terms across different scales: the data term $D_i(\mathbf{t}_i, \mathbf{s}_i)$ at scale $\mathbf{s}_i$ maps to $D_i((\mathbf{s}_i - 1)\mathbf{q} + \mathbf{s}_i\mathbf{t}_i, \mathbf{s}_i = 1.0)$ of the reference scale. I derive it as follows.

The data term $D_i(\mathbf{t}_i, \mathbf{s}_i)$ computes a descriptor distance between $d_1(\mathbf{q})$ at a point $\mathbf{q}$ of

the first image and $d_2(\mathbf{s}_i(\mathbf{q} + \mathbf{t}_i))$ in the second image (see Eq. 4 in the main paper). Here, the corresponding location of descriptor d_2 for a descriptor d_1 is determined by a translation $\mathbf{t}_i$ followed by a scaling $\mathbf{s}_i$ on the point $\mathbf{q}$.

However, if we suppose those two corresponding locations are associated by a common reference scale ($\mathbf{s}_i = 1.0$), their translation can be represented by a simple coordinate difference between them: $\mathbf{s}_i(\mathbf{q} + \mathbf{t}_i) - \mathbf{q} = (\mathbf{s}_i - 1)\mathbf{q} + \mathbf{s}_i\mathbf{t}_i$. That is, a translation $\mathbf{t}_i$ at a scale $\mathbf{s}_i$ is equivalent to the translation $(\mathbf{s}_i - 1)\mathbf{q} + \mathbf{s}_i\mathbf{t}_i$ at the reference scale. As a result, once we have computed the data term at the reference scale, we can map it to other scales without repeating the computation per scale. Algorithm 4 depicts the implementation of our DSP matching method in pseudo-code.

Algorithm 4: DSP match

Data: Two images I_1 and I_2
Result: Dense pixel correspondences between I_1 and I_2

```
/* Extract features (e.g., SIFTs) for every pixel      */
```

input : I_1 and I_2
output: $\mathbf{F_1}$ and $\mathbf{F_2}$, each of which is a set of dense features extracted from I_1 and I_2 respectively.
for $i = 1$ **to** 2 **do**
  foreach *pixel* p_j *in* I_i **do**
    Extract a feature f_j with a fixed size patch (e.g., 16×16 pixels) at l_j;
    Add f_j and its location l_j into $\mathbf{F_i}$;
  end
end

```
/* Build pyramid graph                                 */
```

Divide I_1 into four sub-rectangles and repeat dividing up to n levels;
Build a pyramid graph in grid-layer: Link neighboring grid cells within the level and parent-child cells across the adjacent levels;
Add a pixel-layer: Attach pixels to their parent cell;

Algorithm 4: DSP match: continued

Data: Two images I_1 and I_2
Result: Dense pixel correspondences between I_1 and I_2

`/* Grid-layer optimization */`
input : $\mathbf{F_1}$, $\mathbf{F_2}$, and a set of pre-defined translations $\mathbf{T}$ and scales $\mathbf{S}$
output: Matched cells' translations and scales from I_1 to I_2
foreach *translation* t_i *in* $\mathbf{T}$ **do** `/* fixed-scale, i.e.,` $scale = 1$ `*/`
  Compute an integral image for feature distances between $\mathbf{F_1}$ and $\mathbf{F_2}$ at translation t_i;
  foreach *grid cell* g *in the pyramid graph* **do**
    Compute data-term $D(t_i)$ of g using the integral image;
  end
end
if *multi-scale* **then** `/* Multi-scale extension */`
  foreach *scale* s_i *in* $\mathbf{S}$ **do**
    foreach *translation* t_j *in* $\mathbf{t}$ **do**
      Mapping the feature distances from the original scale (i.e., $scale = 1$) to those at scale s_i (see Sec. 6.2.3);
      Compute an integral image at scale s_i and translation t_j;
      foreach *grid cell* g *in the pyramid graph* **do**
        Compute data-term $D(t_i, s_j)$ of g using the integral image;
      end
    end
  end
end
Run belief propagation to solve the objective in Eq. 6.3, which provides optimal translations $\mathbf{T_g}$ and scales $\mathbf{S_g}$ of matched grid cells from I_1 to I_2;
`/* Pixel-layer optimization */`
input : $\mathbf{T_g}$
output: A set of match pixels' translations $\mathbf{T_p}$ from I_1 to I_2
foreach *pixel* p_i *in* I_1 **do**
  Compute an optimal match translation t_i by a closed-form solution in Sec. 6.2.3;
  Add t_i into $\mathbf{T_p}$;
end

return $\mathbf{T_p}$

6.3 Results

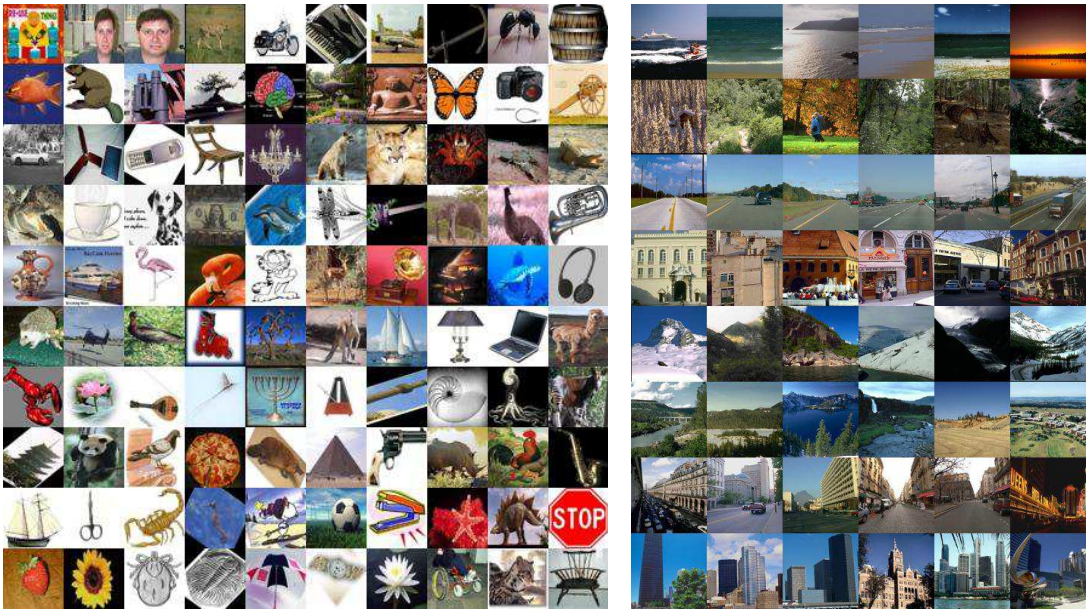
The main goals of the experiments are 1) to evaluate raw matching quality (Sec. 6.3.1), 2) to validate my method applied to exemplar-based sematic segmentation (Sec. 6.3.2), 3) to verify the impact of my multi-scale extension (Sec. 6.3.3), 4) to compare grid-based and region-based hierarchies in both run-time and matching accuracy (Sec. 6.3.4), and 5) to show how various spatial supports from my pyramid model achieve a balance between robust matching and accurate localization (Sec. 6.3.5).

I compare my deformable spatial pyramid (DSP) approach to state-of-the-art dense pixel matching methods, SIFT Flow [62] (**SF**) and PatchMatch [6] (**PM**), using the authors' publicly available code. SIFT Flow adopts a hierarchical optimization as ours, but its graphical model builds on a flat pixel grid at every level of image pyramid. This contrasts to our model built on multi-resolution region hierarchy. PatchMatch avoids an explicit optimization; instead it relies on a randomized search, focusing on fast computation. I will show DSP's speed advantage over PatchMatch. Unless mentioned otherwise, I use the results from the grid-based pyramid model when comparing to the baselines.

I use two datasets: the Caltech-101 and LabelMe Outdoor (LMO) [61]. Figure 6.4 shows some example images from each dataset.

Implementation details: I fix the parameters of my method for all experiments: $\alpha = 0.005$ in Eq. 6.1, $\gamma = 0.25$, and $\lambda = 500$. For multi-scale, I set $\alpha = 0.005$ and $\beta = 0.005$ in Eq. 6.3. I extract SIFT descriptors of 16x16 patch size at every pixel using VLFeat [94]. I apply PCA to the extracted SIFT descriptors, reducing the dimension to 20. This reduction saves about 1 second per image match without losing matching accuracy.[2] For multi-scale match, I use seven scales between 0.5 and 2.0—I choose the search scale as an exponent of $2^{\frac{i-4}{3}}$, where $i = 1, ..., 7$.

[2] I use the same PCA-SIFT for ours and PatchMatch. For SIFT Flow, however, I use the authors' custom code to extract SIFT; I do so because I observed SIFT Flow loses accuracy when using PCA-SIFT.



(a) Caltech-101 (b) LabelMe Outdoor

Figure 6.4: Example images from the datasets used in our experiments. For each dataset, we randomly display one or two images from each object class of the dataset. Caltech-101 dataset is for object matching under intra-class variations. LabelMe Outdoor dataset includes various outdoor images for scene matching.

Evaluation metrics: To measure image matching quality, I use label transfer accuracy (LT-ACC) between pixel correspondences [61]. Given a test and an exemplar image, I transfer the annotated class labels of the exemplar pixels to the test ones via pixel correspondences, and count how many pixels in the test image are correctly labeled.

For object matching in Caltech-101 dataset, I also use the intersection over union (IOU) metric [25]. Compared to LT-ACC, this metric allows us to isolate the matching quality for the foreground object, separate from the irrelevant background pixels.

I also evaluate the localization error (LOC-ERR) of corresponding pixel positions. Since there are no available ground-truth pixel matching positions between images, I obtain pixel locations using an object bounding box: pixel locations are given by the normalized coordinates with respect to the box's position and size.

Formally, I define the localization error as follows. I first designate each image's pixel

Approach	LT-ACC	IOU	LOC-ERR	Time (s)
DSP (Ours)	**0.732**	**0.482**	**0.115**	**0.65**
SIFT Flow [62]	0.680	0.450	0.162	12.8
PatchMatch [6]	0.646	0.375	0.238	1.03

Table 6.1: Object matching on the Caltech-101. DSP outperforms the state-of-the-art methods in both matching accuracy and speed.

coordinate using its ground-truth object bounding box: the pixel coordinate of an object in each image is set such that the top-left of the bounding box becomes the origin and x-and y-coordinate are normalized by width and height of the box respectively. Then, I define the localization error of two matched pixels as: $e = 0.5(|x_1 - x_2| + |y_1 - y_2|)$, where (x_1, y_1) is the pixel coordinate of the first image and (x_2, y_2) is its corresponding location in the second image. We apply this metric to Caltech-101 dataset as it provides bounding box annotations for the foreground objects. Note that LOC-ERR metric is evaluated for the foreground pixels only, as we define bounding box coordinates only for the pixels inside the box.

6.3.1 Raw Image Matching Accuracy

In this section, I evaluate raw pixel matching quality in two different tasks: object matching and scene matching.

Object matching under intra-class variations: For this experiment, I randomly pick 15 pairs of images for each object class in the Caltech-101 (total 1,515 pairs of images). Each image has ground-truth pixel labels for the foreground object. Table 6.1 shows the result. DSP outperforms SIFT Flow by 5 points in label transfer accuracy, yet is about 25 times faster. DSP achieves a 9 point gain over PatchMatch, in about half the runtime. Its localization error and IOU scores are also better.

Figure 6.5 shows example matches by the different methods. We see that DSP works robustly under image variations like appearance change and background clutter. On the other hand, the two existing methods—both of which rely on only local pixel-level appearance—

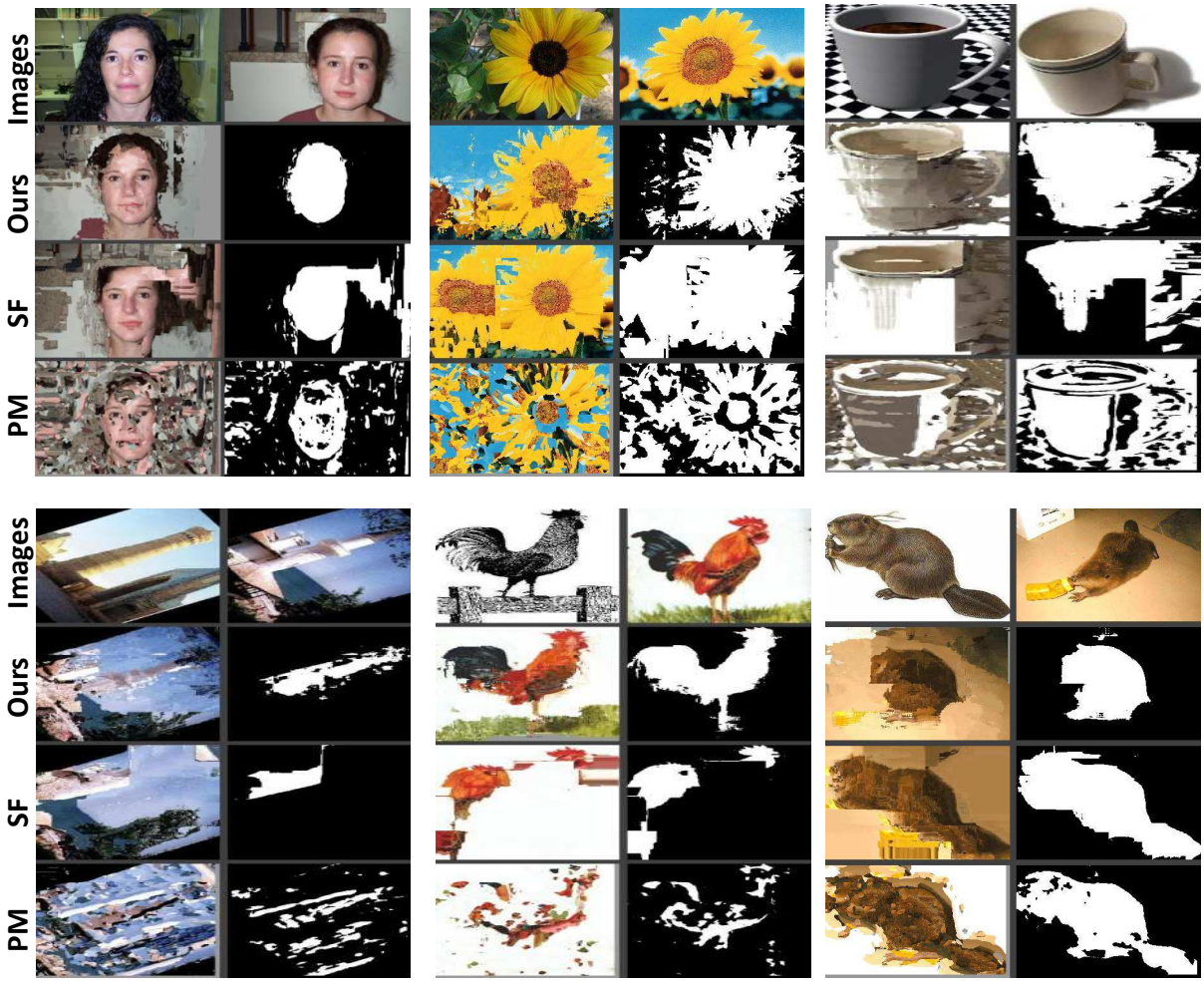


Figure 6.5: Example object matches per method. In each match example, the left image shows the result of warping the second image to the first via pixel correspondences, and the right one shows the transferred pixel labels for the first image (white: foreground, black: background). We see DSP works robustly under image variations like background clutter (1st and 2nd examples in the first row), appearance change (4th and 5th ones in the second row). Further, even when objects lack texture (3rd example in the first row), ours finds reliable correspondences, exploiting global object structure. However, the single-scale version of DSP fails when objects undergo substantial scale variation (6th example in the second row). Best viewed in color.

get lost under the substantial image variations. This shows how the proposed spatial pyramid graph successfully imposes geometric regularization from various spatial extents, overcoming the matching ambiguity that can arise if considering local pixels alone. We also can see some differences between the two existing models. PatchMatch abandons explicit geometric smoothness for speed. However, this tends to hurt matching quality—the matching positions of even nearby pixels are quite dithered, making the results noisy. On the other hand, SIFT Flow imposes stronger smoothness by MRF connections between nearby pixels, providing visually more pleasing results. In effect, DSP combines the strengths of the other two. Like PatchMatch, I remove neighbor links in the pixel-level optimization for efficiency. However, I can do this without hurting accuracy since larger spatial nodes in my model enforce a proper smoothness on pixels.

Scene matching: Whereas the object matching task is concerned with foreground/background matches, in the scene matching task each pixel in an exemplar is annotated with one of multiple class labels. Here I use the LMO dataset, which annotates pixels as one of 33 class labels (e.g., river, car, grass, building). I randomly split the test and exemplar images in half (1,344 images each). For each test image, we first find the exemplar image that is its nearest neighbor in GIST space. Then, we match pixels between the test image and the selected exemplar. When measuring label transfer accuracy, I only consider the matchable pixels that belong to the classes common to both images. This setting is similar to the one in [61].

Table 6.2 shows the results.[3] Again, DSP outperforms the current methods. Figure 6.6 compares some example scene matches. We see that DSP better preserves the scene structure; for example, the horizons (1st, 3rd, and 6th examples) and skylines (2nd, 4th, and 5th) are robustly estimated.

[3]The IOU and LOC-ACC metrics assume a figure-ground setting, and hence are not applicable here.

Approach	LT-ACC	Time (s)
DSP (Ours)	**0.706**	**0.360**
SIFT Flow [62]	0.672	11.52
PatchMatch [6]	0.607	0.877

Table 6.2: Scene matching on the LMO dataset. DSP outperforms the current methods in both accuracy and speed.

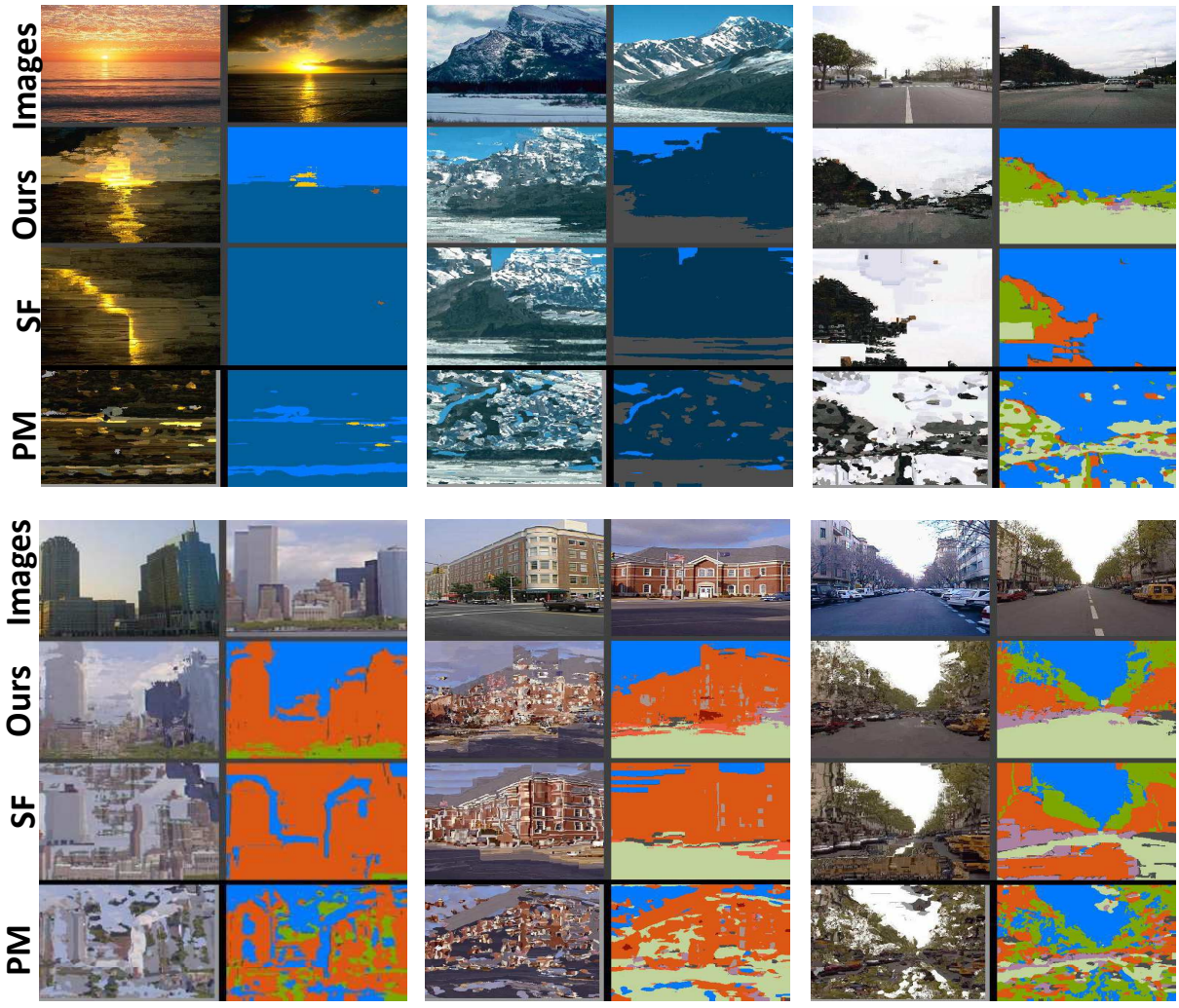


Figure 6.6: Example scene matches per method. Displayed as in Fig. 6.5, except here the scenes have multiple labels (not just fg/bg). Pixel labels are marked by colors, denoting one of the 33 classes in the LMO dataset. Best viewed in color.

6.3.2 Semantic Segmentation by Matching Pixels

Next, I apply the DSP method to a semantic segmentation task, following the protocol in [61]. To explain briefly, we use DSP to match a test image to multiple exemplar images, where pixels in the exemplars are annotated by ground-truth class labels. Then, the best matching scores (SIFT descriptor distances) between each test pixel and its corresponding exemplar pixels define the class label likelihood of the test pixel. Using this label likelihood, I use a standard MRF to assign a class label to each pixel. See [61] for details. Building on this common framework, I test how the different matching methods influence segmentation quality.

Category-specific figure-ground segmentation: First, we perform binary figure-ground segmentation on Caltech. I randomly select 15/15 test/exemplar images from each class. I match a test image to exemplars from the same class, and perform figure-ground segmentation with an MRF as described above. Table 6.3 shows the result. My DSP outperforms the current methods substantially. Figure 6.7 shows example segmentation results. We see that DSP successfully delineates foreground objects from confusing backgrounds.

Approach	LT-ACC	IOU
DSP (Ours)	**0.868**	**0.694**
SIFT Flow [62]	0.820	0.641
PatchMatch [6]	0.816	0.604

Table 6.3: Figure-ground segmentation results in Caltech-101.

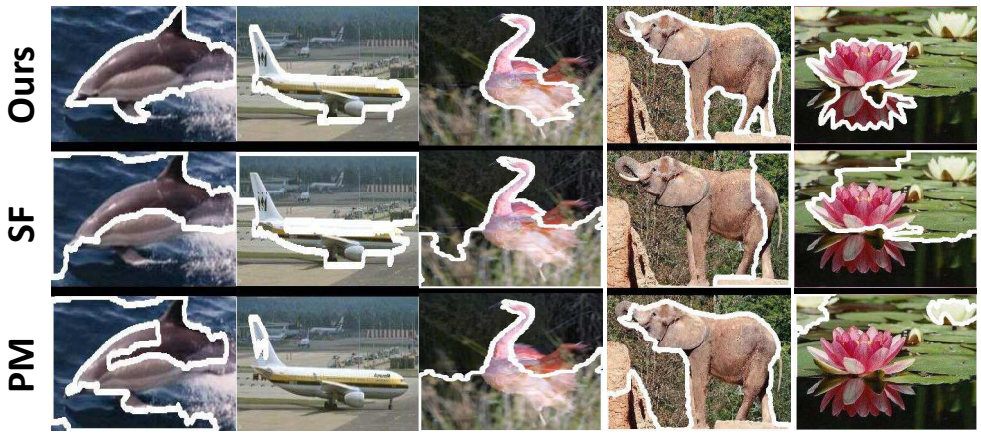


Figure 6.7: Example figure-ground segmentations on Caltech-101.

Multi-class pixel labeling: Next, I perform semantic segmentation on the LMO dataset. For each test image, I first retrieve an initial exemplar "shortlist", following [61]. The test image is matched against only the shortlist exemplars to estimate the class likelihoods.[4] I test three different ways to define the shortlist: (1) using the ground truth (GT), (2) using GIST neighbors (GIST), and (3) using an SVM to classify the images into one of the 8 LMO scene categories, and then retrieving GIST neighbors among only exemplars from that scene label (SVM).

Table 6.4 shows the results. The segmentation accuracy depends on the shortlist mechanism, for all methods. When using ground-truth annotations to choose the shortlist, my method clearly outperforms the others. On the other hand, when using automatic methods to generate the shortlist (GIST and SVM), my gain becomes smaller. This is because (1) the shortlist misses reasonable exemplar images that share class labels with the test image and (2) SIFT features may not be strong enough to discriminate confusing classes in a noisy shortlist—some classes (e.g., grass, field, and tree) are too similar to be distinguished by SIFT match scores alone. Again, my method is more efficient; 15-20 times faster than SIFT Flow, and about twice as fast as Patch Match. Figure 6.8 shows example segmentation results.

Approach	LT-ACC (GT)	LT-ACC (GIST)	LT-ACC (SVM)
DSP (Ours)	**0.868**	0.745	**0.761**
SIFT Flow [62]	0.834	**0.759**	0.753
PatchMatch [6]	0.761	0.704	0.701

Table 6.4: Semantic segmentation results on the LMO dataset.

[4] My test/exemplar split, shortlist, and MRF are all identical to those in [61], except I do not exploit any prior knowledge (e.g., likelihood of possible locations of each class in the image) to augment the cost function of the MRF. Instead, I only use match scores in order to most directly compare the impact of the three matching methods.

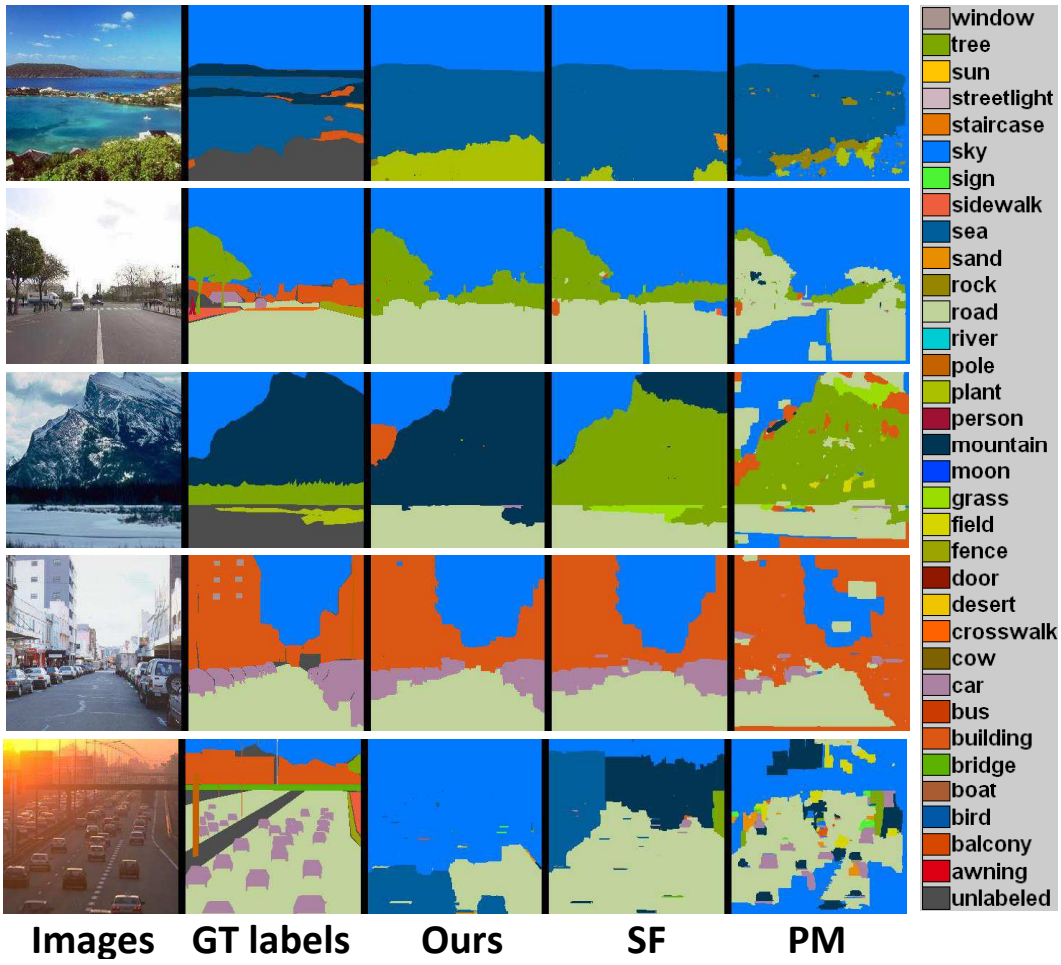


Figure 6.8: Example semantic segmentations on the LMO dataset. My DSP and SIFT Flow (SF) both work reasonably on this dataset, though my segmentation is more consistent to the image's scene layout (e.g., the first and the third row). On the other hand, PatchMatch (PM) results are quite noisy. The last row shows the failure case, where I fail to segment small objects (cars).

6.3.3 Multi-Scale Matching

In this section, I show the results of my multi-scale formulation. For this experiment, I test using the same image pairs from Caltech as used in Sec. 6.3.1. I compare my multi-scale method to various baselines, including all the fixed-scale methods in the previous section and PatchMatch with its multi-scale option on.

Figure 6.9 plots the matching accuracy as a function of scale variation. The scale varia-

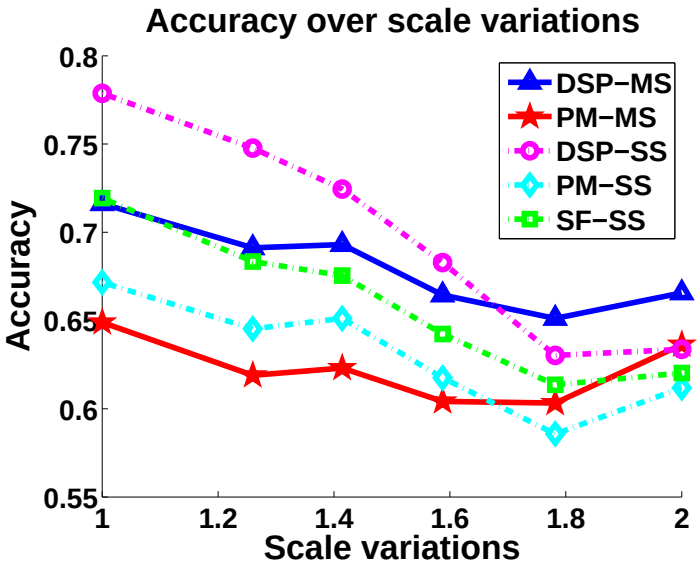


Figure 6.9: Matching accuracy as a function of scale variations. MS and SS denote multi-scale and single-scale, respectively.

tion between two objects is defined by: $\frac{\max(O_1,O_2)}{\min(O_1,O_2)}$, where O_1 and O_2 are the size of matched objects in the first and the second image respectively. We see that the curves from multi-scale methods (DSP-MS and PM-MS) are flatter than the single-scale ones, verifying their relative scale tolerance. In addition, my multi-scale method (DSP-MS) outperforms multi-scale PatchMatch (PM-MS) by a substantial margin. However, we also see our curve is not perfectly flat across the scale changes. This is because scale is not the only factor that affects the matching. In fact, as scale variation increases, I observe that objects undergo more variations in viewpoint or shapes as well, making the matching more challenging.

Figure 6.10 shows some matching examples by my multi-scale method, compared to my single-scale counterpart. The examples show that my multi-scale matching is flexible to scale variations.

6.3.4 Use of Hierarchical Segmentation

So far I used regular grid cells for building DSP graph. However, DSP model actually has no restrictions on the types of input regions. In this section, I test DSP method that builds

Figure 6.10: Example matching results by my multi-scale matching. For comparison, I also show results from my fixed-scale version. My multi-scale method successfully finds the correct scale between objects, providing accurate matching. On the other hand, a single-scale one prefers the fixed size between objects, causing gross errors: e.g., in the 3rd example, Snoopy matches to a globe since they have similar size.

on segmented regions obtained by state-of-the-art hierarchical segmentation [3], which was already used in Chapter 5 for segmentation-driven matching. To get hierarchical regions, I generate three levels of segments using three different threshold values in [3]. Note that hierarchical segmentation inherently provides nested multi-scale regions. Also, these regions form adaptively to the visual cues in an image, which differs from the regular grid pyramid that has fixed structure.

Table 6.5 and Table 6.6 compare segmentation-based DSP to the original implementation. We see that DSP from segmented regions provides slightly better accuracy than the default implementation that is based on regular grid regions. This may be because segmented regions can help separate noisy background pixels from foreground ones when computing matching

Approach	LT-ACC	Time (s)
DSP (Grid Cells)	0.732	**0.65**
DSP (Segmented Regions)	**0.742**	2.57

Table 6.5: Comparison of grid cells and segmented regions for object matching on the Caltech-101. DSP from segmented regions slightly outperforms the one from regular grid cells, at the cost of run-time.

Approach	LT-ACC	Time (s)
DSP (Grid Cells)	0.706	**0.36**
DSP (Segmented Regions)	**0.709**	2.36

Table 6.6: Comparison of grid cells and segmented regions for scene matching on the LMO dataset.

cost. However, run-time becomes higher because we can no longer exploit the regularity of grid cells for speed-up: non-regular regions allow neither integral image technique for efficiently computing regions' matching cost, nor fixed graph data-structure that is fast to build.

6.3.5 Balance across Different Levels of Pyramid

Finally, I show how various spatial supports from our pyramid model achieve a balance between robustness to image variations and accurate localization with fine detail. To this end, I compare matching accuracy from different spatial extents by varying the number of pyramid levels.

Table 6.7 summarizes the results. Each row in the table adds another finer level to the pyramid. The accuracy is then evaluated using the matching given at the finest level in that pyramid, as I did in the previous sections. We see that larger spatial nodes from lower pyramid levels provide better LT-ACC, whereas smaller nodes from higher levels offer better IOU. Given that LT-ACC takes all the pixels for evaluation whereas IOU accounts for foreground pixels only, these results point out (1) larger spatial nodes regularize the matching ambiguity from noisy background pixels, reducing the error at the background; (2) smaller nodes enhance the matching quality on the foreground pixels with fine details.

Pyramid levels	LT-ACC	IOU
level 1	**0.745**	0.442
level 1 + 2	0.745	0.462
level 1 + 2 + 3	0.736	0.477
level 1 + 2 + 3 + pixels	0.732	**0.482**

Table 6.7: Matching accuracy in Caltech 101 in terms of the number of pyramid levels. The first three results come from grid-layer pyramid, in which pyramid level increases from 1 to 2 to 3, respectively. The last row denotes the result of our original implementation in the main paper, adding a pixel-level layer on top of three levels of grid-layer pyramid.

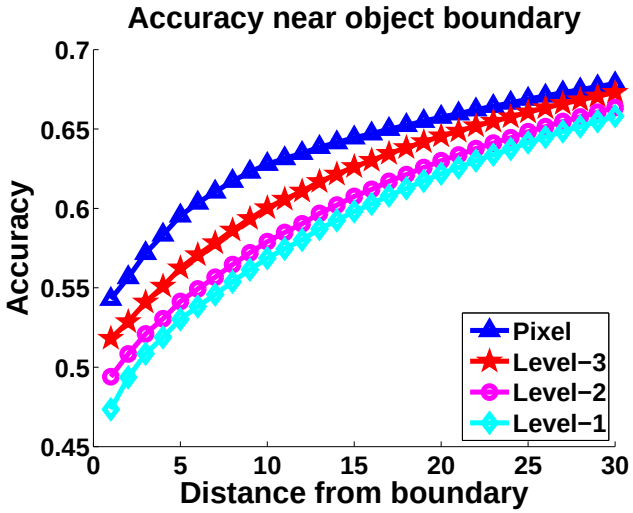


Figure 6.11: Matching accuracy near the object boundary. We evaluate matching accuracy among different pyramid levels as a function of pixel distance from the object boundary (up to 30 pixels). A pyramid with finer spatial nodes (e.g., Pixel) achieves better accuracy for the pixels near the object boundaries.

Figure 6.11 supports our point further, where I evaluate the matching accuracy for the pixels near the object boundaries. We see that as the level of pyramid gets higher, it achieves a larger gain near the object boundaries, demonstrating smaller spatial nodes (e.g., pixels) do better at localizing the finer object structures such as object boundaries.

6.4 Discussion

In this chapter, I introduced a deformable spatial pyramid (DSP) model that exploits regions' hierarchical structure for fast dense correspondences across different objects or scenes.

For fast computation, I developed a coarse-to-fine optimization method on the pyramid graph. Through extensive evaluations, I showed that 1) various spatial supports by our spatial pyramid improve matching quality, striking a balance between geometric regularization and accurate localization, 2) the proposed pyramid structure permits efficient hierarchical optimization, enabling fast dense matching, and 3) the proposed model can be extended into a multi-scale setting, working flexibly under scale variations. As such, compared to the existing methods that rely on a flat pixel-based model, DSP achieves substantial gains in both matching accuracy and runtime for exemplar-based semantic segmentation tasks.

The immediate impact of DSP may arise from its fast run-time. As mentioned, researchers are exploring many new vision and graphics applications that need dense pixel correspondences, such as semantic image segmentation [62], image/video completion [6], and non-parametric video depth estimation [45]. These applications are all based on the "pixel transfer" idea—transferring the information (e.g., class label, color, depth) of a pixel into its corresponding pixel in a data-driven, exemplar-based way. Fast DSP matching can significantly enhance the scalability of these exemplar-based applications.

Through the last two chapters, I introduced two different region-based matching methods, segmentation-driven matching (SDM) in Chapter 5 and DSP in this chapter. Here, I contrast trade-offs between two approaches in order to provide some insights on when SDM will be useful and when DSP is so.

SDM exploits a grouping cue of each region to impose a geometric constraint for matching features. The grouping cue is considered independently for each region, which permits each region to move flexibly when matched. This allows a greater advantage when matching deformable objects or objects with different spatial layout. In Sec. 5.3.1, I showed SDM can address many-to-many and many-to-one matches where objects have varying spatial layout between images. On the other hand, DSP aims to impose stronger geometric regularization by considering the connection across the different regions—regions are encouraged to move

together when matched. This helps disambiguate confusing appearance, improving the robustness to appearance variations and background clutter. In this sense, DSP is particularly useful when matching images that share similar geometric layout but exhibit confusing appearances, like scene images as shown in Sec. 6.3.

Regarding computational cost, DSP is noticeably faster mostly because it discards connections between nearby pixels in its graphical model, from which we can obtain a fast closed-form solution for matching. However, lack of neighbor connections sometimes leads to losing geometric consistency of pixel-level matches—some pixels might mix up when matched (e.g., losing their original spatial ordering in the matching locations). For label transfer task introduced in this chapter, this phenomenon would not matter much, since it focuses on finding matches in terms of class labels, while caring less about very accurate geometric consistency. However, for some graphics applications that require visually smooth and consistent results, SDM would be more desirable since it maintains neighbor linkage via its string representation.

Chapter 7

Conclusion

In this thesis, I explored shape-based visual perception from low-level local shape detection, to mid-level object segmentation, to high-level region-based matching. Not only does each component itself forms a basic building block for vision research, but I also developed an exemplar-based approach to integrate all those components for the ultimate goal of region-based object recognition. Extensive evaluations on challenging benchmark datasets validate the effectiveness of my approach.

Toward region-based visual recognition, I first addressed region detection in both local- and object-level. To capture objects' local shapes, I proposed a Boundary Preserving Local Region (BPLR) detector that integrates multiple segmentation hypotheses to respect object boundaries. By combining shape cues from multiple segmentation hypotheses, the BPLR detector respects object boundaries, robustly capturing object local shape under image variations.

Building on the strength of BPLR detectors, I developed an approach for object-level segmentation. The key insight of my approach is that shapes are often shared between objects of different categories. This shape sharing phenomenon makes it possible to transfer object shapes of one class to those of possibly unknown classes, providing category-independent shape priors. I devised a partial shape match method via BPLRs to retrieve those shape priors. As such, unlike previous top-down segmentation methods, my approach can enhance the segmentation of previously unseen objects.

Having established shape-based region detections, I moved on to region matching, in which I bring together the above ideas for local- and object-level region detection in a novel

approach to efficient image matching. To this end, I proposed a segmentation-driven local feature matching strategy, in which segmented regions in the image control spatial layout of the matched features. To encode such spatial structures, I devised a 1D image representation whose 1D nature enables efficient computation for matching. Putting both region detection and matching together, I achieved an exemplar-based region matching method for object recognition.

To achieve scalable exemplar-based recognition, I explored a fast image matching. The key idea is to exploit regions' hierarchical structure, from which I built a deformable spatial pyramid graphical model that considers various spatial extents for matching—from an entire image, to grid cells, to every single pixel. My pyramid model is well-suited for fast hierarchical optimization. At the same time, various spatial supports by my spatial pyramid improve the matching quality, striking a balance between geometric regularization and accurate localization. As a result, compared to today's most popular and powerful matching methods, my approach obtained substantial gains in both matching speed and accuracy.

Throughout, I addressed key issues of region-based recognition, from region detection to region-based matching to their applications for visual recognition. Extensive evaluations on challenging benchmark datasets validate the effectiveness of the proposed methods. Various applications demonstrates the promising potential of region-based visual recognition. To facilitate the future research, I publicly release all my codes for the works in my thesis.

Chapter 8

Future Work

For future work, I suggest some long-term research goals for region-based visual recognition.

First, it would be very promising to find new problem domains that our region-based methods can effectively work. One interesting direction would be to infer geometric relations across different objects from their segmented shapes as well as pixel correspondences. The goal is to leverage these low-level geometric information to build a high-level object recognition system that is robust to geometric variations. In contrast to existing applications that rely on ad-hoc methods to achieve geometric invariance like a multi-view component model, the idea would be to estimate general patterns of geometric deformations from the shape and pixel correspondences: e.g., we can learn a parametric model for pose and viewpoint variations of an object class from seeing how a shape and its inner pixels in one object move into another object.

Second, it is interesting to integrate both image segmentation and dense pixel matching in a unified framework. My current region-based approach runs in a feed-forward way, where segmented regions are first detected and they subsequently guide the matching. In contrast, I wonder if we could combine both in a single objective function so as for both segmentation and matching to intertwine to improve each other. At a glance, co-segmentation approaches seem somewhat relevant in that they also segment multiple images together. To my best knowledge, however, dense correspondences have not been taken into account for co-segmentation. Dense correspondences will add richer information for segmentation: for example, pixel correspondences can help compute multi-level segmentations by relating different levels of regions between images. Thus, we can detect not only foreground objects as in co-segmentation but also

corresponding object parts across different images. This would be very useful for discovering representative object parts that are common across the class.

Third, one could extend my approaches into video. For example, my shape sharing idea can be generalized into "space-time" sharing. By space-time shape sharing, I mean objects will not only share their shapes in a static form, but also exhibit similar types of shape deformations along the time sequence (e.g., animals run or walk in a similar fashion; people's action categories share many element motions). Based on this intuition, one would build a method to detect object-like space-time volumes in videos. This idea of space-time shape sharing would expand our previous object discovery work in [55], in that it could be applied to both object and action categories.

Fourth, I am interested in expanding the toolkits for the region-based recognition. Effective descriptors for representing BPLR and object segments are one very useful extension. Particularly, a compact representation that encodes both region's appearance and shape would be promising. The other useful extension would be a learning-based region detection that picks the very object-like segments, while discarding noisy regions. We can train such a detector on the regions introduced in the thesis—BPLR and shape sharing.

Last but not least, it is of practical importance to develop computationally efficient methods for region detection and object segmentation. For example, state-of-the-art bottom-up and/or object-level segmentation methods take several minutes for segmenting images, which is a bottleneck for scalable recognition. Particularly, I am interested in efficient segmentation in video, where one could exploit motion cue to prune out many unnecessary regions (e.g., areas without motion) for object hypotheses that otherwise would require expensive computation to process.

Bibliography

[1] B. Alexe, T. Deselaers, and V. Ferrari. Classcut for Unsupervised Class Segmentation. In *European Conference on Computer Vision (ECCV)*, 2010.

[2] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering Shared Structures in Multiclass Classification. In *International Conference on Machine Learning (ICML)*, 2007.

[3] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(5), 2011.

[4] P. Arbelaez, M. Marie, C. Fowlkes, and J. Malik. From Contours to Regions: An Empirical Evaluation. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[5] S. Baker, R. Szeliski, and P. Anandan. A Layered Approach to Stereo Reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, 1998.

[6] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein. The Generalized PatchMatch Correspondence Algorithm. In *European Conference on Computer Vision (ECCV)*, 2010.

[7] E. Bart and S. Ullman. Cross-Generalization: Learning Novel Classes from a Single Example by Feature Replacement. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[8] A. Berg. *Shape Matching and Object Recognition*. PhD thesis, Compute Science Division, Berkeley, 2005.

[9] A. Berg, T. Berg, and J. Malik. Shape Matching and Object Recognition Low Distortion Correspondences. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[10] I. Biederman. Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review*, 44(2):115–147, 1987.

[11] O. Boiman, E. Shechtman, and M. Irani. In Defense of Nearest-Neighbor Based Image Classification. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[12] E. Borenstein and S. Ullman. Class-Specific, Top-Down Segmentation. In *European Conference on Computer Vision (ECCV)*, 2002.

[13] A. Bosch, A. Zisserman, and X. Munoz. Image Classification using Random Forests and Ferns. In *International Conference on Computer Vision (ICCV)*, 2007.

[14] A. Bosch, A. Zisserman, and X.Munoz. Representing Shape with a Spatial Pyramid Kernel. In *Content based Image and Video Retrieval* , 2007.

[15] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(11), 2001.

[16] W. Brendel and S. Todorovic. Segmentation as Maximum Weight Independent Set. In *Neural Information Processing Systems (NIPS)*, 2010.

[17] T. Brox, L. Bourdev, S. Maji, and J. Malik. Object Segmentation by Alignment of Poselet Activations to Image Contours. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[18] L. Cao and L. Fei-Fei. Spatially Coherent Latent Topic Model for Concurrent Segmentation and Classification of Objects and Scenes. In *International Conference on Computer Vision (ICCV)*, 2007.

[19] J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[20] T. Chan and W. Zhu. Level Set Based Shape Prior Segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[21] H. Chen, Y. Lin, and B. Chen. Robust Feature Matching with Alternate Hough and Inverted Hough Transforms. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[22] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[23] O. Duchenne, A. Joulin, and J. Ponce. A Graph-matching Kernel for Object Categorization. In *International Conference on Computer Vision (ICCV)*, 2011.

[24] I. Endres and D. Hoiem. Category Independent Object Proposals. In *European Conference on Computer Vision (ECCV)*, 2010.

[25] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 2010.

[26] L. Fei-Fei, R. Fergus, and P. Perona. A Bayesian Approach to Unsupervised One-Shot Learning of Object Categories. In *International Conference on Computer Vision (ICCV)*, 2003.

[27] P. Felzenswalb, D. McAllester, and D. Ramanan. A Discriminatively Trained Multiscale Deformable Part Model. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[28] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2), 2004.

[29] P. Felzenszwalb and D. Huttenlocher. Efficient Belief Propagation for Early Vision. *International Journal of Computer Vision*, 70(1), 2006.

[30] V. Ferrari, F. Jurie, and C. Schmid. From Images to Shape Models for Object Detection. *International Journal of Computer Vision*, 87(3), 2010.

[31] V. Ferrari, T. Tuytelaars, and L. Gool. Object Detection by Contour Segment Networks. In *European Conference on Computer Vision (ECCV)*, 2006.

[32] V. Ferrari, T. Tuytelaars, and L. Gool. Simultaneous Object Recognition and Segmentation from Single or Multiple Model Views. *International Journal of Computer Vision*, 67(2):159–188, Apr. 2006.

[33] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning Globally-Consistent Local Distance Functions for Shape-Based Image Retrieval and Classificaiton. In *International Conference on Computer Vision (ICCV)*, 2007.

[34] C. Galleguillos, B. Babenko, A. Rabinovich, and S. Belongie. Weakly Supervised Object Localization with Stable Segmentations. In *European Conference on Computer Vision (ECCV)*, 2008.

[35] P. Gehler and S. Nowozin. On Feature Combination for Multiclass Object Classification. In *International Conference on Computer Vision (ICCV)*, 2009.

[36] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *International Conference on Computer Vision (ICCV)*, 2005.

[37] G. Griffin, A. Holub, and P. Perona. Caltech-256 Object Category Dataset. Technical report, CalTech, 2007.

[38] C. Gu, J. Lim, P. Arbelaez, and J. Malik. Recognition Using Regions. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[39] T. Hassner, V. Mayzels, and L. Zelnik-Manor. On SIFTs and Their Scales. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[40] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *International Conference on Computer Vision (ICCV)*, 2005.

[41] D. Hoiem, A. Stein, A. Efros, and M. Hebert. Recovering Occlusion Boundaries from a Single Image. In *International Conference on Computer Vision (ICCV)*, 2007.

[42] A. Ion, J. Carreira, and C. Sminchisescu. Image Segmentation by Figure-Ground Composition into Maximal Cliques. In *International Conference on Computer Vision (ICCV)*, 2011.

[43] F. Jurie and C. Schmid. Scale-Invariant Shape Features for Recognition of Object Categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2004.

[44] J. Kannala, E. Rahtu, B. Brandt, and J. Heikkila. Object Recognition and Segmentation by Non-Rigid Quasi-Dense Matching. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[45] K. Karsch, C. Liu, and S. Kang. Depth Extraction from Video Using Non-parametric Sampling. In *European Conference on Computer Vision (ECCV)*, 2012.

[46] J. Kim and K. Grauman. Asymmetric Region-to-Image Matching for Comparing Images with Generic Object Categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[47] J. Kim and K. Grauman. Boundary Preserving Dense Local Regions. In *Computer Vision and Pattern Recognition (CVPR)*, 2011.

[48] J. Kim and K. Grauman. Shape Sharing for Object Segmentation. In *European Conference on Computer Vision (ECCV)*, 2012.

[49] J. Kim, C. Liu, F. Sha, and K. Grauman. Deformable Spatial Pyramid Matching for Fast Dense Correspondences. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[50] P. Koniusz and K. Mikolajczyk. Segmentation Based Interest Points and Evaluation of Unsupervised Image Segmentation Methods. In *British Machine Vision Conference (BMVC)*, 2009.

[51] A. Kumar and C. Sminchisescu. Support Kernel Machines for Object Recognition. In *International Conference on Computer Vision (ICCV)*, 2007.

[52] M. Kumar, P. Torr, and A. Zisserman. OBJ CUT. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[53] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[54] Y. Lee and K. Grauman. Foreground Focus: Unsupervised Learning from Partially Matching Images. *International Journal of Computer Vision*, 85(2), May 2009.

[55] Y. Lee, J. Kim, and K. Grauman. Key-Segments for Video Object Segmentation. In *International Conference on Computer Vision (ICCV)*, 2011.

[56] V. Lempitsky, A. Blake, and C. Rother. Image Segmentation by Branch-and-Mincut. In *European Conference on Computer Vision (ECCV)*, 2008.

[57] M. Leordeanu and M. Hebert. A Spectral Technique for Correspondence Problems using Pairwise Constraints. In *International Conference on Computer Vision (ICCV)*, 2005.

[58] A. Levin and Y. Weiss. Learning to Combine Bottom-Up and Top-Down Segmentation. In *European Conference on Computer Vision (ECCV)*, 2006.

[59] A. Levinshtein, C. Sminchisescu, and S. Dickinson. Multiscale Symmetric Part Detection and Grouping. In *International Conference on Computer Vision (ICCV)*, 2009.

[60] Y. Lin, T. Liu, and C. Fuh. Local Ensemble Kernel Learning for Object Category Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[61] C. Liu, J. Yuen, and A. Torralba. Nonparametric Scene Parsing via Label Transfer. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(12), 2011.

[62] C. Liu, J. Yuen, and A. Torralba. SIFT Flow: Dense Correspondence across Different Scenes and Its Applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(5), 2011.

[63] C. Liu, J. Yuen, A. Torralba, J. Sivic, and W. Freeman. Sift Flow: Dense Correspondences across Different Scenes. In *European Conference on Computer Vision (ECCV)*, 2008.

[64] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2), 2004.

[65] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative Sparse Image Models for Class-Specific Edge Detection and Image Interpretation. In *European Conference on Computer Vision (ECCV)*, 2008.

[66] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using Contours to Detect and Localize Junctions in Natural Images. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[67] T. Malisiewicz and A. Efros. Improving Spatial Support for Objects via Multiple Segmentations. In *British Machine Vision Conference (BMVC)*, 2007.

[68] J. Matas, O. Chum, M. Urba, and T. Pajdla. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *British Machine Vision Conference (BMVC)*, 2002.

[69] K. Mikolajczyk and C. Schmid. Scale and Affine Invariant Interest Point Detectors. *International Journal of Computer Vision*, 1(60):63–86, October 2004.

[70] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A Comparison of Affine Region Detectors. *International Journal of Computer Vision*, 65:43–72, 2005.

[71] M. Muja and D. Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Applications*, 2009.

[72] E. Nowak, F. Jurie, and B. Triggs. Sampling Strategies for Bag-of-Features Image Classification. In *European Conference on Computer Vision (ECCV)*, 2006.

[73] A. Opelt, A. Pinz, and A. Zisserman. Incremental Learning of Object Detectors Using a Visual Shape Alphabet. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[74] M. Pandey and S. Lazebnik. Scene Recognition and Weakly Supervised Object Localization with Deformable Part-Based Models. In *International Conference on Computer Vision (ICCV)*, 2011.

[75] C. Pantofaru, G. Dorko, C. Schmid, and M. Hebert. Combining Regions and Patches for Object Class Localization. In *Beyond Patches, Workshop in conjunction with CVPR*, 2006.

[76] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in Quantization: Improving Particular Object Retrieval in Large Scale Image Databases. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[77] T. Quack, V. Ferrari, B. Leibe, and L. Gool. Efficient Mining of Frequent and Distinctive Feature Configurations. In *International Conference on Computer Vision (ICCV)*, 2007.

[78] A. Quattoni, M. Collins, and T. Darrell. Transfer Learning for Image Classification with Sparse Prototype Representations. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[79] X. Ren, C. Fowlkes, and J. Malik. Figure/Ground Assignment in Natural Images. In *European Conference on Computer Vision (ECCV)*, 2006.

[80] X. Ren and J. Malik. Learning a Classification Model for Segmentation. In *International Conference on Computer Vision (ICCV)*, 2003.

[81] C. Rother, V. Komogorov, and A. Blake. Grabcut: Interactive Foreground Extraction Using Iterated Graph Cuts. In *SIGGRAPH*, 2004.

[82] M. Sabuncu and P. Ramadge. Using Spanning Graphs for Efficient Image Registration. *IEEE Trans. on Image Processing*, 17, 2008.

[83] D. Scharstein and R. Szeliski. A Taxonomy and Evaluation of Dense Two-frame Stereo Correspondence Algorithms. *International Journal of Computer Vision*, 47, 2002.

[84] T. Sebastian, P. Klein, and B. Kimia. Recognition of Shapes by Editing their Shock Graphs. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26:551–571, 2004.

[85] H. Shum and R. Szeliksi. Construction and Refinement of Panoramic Mosaics with Global and Local Alignment. In *International Conference on Computer Vision (ICCV)*, 1998.

[86] P. Srinivasan, Q. Zhu, and J. Shi. Many-to-one Contour Matching for Describing and Discriminating Object Shape. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[87] M. Stark, M. Goesele, and B. Schiele. A Shape-Based Object Class Model for Knowledge Transfer. In *International Conference on Computer Vision (ICCV)*, 2009.

[88] S. Todorovic and N. Ahuja. Extracting Subimages of an Unknown Category from a Set of Images. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.

[89] S. Todorovic and N. Ahuja. Learning Subcategory Relevances for Category Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[90] S. Todorovic and N. Ahuja. Region-Based Hierarchical Image Matching. *International Journal of Computer Vision*, 78(1):47–66, Jan. 2008.

[91] A. Torralba, K. Murphy, and W. Freeman. Sharing Visual Features for Multiclass and Multiview Object Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(5), 2007.

[92] T. Tuytelaars. Dense Interest Points. In *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[93] M. Varma and D. Ray. Learning the Discriminative Power-Invariance Trade-Off. In *International Conference on Computer Vision (ICCV)*, 2007.

[94] VLFeat Open Source Library. *http://www.vlfeat.org/*.

[95] N. Vu and B. Manjunath. Shape Prior Segmentation of Multiple Objects with Graph Cuts. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[96] Z. Wu, Q. Ke, M. Isard, and J. Sun. Bundling Features for Large Scale Partial-Duplicate Web Image Search. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[97] D. Xu, T. Cham, S. Yan, and S. Chang. Near Duplicate Image Identification with Spatially Aligned Pyramid Matching. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[98] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered Object Models for Image Segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2001.

[99] H. Zhang, A. Berg, M. Marie, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classfication for Visual Category Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2006.