Superparsing with Improved Segmentation Boundaries through Nonparametric Context

by

Hong Pan

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.C.S degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Declaration

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of University of Ottawa's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

Abstract

Scene parsing, or segmenting all the objects in an image and identifying their categories, is one of the core problems of computer vision. In order to achieve an object-level semantic segmentation, we build upon the recent superparsing approach by Tighe and Lazebnik, which is a nonparametric solution to the image labeling problem.

Superparsing consists of four steps. For a new query image, the most similar images from the training dataset of labeled images is retrieved based on global features. In the second step, the query image is segmented into superpxiels and 20 different local features are computed for each superpixel. We propose to use the SLICO segmentation method to allow control of the size, shape and compactness of the superpixels because SLICO is able to produce accurate boundaries. After all superpixel features have been extracted, feature-based matching of superpixels is performed to find the nearest-neighbour superpixels in the retrieval set for each query superpxiel. Based on the neighbouring superpixels a likelihood score for each class is calculated. Finally, we formulate a Conditional Random Field (CRF) using the likelihoods and a pairwise cost both computed from nonparametric estimation to optimize the labeling of the image. Specifically, we define a novel pairwise cost to provide stronger semantic contextual constraints by incorporating the similarity of adjacent superpixels depending on local features. The optimized labeling obtained with the CRF results in superpixels with the same labels grouped together to generate segmentation results which also identify the categories of objects in an image.

We evaluate our improvements to the superparsing approach using segmentation evaluation measures as well as the per-pixel rate and average per-class rate in a labeling evaluation. We demonstrate the success of our modified approach on the SIFT Flow dataset, and compare our results with the basic superparsing methods proposed by Tighe and Lazebnik.

Acknowledgements

I would like to express my utmost gratitude to my supervisor, Dr. Jochen Lang for all the help and support he provided in my research and writing. He has always been extremely patient and expertise to guide me with a very useful insight which helped solve the problems and improve the quality of my work.

Additionally, I would like to thank Ziyun Li and Meng Zhou for their helpful advices.

Lastly but most importantly, I thank my family for their ongoing support throughout my education, specially my boyfriend Yichen Tang for his indispensable help and encouragement in the writing of this thesis.

This research was funded in part by Graphics, Animation and New Media Canada (GRAND).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

We have developed an image parsing system to produce object-level segmentations for an haptic image exploration system which also addresses the image labeling problems at the same time. Our work is motivated by the research done on haptic image exploration by Lareau and Lang [31].

As introduced in [31], the haptic image exploration system enables the exploration of digital photographs using haptic technologies in which users explore an image only with the sense of touch. In this haptic exploration system, the image content is transferred from the visual to the haptic sense based on an interactive segmentation at multiple levels-of-detail (LOD). The levels-of-detail form a strict object-level segmentation hierarchy. This design offers new opportunities in situations where our visual sense is occupied or for users with vision-loss. However, the interactive authoring of a segmentation hierarchy for an image becomes a major bottleneck in the haptic system. If the hierarchal segmentation could be done automatically, haptic exploration of images could be done without an author, e.g., while browing the web [55]. Unfortunately, automatic hierarchical segmentation at an object-level is currently not possible at high quality.

The big challenge in automatic hierarchical segmentation is the coarse-level segmen-

tation, i.e., segmentation at an object-level. Usually humans perceive the scene in an image at a first glance based on its general structure rather than the details, but typical automatic segmentation algorithms are not able to recognize objects in an image at coarse levels. Segments generated by automatic segmentation algorithms are often unable to describe meaningful high-level objects. Thus, we like to use object recognition to help decide which superpixel segments correspond to objects that should appear at a coarse level.

Object recognition is challenging but partial solutions have appeared recently as image parsing approaches [35, 63]. Image parsing approaches can be taken pixel by pixel, over segmentation regions or bounding boxes. Specifically, we follow the superparsing approach by Tighe and Lazebnik [63] to segment all the objects in an image into superpixels and identify their categories at first, and then we can just simply merge these small adjacent regions belonging to the same category into bigger regions based on their (common) labels. Image parsing approaches work well in the accurate delineation of objects at coarse levels, which also may be a possible solution to automatic segmentation at an object-level.

In our work, we adapt and implement the superparsing system of Tighe and Lazebnik but more importantly, we propose an improvement for the contextual inference based on conditional random fields (CRF) to achieve better labeling and segmentation results.

1.2 Thesis Statement

It is possible to obtain object-level segmentations of images by solving the image labeling problem. The superparsing approach can meet this challenge by labeling superpixles produced by state-of-the-art segmentation methods and merging adjacent regions that have been assigned the same label. With the enhancement of contextual inference, some improper assignment in the initial labeling are corrected to improve the labeling results and achieve more accurate boundaries for object-level segmentations.

1.3 Overview

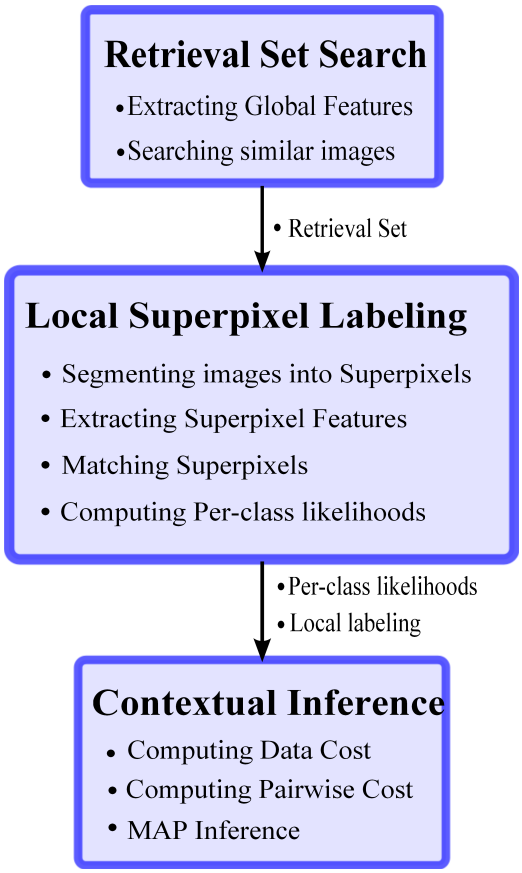

Figure 1.1: High-level overview of general image parsing systems based on random fields.

In Figure 1.1 we give a high level overview of our proposed system in the form of a flow-chart depicting the process that we follow. To solve the image labeling problem, there is a retrieval set search stage, a local labeling stage, and a contextual inference stage. Next we describe the details of the basic superparsing system according to Tighe and Lazebniks work at first.

The first stage computes global features of each image, and then finds similar images to the query image to form a relatively small retrieval set of training images depending on the similarity of their global features. This is beneficial for computational efficiency

as well as providing scene-level context for post-processing.

The second stage comprises several steps, which produce the local superpixel labeling and prepare for the contextual inference. The first step is to segment images into superpixels using an automatic bottom-up graph-based segmentation method [12]. Then Tighe and Laznebik suggest to compute 20 different local features to describe the appearance of superpxiels in the second step. After all superpixel features are extracted, a feature-based superpixel match step is performed as a trained classifier to find the nearest-neighbour superpixels in the retrieval set for each superpixel of the query image. Finally, given the ground truth labels, a per-class likelihood ratio score for each superpixel is calculated based on the superpixel matches. Now one can obtain a local labeling of the query image depending on the per-class likelihoods (i.e., assigning to each superpxiel the class that maximizes the per-class likelihood).

The purpose of the semantic contextual inference stage is to optimize the local labeling produced by the second stage as well as achieve better segmentation results. Tighe and Lazebnik formulate the global image labeling problem as a Markov Random Field (MRF) to incorporate the contextual information. A MRF energy function is defined in form of the sum of a data cost and a pairwise cost (or smoothing cost), where an adjacency graph is defined over the superpixels of the image and the pairwise cost is used to smooth the initial labeling. The data cost is computed using the likelihood ratio scores obtained in the second stage, while the pairwise cost is defined based on probabilities of label co-occurrence. Finally, a maximum a posteriori (MAP) inference on the MRF using the efficient graph-cut optimization algorithm [7] is performed.

Tihge and Lazebnik perform a simple and effective nonparametric approach to the problem of image parsing. But in terms of image segmentation, we expect results with more accurate boundaries. In order to do this, we make some modifications and extension to their current superparsing system.

At first in the retrieval set search, instead of directly computing and ranking the dis-

tance between the query and training images every time, we choose to perform FLANN, a nearest neighbour searching algorithm to speed up the process. Then in the second stage, since superpixels produced by the graph-based segmentation method are highly irregular, SLICO [1] is applied to control of the size, shape and compactness of superpixels. The current semantic contextual inference is not powerful enough only based on the properties of label co-occurrence. Thus in the last stage, we propose a novel Conditional Random Field (CRF) inference that incorporates more powerful neighbourhood context for labeling optimization. Specifically, a new pairwise cost is defined to integrate the similarity of neighbouring superpxiels depending on local features into the CRF inference.

A simple example of the different labeling results based on the graph-based segmentation algorithm is shown in Figure 1.2.

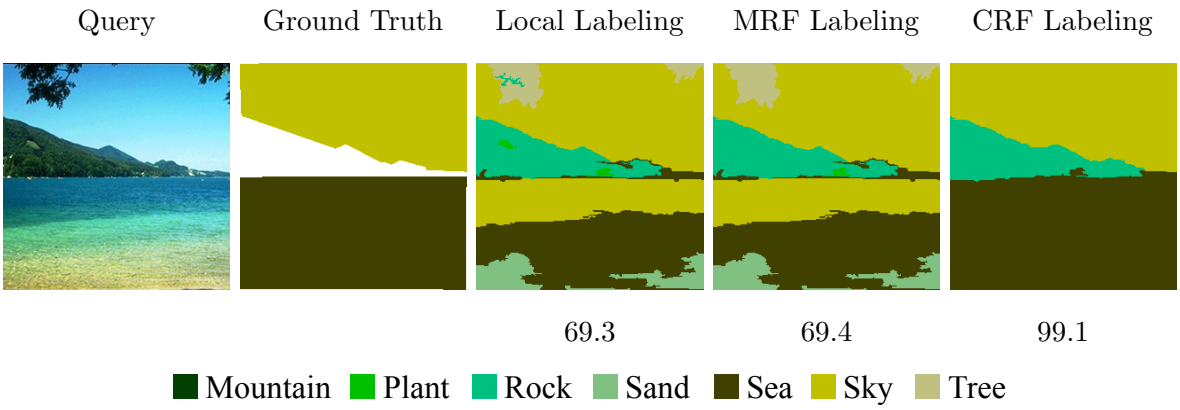

Figure 1.2: (©2015 IEEE) Example labeling results of superparsing system incorporating semantic context based on graph-based segmentation. The number under the image is the percentage of correctly labeled pixels. Note the empty area in the ground truth is non-labeled.

1.4 Contributions

The following outlines the contributions of our work:

- Use of the FLANN algorithm to speed up the search of the retrieval set based on different global features.

- Change of the segmentation method to produce more superpixels with more accurate boundaries but smaller regions using the SLICO segmentation.
- Definition of a novel pairwise cost in the CRF inference to optimize and smooth the local labeling results.

To achieve the improvement above, we have adapted and implemented the basic superparsing system of Tighe and Lazebnik using C++ with the help of the OpenCV library.

1.5 Thesis Organization

The rest of this thesis is organized as follows.

- **Chapter 2** reviews the related work we introduced above in detail, providing a thorough background on image parsing, and overviews of automatic segmentation methods, superpixel algorithms, feature extraction algorithms and contextual inference based on random fields. A brief overview of some segmentation evaluation measures is also covered in this chapter.
- **Chapter 3** introduces the basic superparsing system of Tighe and Lazibnik, and presents the details of our implementation and modifications.
- **Chapter 4** extends the superparsing approach using SLICO segmentation and our novel contextual inference to improve the segmentation and labeling results.
- **Chapter 5** provides more results, compares the proposed method to the basic superparsing approach from Tighe and Lazebnik. We examine the overall performance of our modified superparsing system and provide some discussions.
- **Chapter 6** gives our concluding remarks including limitations and future works for our system.

Chapter 2

Related Work

Our work has focused on how to obtain object-level segmentations automatically, so in this chapter we provide an overview of the automatic image segmentation and image parsing literature, including the necessary techniques for superpixel preparation, feature extraction and labeling optimization in the image parsing approach. Section 2.1 introduces some typical automatic segmentation algorithms. We discuss several image parsing approaches for our object-level segmentation purpose in Section 2.2. Then Section 2.3 presents the superpxiel methods used in our implemented superparsing system, and some other options are included as well. The global and local features used to describe images and superpxiels are introduced in Section 2.4. And in Section 2.5, we explain how to optimize the labeling and segmentation results using contextual information based random fields. Several possible MRF and CRF methods are also compared here. A summary of this chapter is given in Section 2.6

2.1 Automatic Segmentation

For the automatic hierarchical segmentation at an object-level, we first consider segmenting images using existing automatic segmentation algorithms. There are numerous algorithms for automatically segmenting images. Here we provide a brief review of some widely used methods: Mean Shift algorithm by Comaniciu and Meer [9], Statistical Re-

gion Merging of Nock and Nielsen [47], and Ultrametric Contour Map of Arbelaez [3]. For a more comprehensive review of image segmentation algorithms the reader is encouraged to look at surveys by Freixenet et al. [14] and Haralick et al. [20], and a paper on automatic image segmentation by Arbelaez et al. [2].

Mean-shift

The Mean-shift image filter proposed by Comaniciu and Meer [9] is a mode-seeking filter that assigns each pixel of an image to the closet mode of its local density distribution in the spatial and color domain. The closet mode is found by iteratively moving to the kernel soothed centroid for every pixel, and the method is guaranteed to converge. The Mean-shift filter can be applied to achieve image segmentation in two steps. In the first step, the image is smoothed with the Mean-shift filter, preserving the discontinuities. Then the same segment label is assigned to pixels belonging to nearby modes in both space and color. The generated segments can be large or small based on the input kernel parameters, an optional step can be used to eliminate segments smaller than a given size. Mean-shift can also be used to get superpxiels, which we will introduce in Section 2.3.

Statistical Region Merging

Nock and Nielsen describes a color image segmentation algorithm, Statistical Region Merging (SRM), based on region growing and merging [47]. The method models segmentation as an inference problem, in which the segmentation can be achieved with low errors. This algorithm builds an adjacency graph of an image, where each individual pixel is considered as a segment and every segment is connected to its spatial neighbours by weighted edges. The weights are computed depending on the color difference between the adjacent segments. Then a simple merging order is derived by sorting these edge weights, and it does not change after pixels have been merged to regions. Nock and Nielsen define a merging predicate to provide a quantitative bound on the segmentation error. Once the merge order is obtained, a single pass through the edges is performed, in which two segments are merged if the merging predicate is satisfied. Because the produced segments often tend to be large regions and many small regions, it usually extends to merge areas less than a prescribed threshold with their nearest neighbours in

a post-processing step.

Since the merging order is fixed, it only needs to be computed once from the original adjacent graph with high computation efficiency. The SRM method also gives simplicity to directly control the scale of segmentation as well as good performance.

Ultrametric Contour Map

Arbelaez el al. describes the Ultrametric Contour Map (UCM) algorithm to create hierarchical image segmentation based on the contours of images in [3]. UCM defines a duality between closed, non-self-intersecting weighted contours and a hierarchy of regions, which allows all regions in a single level of the hierarchy being disjoint. As moving towards to the base level of the hierarchy, it considers even weak contours corresponding to an over-segmentation of the image. Upper levels respect only strong contours which results in under-segmentation.

This segmentation starts with constructing a set of initial regions using Oriented Watershed Transform (OWT) [3]. The initial regions resulting from an over-segmentation are considerate to be the finest partition in the hierarchy. OWT is applied to approximate closed contours of these regions by detecting the dividing lines between homogenous areas. The weights of the contours are also computed depending on boundary strength in the Watershed transform. Then, the UCM algorithm transforms the weighted contours into hierarchy of regions. The hierarchy is contracted by a greedy graph-based region merging algorithm, in which the most similar regions are iteratively merged. Specifically, the similarity between regions are indicated by the weight of their contour, and the two adjacent regions separated by the weakest contour are considered as the most similar ones to be merged. The merging of similar regions is created according to a contour threshold. Increasing this threshold removes contours and merges regions that used to be separated.

Having the segmentation hierarchy, we can find the best segmentation with the help of other knowledge about the scene. However, it is not adequate to produce the object-

level segmentation hierarchy we expect in Chapter 1.

There are also some other popular superpixel methods that can be used to achieve automatic segmentation, such as graph-based segmentation algorithm [12] and normalized cut algorithm [58]. We will review the stat-of-the-art superpixel methods in Section 2.3.

2.2 Image Parsing

In Chapter 1, we discussed the inadequacy of automatic segmentation for producing object-level segmentations. Here, we explore image parsing approaches to segment all the objects in an image and identify their categories. As discussed in [63], many approaches to the image parsing problem haven been proposed recently to associate object category labels with pixels [30, 60, 59], regions [18, 51, 39] or object bounding boxes [24, 56] (see Section 2.5 for more details). Most of these approaches work with a fixed number of predefined classes and require training generative or discriminative models for each class in advance.If new categories are added to the dataset, the tedious training has to be repeated for new models of these categories, and in learning-based parametric systems, parameters need to be adjusted. Moreover, when the contextual information is incorporated, the contextual relationships among objects will also increase rapidly as the number of categories expands. Even processing a test image is quite slow in most cases due to the use of multiple object detectors.

Recently, instead of training complex parametric models, nonparametric data-driven approaches have shown potentials to solve the image parsing problem more efficiently based on image retrieval and matching [56, 22, 65, 35, 36, 63]. Usually for a new test image, they try to retrieve the most similar training images and transfer the desired information from the training images to the query, which greatly reduces the computational cost and makes it more flexible to parse the objects in the query image.

Next, we would like to review some examples of both types of object recognition

methods in greater depth.

2.2.1 Nonparametric Approaches

With the emergence of large databases, a new family of nonparametric methods have been proposed: large database-driven approaches. In the area of nonparametric methods, we can find several systems proposed recently.

Russell el al. [56] present a object recognition system by scene alignment on the LabelMe database [57]. In their system, an unknown input image is matched to a large training database of densely labeled images, and the most similar images in the retrieval set are used to build a label probability map for the input image. With this probability map, they achieve recognizing and localizing possible object categories that may appear in the input image.

Torralba et al. [65] build a database of 80 million tiny images collected from the Web. The images in the database are stored as $32 - by - 32$ color images and loosely labeled with different categories. They also show that, for certain classes like people, a simple sum of squared difference (SSD) match is able to give semantically meaningful parsing.

Liu et al. also propose a nonparametric label transfer method based on estimating "SIFT flow", which represents a dense deformation field between images. To parse an input image, they match the visual objects in the input image to the images in the training database. For a new query image, this system first retrieves its nearest neighbours from a large database consisting of labeled images. The images in the retrieval set share similar scene structure such as objects and relationships. Then dense correspondences between the query image and each of the nearest neighbours set are established using the dense SIFT flow algorithm [36]. Based on SIFT flow matching scores, the nearest neighbours are reranked and form a top M-voting candidate set. Finally, according to the estimated dense correspondences, they map the existing annotations from the voting candidates and integrate multiple cues in a Markov Random Field model to segment and

parse the query image. This system outperforms existing recognition-based approaches that require training classifiers or appearance models for each object category.

However, Tighe and Lazebnik point out some drawbacks of scene parsing system via label transfer in [63]. Finding the SIFT flow is fairly complex and expensive, and it may not be necessary to estimate a dense per-pixel flow in scene matching because scenes are collected by discrete objects with spatial support. Compared to their method, Tighe and Lazebnik propose a straightforward and efficient superparsing system that also makes use of a retrieval set of images with similar scenes. As introduced in [63], the superparsing system requires no training except some basic computation of dataset statistics. The details of this method along with our adaptation of it will be explained in Chapter 3.

2.2.2 Parametric Approaches

For the parametric approaches that consist of learning generative / discriminative models, we focus on a parametric approach specifically designed for the image segmentation purpose.

Hariharan et al. [21] propose Simultaneous Detection and Segmentation, SDS, for object detection and semantic segmentation. Unlike classical semantic segmentation, this method categories individual object instances, i.e., detects all instances of a category in an image and correctly masks the pixels belonging to each instance. The SDS algorithm consists of 4 steps. At first, the SDS starts with category-independent bottom-up object proposals to generate segmentations as well as bounding boxes. Multicale combinatoral grouping (MGG) [4] is used in this step to prepare 2000 region candidates per image. In the second step, a convolutional neural network (CNN) is applied to extract features on each region from both its bounding box and region foreground. Then they train a support vector machine (SVM) on the CNN features from the previous step to assign a score for each category to each candidate. The initial segmentation obtained from the region classification step, however, is prone to undershooting and overshooting due to the lack of specific shape information. Thus, a region refinement step is performed using non-maximum suppression (NMS) to refine the surviving candidates with the help of

category-specific coarse mask predictions.

Hariharan et al. evaluate SDS algorithm using a precision recall (PR) curve, and the average precision (AP) corresponding to region and bounding box respectively. On the aspect of individual object instance categorization, they show an improvement compared to state-of-the-art methods on semantic segmentation.

Section 2.5 provides more reviews of parametric methods for the image parsing problem as well as discussing how to incorporate contextual information into object recognition.

2.3 Superpixel Methods

Superpixels provide an internal over-segmentation of an image which produces segments that for example are used as the smallest units (or primitive) in the matching step of the superparsing approach [63]. It is also much more convenient to compute local image features based on the grouping of pixles. In this section, we start by reviewing three most commonly used superpixel methods: the graph-based segmentation algorithm of Felzenszwalb and Huttenlocher [12] and Simple Linear Iterative Clustering (SLIC) by Achanta et al. [1] and their variants. Then we briefly introduce and categorize the existing superpixel algorithms into graph-based and gradient-ascent-based following the surveys by Achanta et al. [1] and Stutz et al. [62]. There are also some image segmentation algorithms included in this classification, but we focus on their suitability for producing superpixels.

Graph-based Segmentation

Felzenszwalb and Huttenlocher [12] presented a greedy graph-based segmentation algorithm that is often used to create superpixels. This method aids in many computer vision tasks because of its good performance and computational efficiency (i.e., $O(N \log N)$

where N is the number of pixels).

Their algorithm inteprets the image as a graph and two ways are proposed to see the edges. The first form is simple in which the image grid is used to define the local neighbourhood between image pixels. Alternatively, they propose to use a nearest neighbour graph, where each pixel is mapped to a feature point (x, y, r, g, b) and edges connect points that are close together in this feature space. Both graphs work well, but the second form captures more spatially non-local properties of the image. Independent of the graph representation, each node is initially a separate region and each edge is assigned a weight. The weight of an edge measures the dissimilarity between its connected nodes and the measure can be as simple as the difference in grayscale intensity (or intensity difference). This method merges two regions if the minimum weight across the boundary is smaller than the internal difference of both components. The internal difference of a component $Int(C)$ is computed as the maximum edge weight within the minimum spanning tree of the component, but the internal difference of two components is defined as:

$$MInt(C_1, C_2) = min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)), \tag{2.1}$$

where the threshold function $\tau(C) = k/\left|C\right|$ depends on the size of C and a constant parameter k. A large component is preferred when k is large. The authors have released C++ source code.

Tighe and Lazebnik choose to use the graph-based segmentation in the original super parsing system due to its ability to produce a slight over-segmentation. But superpixels produced by this algorithm tend to be highly irregular in shape and size, which could be undesirable in some situations.

Simple Linear Iterative Clustering

Achanta et al. introduce gradient-based Simple Linear Iterative Clustering (SLIC) Superpixels [1], a modification of k-means clustering. This method works well to produce

high quality, compact, and nearly uniform superpixels as shown in Figure 4.2. Here clustering is performed in $5D$ $[labxy]$ space defined by the pixel color vector in CIELAB space and pixel position. The algorithm clusters the image in the spatial and color domains similar to Mean-shift [9]. Let K be the number of superpixels desired for an image with N pixels, the approximate size of each superpixel is N/K. Since the superpixels are uniform in size and shape, the center of every superpixels is roughly placed at every $S = \sqrt{N/K}$ pixels. For each cluster center, an area of $2S \times 2S$ around it is used to search for the pixels nearest to this cluster center. The CIELAB color space is used because it is considered to be uniform for small color distances, which also results in limited color distance. However, the spatial distance depends on the image size which could easily exceed the color distance limit and outweigh pixel color similarities in a simple Euclidean norm. To avoid this, Achanta et al. define a new distance measurement with the help of the Euclidean distance in each space as follow:

$$D = \sqrt{d_c^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{2.2}$$

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{2.3}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \tag{2.4}$$

where d_c is for the lab distance and d_s for the xy plane distance between pixles i and j. In this distance definition, S is expected to be the maximum spatial distance while the parameter m determines the maximum color distance and controls compactness of a superpixel. A larger m will emphasize more spatial proximity and encourage more compact clusters. Empirically, $m = 10$ is used to provide a good balance between color similarity and spatial proximity.

The algorithm begins with initializing K cluster centers placed regularly at every S grid intervals, and moves the cluster centers according to the pixels associated to them. Then an updated center is computed as the average $labxy$ vector of the pixels belonging to that center. This classification of pixels to clusters repeats until convergence. Since

connectivity is not enforced in this algorithm, an extra step is added to relabel disjoint segments with the labels of the largest neighbouring cluster.

Achanta et al. also give SLICO, the zero parameter version as a variant of the SLIC algorithm (i.e., the Adaptive-SLIC or ASLIC in [1]). The difference comes from the compactness parameter m. SLIC uses the same m set by the user for all superpixels in the image, which produces smooth regular-sized superpixels in the smooth regions and highly irregular superpixels in the textured regions. To avoid this problem and make it easy to choose the right compactness for each image, SLICO adapts the normalized distance measure as:

$$D = \sqrt{\left(\frac{d_c}{m_c}\right)^2 + \left(\frac{d_s}{m_s}\right)^2}. \tag{2.5}$$

Different from the constant values used in Equation 2.2 for SLIC, the proximities for each cluster in SLICO are normalized dynamically. After using constant normalization factors in the first iteration, the maximum observed spatial and color distance (m_s, m_c) for each cluster from the previous iteration are subsequently used for the normalization in the following process. By this improvement, SLICO is able to adaptively choose the compactness parameter for each superpixel and generate regular sized and shaped superpxiels in both textured and non-textured regions. Although SLICO is slightly slower than SLIC, it is still quite efficient.

As described above, both SLIC and SLICO algorithms have a $O(N)$ complexity in computational cost and memory usage, which is advantageous compared to the Normalized cuts algorithm [58]. Compared to graph-based segmentation, direct control of the number, size and compactness of the output superpixels is offered in SLICO by simply changing the corresponding parameter. For these reasons, we choose to use SLICO superpixels to extend the superparsing system in our work (see Chapter 4).

After introducing the two superpixel algorithms used in the superparsing system, we

briefly review more superpixel methods following the graph-based and gradient-ascent-based strategies, which gives some other options for obtaining superpixels.

Graph-based Superpixel Algorithms

In graph based algorithms, pixels are regarded as nodes in a graph, and edge weight between two nodes indicates their similarity. A cost function is defined on the graph and superpixels are generated by minimizing this cost function. SLIC and Normalized Cuts are graph-based methods. In addition to SLIC and Normalized cuts, more graph based methods are introduced below.

N-Cuts Shi and Malik describe a graph partitioning image segmentation algorithm, Normalized cuts [58]. In this graph partitioning method, all nodes (i.e. pixels) in the image are initially considered to be in the same segment, and all internal nodes in the segment are connected by weighted edges. An objective cost function is used to compute the weights based on the similarity between two nodes. Under these assumptions, the algorithm recursively partitions a given graph using contour and texture cues, thereby globally minimizing the internal dissimilarity of the partitions. N-Cuts has a complexity of $O(N^{\frac{3}{2}})$, where N is the number of pixels.

LS A lattice superpixel [44] is generated by detecting vertical or horizontal paths that cut the image. The algorithm initializes the generation with a grid and uses graph cuts to iteratively optimize the vertical and horizontal strips. While this algorithm has control over the size and number of superpixels, the speed is not very desirable due to the pre-computation of boundary maps.

CI and CIS Veksler et. al. [72] propose Compact Superpixels and Constant Intensity Superpixels. These two are graph-based methods defined on grayscale images. An image is initially covered with overlapping square patches of fixed size, which results in each pixel being coverd by several patches. Under their energy function definition, the patches are stitched so that the seam are encouraged to align intensity edges. Each

pixel can get assigned to one of the overlapping patches, each of which represents a superpixel with regulized boundaries. Compared to the previous work, the method has good computational efficiency and principled optimization.

ERS Liu et al. presents another graph-based method, Entropy Rate Superpixels, that defines a new objective function for superpixel segmentation in [37]. This objective function consists of a entropy rate of a random walk on a graph and a balancing term. The entropy rate encourages superpixels with homogeneous clusters, while the balancing term favours superpixles with similar size. Compared to the state-of-the-art methods, it is faster than CS and CIS, Turbopixles [33] and N-Cut, while it is slower than the graph-based segmentation method.

Gradient-ascent-based Superpixel Algorithms

Like SLIC superpxiels [1], gradient ascent methods start from initial rough clustering, and refine the clusters from the previous iteration to obtain better segmentation until convergence.

TP Turbopixels proposed by Levinshtein et al. segments an image into superpixels by dilating seeds in the image plane using geometric flow based on local image gradients [33]. Superpixels produced by this algorithm respect object boundaries with uniform size and compactness. The algorithm is very efficient in computation with approximate linear complexity in image size.

SEEDS Superpixels Extracted via Energy-Driven Sampling [69] proposed by Van den Bergh et al. is a new approach depending on a simple hill-climbing optimization. Based on an initial superpixel partitioning, the superpixels are refined iteratively by moving the boundaries of superpixels, i.e., exchanging pixels between neighbouring superpixels using color histograms.

We like to introduce two segmentation algorithms that can also generate slight over-

segmentation: Mean-shift [9] and Quick-shift [71]. Like Mean-shift which we described in Section 2.1, Quick-shift is also a mode-seeking segmentation. But there is no direct control over the number, size or compactness of the output superpixels in these two algorithms.

2.4 Feature Extraction

Numerous image features are used in the superparsing approach. In the retrieval set search step, global features are used to capture the similarities between images based on scene types, objects and spatial layout. After having the superpxiels of the query and its similar images using algorithms in Section 2.3, the appearance of superpixels is described with the help of local features associated with shape, location, color and texture. Now we give a general review of several algorithms to extract global and local features in the superparsing approach.

Color Histogram

In computer vision, images are generally represented as a series of pixel values, corresponding to their colors. A color histogram is a representation of the distribution of colors in an image. Based on the idea that similar images would contain similar proportion of certain colors [27], comparing color histogram is a common approach for color based image retrieval. This method is efficient and easy in processing content information, ignoring the size, orientation and position of a certain image. Hence, a main drawback of color is the inability to incorporate the spatial features of colors in an image.

The color histogram can be built in different color spaces, e.g. the most commonly used three-dimensional spaces are RGB and HSV. A RBG color histogram can be understood as a combination of three 1D histograms based on the R, G and B channels of the RGB color space. First, the color is split into R, G, B components where each color channel is used to create a histogram. Each histogram consists of 8 bins where each bin defines a fixed color range, containing similar color values. The value stored in each

bin represents the number of pixels in the image that are within that range. Finally, normalizing the values in each bin by dividing the total number of pixels in the image. Note that more bins and a smaller range may give better performance in distinguishing colors, however, at a higher computational cost.

GIST

The GIST descriptor, an abstract representation of the scene, has recently received increasing attention in the recognition of real world scenes. The methodology relies on developing a low dimensional representation of the scene without segmenting images or processing individual objects. A set of perceptual properties (naturalness, openness, roughness, ruggedness and expansion) are proposed as a spatial envelope of a scene to describe the dominant spatial structure of a scene [48]. In this way, scenes sharing a similar and stable spatial structure more likely belong to the same semantic category.

The GIST descriptor of an image is computed by first filtering the image using a filter bank of Gabor filters, and then averaging the responses of the filters in each block on a $4-by-4$ non-overlapping grid. The Gabor filter is a linear band-pass filter defined by a Gaussian function modulated with a complex sinusoid. Filters in a Gabor filter bank can be considered as a set of edge detectors with tuneable scale and orientation which derive the texture information from statistics of the outputs of those filters [54]. Note that only the respective mean of the filter responses are used in GIST (ignoring the standard deviation distribution of the filter responses).

Spatial Pyramid

Lazebnik et al. propose spatial pyramid matching (SPM) based on pyramids that describe the spatial layout of an image very well [32]. It is an extension of the bag-of-feature (BoF) model. In recent years BoF has been popular for content based image categorization. The basic idea of BoF is to quantize local invariant descriptors (e.g. SIFT) into a set of discrete visual words, and then compute a histogram of these visual words

to represent the image. While the algorithm is simple with good performance, the BoF method discards the information about the spatial order of features, which limits its capability of capturing shapes and locations of objects.

To overcome this drawback, the SPM method first partitions an image into $(2^l) * (2^l)$ subregions in different scales, typically $l = 0, 1, 2$, and computes the BoF histogram within each subregion. Then a vector representation of the image is formed by concatenating all the histograms. When the scale $l = 0$ is used, SPM reduces to BoF.

There are 4 stages in the SPM approach. At first in the "Descriptor" layer, feature points are detected densely located on input image and the downsampled SIFT descriptors [38] are extracted from each feature point. Then, K-means clustering is performed on these descriptors, where the number of clusters $k = 200$ is used. We will use SPM with an independent dataset in our experiments to compute the clusters. The cluster centroids are referred to as visual words. Each SIFT descriptor of a given image i is assigned to the closest virtual word according to the Euclidean distance, witch generates the "Code" layer. Next, in all levels of the pyramid, each spatial histogram is weighted by aggregating multiple codes inside each sub-region and subsequently normalized with the L2 norm. Finally, the histograms are concatenated from all sub-regions to generate the histogram representation (4200-dimensional) of the image for image classification.

Scale Invariant Feature Transform

Lowe proposes the scale-invariant feature transform, SIFT [38], a method able to extract distinctive invariant local features from images. There are four stages to produce the features that are invariant to image scale and rotation. The first two stages deal with the locations of the scale-invariant interest points. Then the third stage assigns orientations to each keypoint location based on the most dominant direction of local image gradients. With the location and orientation of the keypoint, the local image gradients are transformed into a rotation-invariant representation in the final stage.

In the superparsing system, however, the local SIFT descriptors are extracted using a dense regular grid instead of interest points. This is following the decision made in Spatial Pyramid by Lazebnik et al., for better scene classification. Now we describe the details of the feature extraction stage adapted for dense SIFT (DSIFT). For every pixel in an image, a 16x16 neighbourhood is divided into 4x4 cells consisting of 16 pixels each, and the gradient orientations inside each cell are quantized into 8 bins. Then by concatenating the 16 orientation histograms, we can obtain a $4 - by - 4 - by - 8 = 128$-dimensional vector as the SIFT representation for a pixel [36]. Following this, the vector is normalized to unit length, which makes it invariant to illumination changes. Finally, a multiplier is applied to put elements of these descriptors within a byte (0-255) range and then quantized.

Maximum Response Sets

The MR8 filter bank proposed by Varma and Zisserman [70] consists of 38 filters but only 8 filter responses. To achieve rotation invariance, the 38 filters from a common Root Filter Set (RFS) at multiple orientation are collapsed by recording only the maximum filter response over each orientation. The filter bank consists of 2 anisotropic filters at 6 orientations and 3 scales $(\sigma_x, \sigma_y) = (1, 3), (2, 6), (4, 12)$, and 2 isotropic filters with the property of rotational symmetry. The 2 orientated filters are made up of an edge filter and a bar filter, while a Gaussian and a Laplacian of Gaussian (LoG) both with $\sigma = 10$ pixels form the rotationally symmetric ones. But the number of responses is reduced to 8 (3 scales for 2 oriented filters, plus 2 isotropic) by measuring only the maximum response of the anisotropic filters across all orientations.

However, as Martinelli states in [42], the MR8 filter bank with the property of rotation invariance may not be desired as a texture filter bank used for segmentation. Since some classed could be described by their feature orientation, recording only the maximum response over all orientations will loose some important texture information. Therefore, together with the dense SIFT algorithm, the RFS filter bank where all the 38 responses are kept is used to produce the texture descriptors of superpixels in the superparsing system.

2.5 Contextual Constraints in Image Parsing

As introduced in the system overview in Section 1.3 and described in [64], a common framework for image parsing with Markov Random Fields (MRFs) or Conditional Random Fields (CRFs) over segmentation regions roughly contains the following steps:

- For each superpixel of each image in the training set, extract local features from an area around that superpixel.
- Given this set of features and ground truth labels, train a local model to produce a compatibility score for each feature and each label.
- Run the trained classifier on a query image and use the output as the unary term of an MRF or CRF.
- Define a pairwise term (or a smoothing term) of the MRF or CRF, where typically an adjacent graph is defined over the superpixels of the image, or optionally train parameters of the MRF or CRF.
- Perform maximum a posteriori (MAP) inference on the MRF or CRF.

Beyond simple smoothing in the MRF model, context emerges as an important factor to enforce constraints on the image labeling. Tighe and Lazebnik [64] learn contextual relationships, like boats occur surrounded by water and cars with roads, via co-occurrence between labels. A pairwise term in the MRF formulation is used to model simple next-to relationships between labels based on label co-occurrence counts, which penalizes classes being assigned to adjacent regions if they rarely or never occurr next to each other.

Rabinovich et al. also learn contextual relationships via simple co-occurrence statistics in the training dataset [51]. They represent the statistics using a matrix of label co-occurrence counts, in which an entry (i, j) counts the times an object with label i appears in a training image with an object with label j. These relationships are then

incorporated into a CRF as penalty terms.

Following this trend, multiple forms of context based on co-occurrence, location, and appearance are incorporated into a CRF to provide stronger contextual constraints [19, 17, 18]. Galleguillos et al. introduce a novel model for multi-class object localization that incorporates contextual interactions at pixel, region and object level [18]. The contextual interactions are studied using three different sources of context including semantic, boundary support and contextual neighbourhoods.

Ladicky et al. showed how to efficiently incorporate a global context penalty within hierarchical CRFs for object class image segmentation , i.e., penalizing unlikely pairs of labels from being assigned anywhere in a image [30]. A global energy function is defined for such CRF model, which combines features at quantization levels of the image on pixels and segments.

There is also some work on learning contextual smoothing directly from the images instead of separately modelling and computing stages in many energy minimization algorithms (e.g., the random fields based image parsing approaches). The auto-context algorithm proposed by Tu integrates the image appearances together with the context information by learning a cascade of classifiers [66]. In this algorithm, the output of the classifier from one round is produced and then used to train another classifier. As Tu states in [66], the classifiers in different rounds may choose different supporting neighbours to either enhance or suppress the current probability towards the ground truth, which gives more flexibility compared to a fixed neighbourhood structure in MRFs or CRFs.

Next, we would like to describe in detail two typical methods to incorporate the contextual constraints into a CRF model for the object categorization and class segmentation [16, 19].

Class Segmentation with Superpixel Neibourhood

Fulkerson el al. present a method to identify and localize object classes in images for class segmentation [16]. This method segments the input images into superpixels using quick shift [71] at first. Then a bag-of-features classifier on the superpixel level is constructed using a support vector machine (SVM) on the histogram of local features extracted from each superpixel. Having this classifier, they regularize it by aggregating histograms in the neighbourhood of a superpixel. The size of the neighbourhood N can be increased to include more features in histograms of adjacent superpixels and provide a better description of the objects in the image. However, the merging of histograms results in larger N and blur the boundaries of the objects. Thus, in order to recover more accurate boundaries while still keep the advantage of increasing N, a conditional random field is defined on the superpixel graph $G(S, E)$ to refine the results. Let $P(\mathbf{c}|G; w)$ be the conditional probability of the set of class label assignments $\mathbf{c}$, the energy function is defined as:

$$-\log(P(\mathbf{c}|G; w)) = \sum_{s_i \in S} \Psi(c_i|s_i) + w \sum_{(s_i, s_j) \in E} \Phi(c_i, c_j|s_i, s_j), \tag{2.6}$$

where w is weight used to balance the spatial regularization. The unary potential is defined by the probability from the SVM classifier for each superpxiel s_i:

$$\Psi(c_i|s_i) = -\log(P(c_i|s_i)) \tag{2.7}$$

and the pairwise potential is formulated to represent the similarity of adjacent superpixels as:

$$\Phi(c_i, c_j|s_i, s_j) = \left(\frac{L(s_i, s_j)}{1 + \|s_i - s_j\|}\right)[c_i \neq c_j] \tag{2.8}$$

where $L(s_i, s_j)$ is the shared boundary length between superpixels s_i and s_j, and $||s_i, s_j||$ is the norm of color difference between them. Multiplication by $[\cdot]$ is a zero-one indicator function. Once the CRF model is learned, a graph-cut inference is performed

with the multi-label graph using α-expansion.

As authors state, the graph is used in the algorithm only to define neighbourhoods and construct a conditional random fields other than mine discriminative substructures, which avoids heavy classifier training in some degree. Following the idea of integrating stronger local features and learn context via similarity between adjacent superpixels, we adapt this algorithm into the superparsing system to achieve better refinement for the local labeling in Chapter 4.

CoLA

An object categorization approach, CoLA (for Co-occurrence, Location and Appearance), is proposed by Galleguillos et al. in [19]. CoLA incoporates conext of co-occurrence and relative location into a conditional random field (CRF) based on the appearance of images. The CRF model is used to maximizes object class assignment according to both semantic and spatial relevance. This approach starts with computing multiple stable segmentations for the input image. Each segment is regarded as an individual image and integrated into the BoF model [50] for recognition depending on the appearance of these images. After this recognition, each segment is associated with a list of ranked candidate labels. Then to incorporate spatial and semantic context into the initial recognition system, a CRF is modelled on a fully connected graph, where each segment is a node. Finally, based on local appearance, contextual relevance and spatial arrangement, each segment can be assigned to a proper category label.

The relative location between objects is modelled using simple pairwise features in CoLA and the basic spatial relationships include *above, below, inside* and *around*. Within these relationships, four spatial context matrices are learned to capture the co-occurrence of category labels. Each entry (i, j) in a matrix counts the times an object with label i appears in a training image with an object label j according to a given pairwise relationship.

The CoLA system provides benefits for improving recognition accuracy by incorporating strong semantic and spatial context. This also gives us some inspiration on how to improve the superparsing system.

2.6 Evaluation Methods

Since we aim to achieve meaningful semantic segmentations as well as object recognition in the image parsing process, we need to decide on evaluation metrics. Unlike simple image labeling evaluation such as the percentage of correctly labeled pixels compared to the ground truth, the evaluation of segmentation is more complex. Here we give a brief review of the existing region-based and boundary-based segmentation evaluation measures. In addition, there are more segmentation evaluation measures reviewed in [46, 41].

2.6.1 Region-based Measures

The Variation of Information (VoI) metric is introduced for the purpose of clustering comparison [43]. It defines the distance between two segmentations according to their average conditional entropy as:

$$VI(S, S') = H(S) + H(S') - 2I(S, S'), \tag{2.9}$$

where S and S' are two different segmentations, while H represents the entropies and I is the mutual information between them.Thus, it roughly measures how different these two segmentations are using the amount of randomness in one segmentation which is not able to explained by the other.

The Rand Index (RI) was also originally introduce for general clustering evaluation [53]. It compares the compatibility of assignments by counting the pairs of pixels whose labeling are consistent between the computed segmentation S and the ground truth G. The Rand Index is computed by the sum of the number of pairs of pixels that

have the same label and those with different labels in S and G, divided by the total pairs of pixels. Variants of the Rand Index have been proposed such as Probabilistic Rand Index (PRI) [67] and Normalized Probabilistic Rand (NPR) [68] for the case of multiple ground-truth segmentations.

2.6.2 Boundary-based Measures

The Boundary Displacement Error (BDE) measures the average displacement error of corresponding boundary pixels between two segmented images [15]. Particularly, the error of one boundary pixel is defined as the distance between the pixel and the closest pixel in the other boundary image.

The F-measure is a precision-recall framework operating by comparing generated contours of segmentations to human ground-truth data [40]. It allows evaluation of segmentations by regarding region boundaries as contours. The precision-recall curve in the F-measure captures the trade-off between accuracy and noise. Precision is the probability that the segment boundaries are valid, and recall is the probability that the ground truth data is detected. The precision-recall measures are very meaningful in the context of boundary detection with the help of boundary maps. Let P be the precision and R for the recall, the F-measure is defined as:

$$F = PR/(\alpha R + (1 - \alpha)P), \tag{2.10}$$

which captures the this trade off as the weighted mean of P and R.

Similar to F-measure, a simple formulation of precision/recall values is defined to characterize the agreement between the region boundaries of two segmentations [10]. A more detailed description of this measurement will be provided in Chapter 5.

2.7 Summary

In this chapter, we discussed related work on image segmentation and image parsing problems. Our effort is directed to the question on how to apply the object recognition approaches to obtain meaningful object-level segmentations. Section 2.2 explored existing parametric and nonparametric approaches for possible solutions to our problem, and focused our attention on the superparsing system of Tighe and Lazebnik. Following the approach of the superparsing system, we presented various superpixel methods to obtain a slight over-segmentation in Section 2.3 and discussed different methods to incorporate contextual constraints into random field based image parsing system in Section 2.5. We will show in this thesis how our insights from this review leads us to improve semantic segmentation of images as well as image labeling.

Chapter 3

Background on Superparsing

In this chapter we present the details regarding to the implementation of the basic retrieval-based image parsing system proposed by Tighe and Lazebnik [63]. It is a non-parametric data-driven approach which can transfer the labels from the existing most relevant training samples to the interested query image, as illustrated in Figure 3.1 (from [63]). At first, we will give a brief introduction of the system overview below. After that, we will explain how the superparsing framework works and how we decide to reimplement it the next sections with examples from a sample dataset. Different from Tighe and Lazebnik's original Matlab code version, we use C++ with the help of Opencv library for reimplementation. Here is a list of of other obvious modifications.

- For global features in Section 3.1, we compute Gist feature using Lear's gist library and reimplement the Spatial Pyramid algorithm with the help of dense SiftFlow package from Liu et al. [36].
- Also in Section 3.1, a FLANN library is used to find the nearest neighbouring images in the retrieval set.
- In Section 3.2, we use SiftFlow library form Liu et al. [36] to compute SIFT descriptors and reimplement a filter bank algorithm, RFS, for textons using C++ based on the Matlab code from Varma and Zisserman [70].

For a given query image, the system first retrieves a small set of similar images and their associated semantic labels from the database as the top matches. Since these top matches are lablled, they compute the mapping between the query and retrieved images at a superpixel level to assign the query with matching class annotations. At this point, an initial local labeling for the query is obtained which can be used for further optimization with the help of a Markov Random Field (MRF) or a Conditional Random Field (CRF). The following is a brief summary of the core steps during the processing to obtain the local semantic labeling for every query image.

1. Use global image descriptors to identify a *retrieval set* of training images similar to the query image (Section 3.1, Figure 3.1b).

2. Segment the query image and images in the retrieval set into superpixels using the fast graph-based segmentation method, and then compute 20 different local features representing location, shape, color and texture for all superpixels (Section 3.2, Figure 3.1c).

3. Match each query superpixel to ones in the retrieval set according to these local features to generate the nearest-neighbour superpixels. Compute a likelihood ratio score for each class for all superpixels in the query image based on the matches.Then obtain initial labeling of the image by assigning to each superpixel the class with maximum likelihood (Section 3.3).

4. Use the computed likelihoods together with pairwise co-occurrence energies in an Markov Random Field (MRF) framework to compute the optimized global labeling of the image (see Section 4.2.2).

3.1 Retrieval Set

Similarly to several other data-driven methods [56, 65, 35], image retrieval is the first step for parsing a query image, which gives a relatively small retrieval set of training images. The purpose of finding such a subset of training images is to expedite the parsing

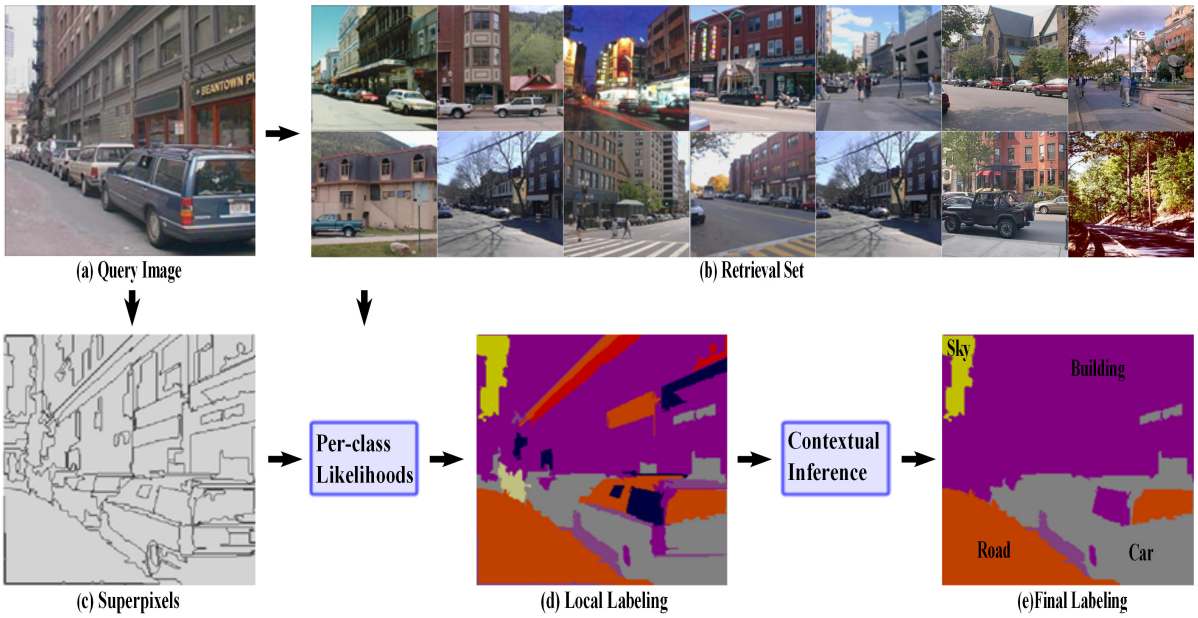

Figure 3.1: System pipeline of scene parsing. Note that the superpixels in (c) are produced by graph-based segmentation [12]. In our work, we will use SLICO [1] superpixels (see Chapter 4).

process and reduce confusion by ignoring irrelevant information. This is critical since it determines the labels used to parse the query image. A good retrieval set will contain images that have similar scene type, objects, color distribution and spatial layouts to the query image. On the other hand, if the retrieval algorithm fails to find enough similar images and true labels, the method is not able to recover them in the subsequent superpixel matching step. Therefore the likelihood score will be lower and we can have less confidence in the resulting labeling. In order to capture the similarity, a spatial pyramid [32], Gist [48], and a color histogram are used as the global image features with details given in Table 3.1. Therein, as described in Section 2.2, a spatial pyramid indicates the shape and location of objects in an image based on the bag-of-features (BoF) image matching algorithm, and the 4200-dimensional spatial pyramid histogram is extracted from the sub-regions in a 1x1, 2x2, 4x4 pyramid. Besides, Gist is a low dimensional representation of a scene that describes a spatial structure of a scene using the concept of naturalness, openness, roughness, ruggedness and expansion. For each image, we use lear's gist library to compute a 960-dimensional color gist in 3 scales with 8, 8, 4 orientations respectively.

After global features are extracted, Tighe and Lazebnik used a simple algorithm to compute the Euclidean distance between the query image and training images according to each global feature, and then sort the distance for each feature type to get three separate rankings of images as well as their corresponding indices. However, the drawback of the method is that this process has to be done for every single query image. To overcome this inconvenience, there are a number of tree-based methods [5, 61, 45] for nearest-neighbour binary search which can complete this kind of search in O(logN) time complexity once the index of the kd-tree is created. The Fast Library for Approximate Nearest-Neighbours (FLANN) [45], a library of approximate nearest-neighbour search algorithms, is a publicly-available software that will automatically choose the optimal algorithm for a given dataset and desired precision. The kd-tree based FLANN works well for exact nearest-neighbour search in low dimensional data, but quickly become less effective as the number of dimensions grows beyond 10. If higher precision in higher dimensions has to be achieved, it results in increased complexity and hence longer computation time.

As the global features (spatial pyramids and Gist) used to find the retrieval set are both high dimensional vectors, and we need best nearest-neighbour matches to find most similar images which contains the true labels in parsing the query image. In our work, although the FLANN method suffers from the disadvantage of reduced precision for high dimensional data, it is not affecting our result because our training database is not very large. In practice, FLANN can give us very efficient performance by achieving very high precision for color histograms (24-dimensional), Gist descriptors (960-dimensional) and spatial pyramids (4200-dimensional). Thus, we choose to use FLANN to find the nearest neighbour for the query image.

For each feature type, all training images are ranked in ascending order of Euclidean distance from the query image. Then, the top K most similar images from each ranking are taken as a pool to produce a single ranking for the query image. In particular, the minimum of the per-feature ranks is assigned to each image in the pool, and the final

Name	Parameter	Dimension
Spatial pyramid	3 levels, SIFT dictionary of size 200	4200
Gist	3-channel RGB, 3 scales with 8, 8 and 4 orientations	960
Color histogram	3-channel RGB, 8 bins per channel	24

Table 3.1: Global features for retrieval set computation

top-ranking K images are used as the retrieval set. We keep using 200 as a typical value of K in the experiments. (a typical value of K in the experiments is 200).

Algorithm 1 Ranking of the retrieval set

Input: Stacks of highest ranking images for similarity in global features {color histogram, gist, spatial pyramid}
Output: Map M with K highest ranking similar images without duplicates
k = 0
while $k < K$ {
  for feature ∈ {gist, color histogram, spatial pyramid}{
    if $I = S_{feature}.pop()$ not in M
    M.add(I), k++
  }
}

Empirically, this method works betters than other schemes, such as simply averaging the ranks. Intuitively, taking the best scene matches from each of the global descriptors leads to better superpixel-based matches for region-based features that capture similar types of cues as the global features.

Tighe and Lazebnik also examine the contributions of different global features and the effect of changing the retrieval set size K in the experiments in [63].

3.2 Superpixel Features

The goal is to assign every pixel of the query image with semantic labels, based on retrieval images and their corresponding ground truth semantic labels (annotated by human). Assigning labels on a per-pixel basis [23, 35] is an option. However, it tends to be too inefficient. Also, a single pixel alone does not contain sufficient information for recognition. On the other hand, like [25], [39] and [52], recognizing pixels together with their proper neighbouring regions, i.e. superpixels, not only reduces the complexity of the problem by improving the computational efficiency, but also provides more information about the structure of the scene. Other than producing superpixels using fixed-size square windows centered on every pixel in the image, a bottom-up segmentation supports better spatial grouping for aggregating features that could belong to a single object (and gives an intuitive representation of context based interactions between objects in the image).

The fast graph-based segmentation algorithm of [12] is used to obtain superpixels. Beginning from single-pixel regions, this method merges two regions if the minimum intensity difference across the boundary is greater than the maximum difference within the regions, with a bias towards larger regions. In our implementation, we use the code publicly released by the authors with the parameters $\sigma = 0.8, k = 200, min\_size = 100$. Therein, σ indicates smooth term; k is the constant for the threshold function ad min_size is used to enforce the minimum size of each component in the segmentation in the post-processing stage. Some segmentation results are shown in Figure 3.2.

Similar to Malisiewcz and Efros's work [39], 20 different local features are used to describe shape, location, color and texture of a superpixel. A complete list of these features is given in Table 3.2.

By modelling color, we can implicitly identify materials and objects that correspond to particular semantic classes, making color a powerful cue. For instance, the sea is usually blue and green for grass. Similarly to color, texture provides a cue for the semantic class of a superpixel though its relationship to materials and objects in the world. Textures are typically intensity (or color) variations that indicate roughness of object surface.

Figure 3.2: Graph-based segmentation results: left is the original sea view images in the dataset. After segmentation using graph-based method, right shows their segmented results filled with random colors.

Name	Detail	Dimension
Shape	Mask of superpixel shape over its bounding box (8×8)	64
	Bounding box width (or height) relative to image width (or height)	2
	Superpixel area relative to the area of the image	1
Location	Absolute mask of superpixel shape over the image (8×8)	64
	Top height of bounding box relative to image height	1
Texture/SIFT	Texton histogram	100
	Dilated by 10 pixel texton histogram	100
	Quantized SIFT histogram	100
	Dilated by 10 pixel quantized SIFT histogram	100
	Left/right/top/bottom boundary quantized SIFT histogram	100×4
Color	RGB color mean	3
	RGB color std. dev.	3
	Color histogram (RGB, 11 bins per channel)	33
	Dilated by 10 pixel color histogram (RGB, 11 bins per channel)	33
Appearance	Color thumbnail, an image crop along the bounding box (8×8)	192
	Masked color thumbnail	192
	Grayscale GIST over superpixel bounding box	320

Table 3.2: Local features for superpixels

In terms of texture, for each image, we compute textons using an anisotropic filter bank, Root Filter Set (RFS, as shown as full MR8 in the code) with Matlab code [1] written by Varma and Zisserman. RFS is a multi-scale, multi-orientation filter bank with 38 filters which consists of 6 orientations at 3 scales with 2 oriented filters each, plus 2 isotropic filters. To compute histograms of textons, filter responses of all training images are generated by this RFS filter bank and a set of exemplar filter responses are chosen from 2000 random training images via K-means clustering as the texton cluster centres. In our work, we have 100 of this kind of cluster centres collected as virtual words in a texton dictionary. Having learnt such a dictionary, each filter response of a given image is assigned to the texton of the closest centre in the dictionary according to Euclidean distance. Finally, the frequency with which each texton occurs in the assigning forms the histogram of textons for the given image.

In addition, dense SIFT descriptors are extracted as another texture type using SiftFlow package from C. Liu et al. with $cell\_size = 4, step\_size = 1, isBoundaryIncluded = true$. The same processing is done to build the histogram of dense SIFT descriptors by quantizing the features from a vocabulary of 100 SIFT words. It is also similar to steps in the spatial pyramid described in Section 2.1. Figure 3.3 shows the visual representation of both features.

In particular, the system computes not only the normalized histograms of textons and dense SIFT descriptors over the superpixel region as basic texture features, but also a dilated version of that region to incorporate the contextual texture information. This is done by dilating the superpixel mask by 10 pixels at first, and then extracting the same features from the dilated superpixel region. In addition, the left, right, top and bottom boundary histograms of SIFT descriptors are also computed to get more powerful superpixel representation. In order to do this, the superpixel region is dilated and eroded by 5 pixels to compute the difference between them as the boundary region, and then obtain the left/right/top/bottom parts of the boundary by cutting it with an

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

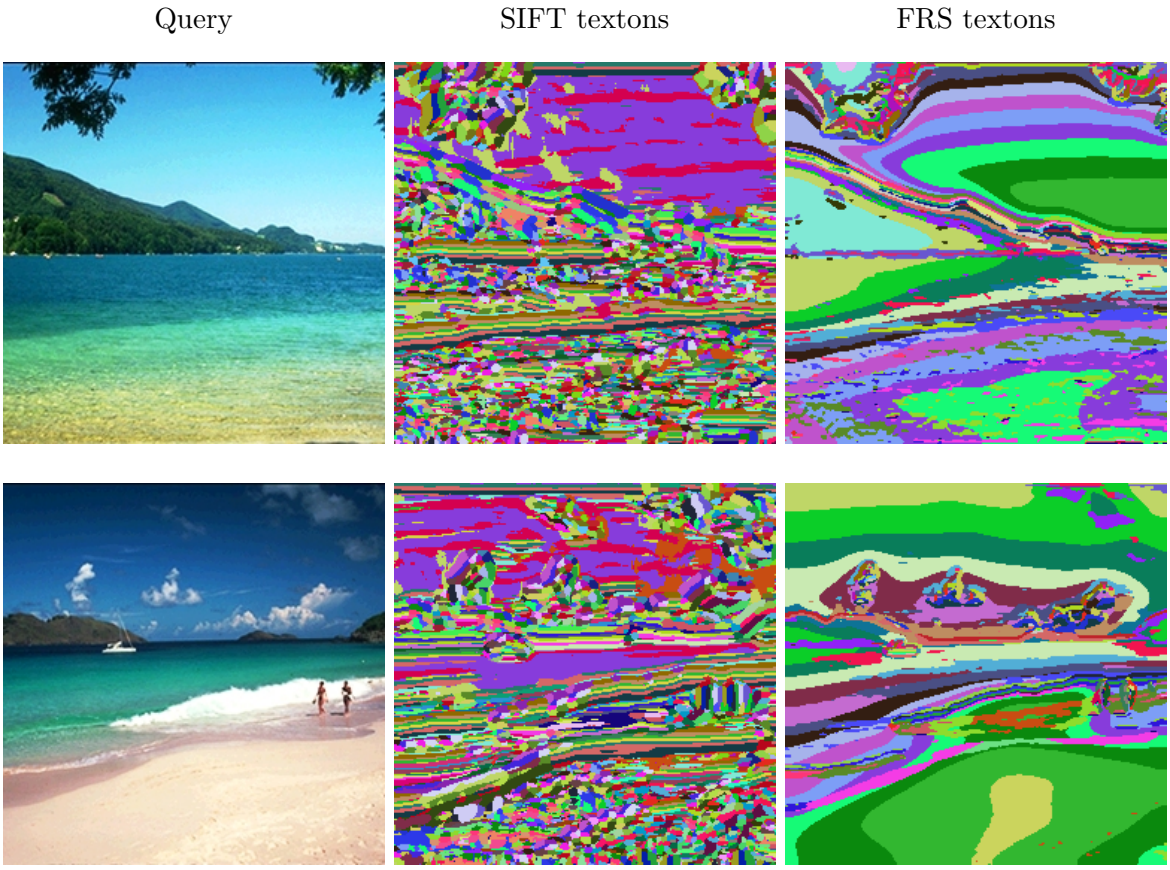

Figure 3.3: The visual representation of SIFT textons and FRS textons.

"X" drawn over the superpixel bounding box.

For each superpixel in the training dataset, all these 20 features are computed and stored together for pre-processing. However, only valid superpixels with a certain label can be used as training superpixels for further superpixel matches. We have 33 classes in our SiftFlow dataset, to associate a class label with a superpixel, we start by masking the ground truth image with the segments of its original image produced by the segmentation algorithm. Then for each ground truth segment mask, we define a valid superpixel if 50% or more of superpixels belong to the same class. For the valid ones, a label (from 1 to 33 in our database) of that class will be assigned to it, otherwise, 0 will be used to represent a non-labeled superpixel. For the preparation of training dataset, invalid superpixels will be discarded, remaining valid ones will be used to generate a "SegmentIndex" table consisting of image id, superpixel id, superpixel label and its size.

3.3 Local Superpixel Labeling

Having segmented the query image and extracted the features of all its superpixels, we intend to label these query superpixels by matching them with the labeled training superpixels according to their local feature, i.e. decide which class a query superpixel belongs to. At this purpose, we would like to use the idea of Bayes' classification [6] that compares the probabilities of different hypotheses in order to make a classification.

3.3.1 Background on Bayes' Classifier

A Bayes' classifier (Peters et al.[49]; Bishop et al.[6]) generally assigns a data point to the class with the highest probability of membership for that data point. For c different classes $C_1, ..., C_c$, the posterior probability $P(C_k|x)$ gives the probability of the pattern belonging to class C_k once we have observed the data point x. The probability of misclassification is minimized by selecting the class C_k having the largest posterior

probability. A feature vector x is assigned to class C_k if

$$P(C_k|x) > P(C_j|x) \tag{3.1}$$

for all $j \neq k$.

To compute the posterior probabilities $P(C_k|x)$, Bayes' theorem is used:

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \tag{3.2}$$

where $P(C_k)$ is the priori unconditional probability that a data point is in class k, $P(x|C_k)$ is the probability for x to be observed given the assumption that the data is sampled from class k, and $P(x)$ is the unconditional probability of observing x. Since the unconditional density $P(x)$ is independent of the class, it may be dropped from the Bayes' formula for the purpose of comparing posterior probabilities. Thus, combining Bayes' theorem with the classification rule 3.1 gives:

$$P(x|C_k)P(C_k) > P(x|C_j)P(C_j) \tag{3.3}$$

for all $j \neq k$.

From another perspective, the classification process can be reformulated in terms of a set of discriminant functions $y_1(x), ..., y_c(x)$ such that an input vector x is assigned to class C_k if

$$y_k(x) > y_j(x) \tag{3.4}$$

for all $j \neq k$.

If we use Bayes' theorem, and note that the unconditional density $P(x)$ in the denominator does not depend on the class label C_k, and therefore does not affect the classification decision, we can write an equivalent discriminant function in the form

$$y_k(x) = P(x|C_k)P(C_k) \tag{3.5}$$

Since only the relative magnitudes of the discriminant function are important in determining the class, we can replace $y_k(x)$ by $g(y_k(x))$, where $g(\cdot)$ is any monotonic

function, and the decision of the classification will not be affected. For example, we can rewrite our discriminant function using logarithms as

$$y_k(x) = \log P(x|C_k) + \log P(C_k) \tag{3.6}$$

For two-class decision problems, instead of using two discriminants $y_1(x)$ and $y_2(x)$, usually a single discriminant function will be used in the form

$$y(x) = y_1(x) - y_2(x) \tag{3.7}$$

and now we use the rule that x is assigned to class C_1 if $y(x) > 0$ and to class C_2 if $y(x) < 0$, in other words, $y(x)$ decides if x belongs to class C_1 or not. From the remarks above, we can use several forms for $y(x)$ including

$$y(x) = P(C_1|x) - P(C_2|x) \tag{3.8}$$

or alternatively

$$y(x) = (\log P(x|C_1) + \log P(C_1)) - (\log P(x|C_2) + \log P(C_2)) \tag{3.9}$$

$$= \log \frac{P(x|C_1)}{P(x|C_2)} + \log \frac{P(C_1)}{P(C_2)} \tag{3.10}$$

By this way, it is often possible to determine suitable discriminant functions from the training data without having to go through the intermediate step of probability density estimation.

Following this idea, for each special class C_k in multi-class case, we can use a single discriminant function similar to formula 3.7 to decide if x belongs to class C_k or not.

$$L(x, C_k) = y_c(x) - y_{\bar{c}}(x) \tag{3.11}$$

Similarly, an alternative form is:

$$L(x, C_k) = \log \frac{P(x|C_k)}{P(x|\overline{C_k})} + \log \frac{P(C_k)}{P(\overline{C_k})} \tag{3.12}$$

which not only keeps the monotonicity of (3.5), but also gives a bounded range output.

3.3.2 Labeling Details

In our problem, the query superpixels are our observed data points. Based on the Bayes' classification method, for each query superpixel s_i and each class c that is present in the retrieval set, we use similar logarithmic discriminant function in the form

$$L(s_i, c) = \log \frac{P(s_i|c)}{P(s_i|\bar{c})} + \log \frac{P(c)}{P(\bar{c})} \tag{3.13}$$

where $\bar{c}$ is the set of all classes excluding c and $P(s_i|c)$ is the probability for s_i to be observed given the assumption that the data is sampled from class c. Since the priori unconditional probability $P(c)$ for each class in our dataset is a constant and equals to each other, it may be dropped with no effect for the comparison. Thus, the discriminant function can transform to a log likelihood ratio score. We make the Naive Bayes assumption that features f_i^k are independent of each other given the class, then the log likelihood ratio is defined as

$$L(s_i, c) = \log \frac{P(s_i|c)}{P(s_i|\bar{c})} = \log \prod_k \frac{P(f_i^k|c)}{P(f_i^k|\bar{c})} \tag{3.14}$$

$$= \sum_k \log \frac{P(f_i^k|c)}{P(f_i^k|\bar{c})} \tag{3.15}$$

For each query superpixel and each feature type f_i^k, we compute the kth feature distance between f_i^k to all superpixels in the retrieval set in order to find its nearest-neighbouring superpixels whose kth feature distance from f_i^k is below a fixed threshold t_k. Let N_i^k denote such neighbourhood of f_i^k, and D denote the set of all valid superpixels in the training set. Usually we can compute such likelihood ratio $P(f_i^k|c)/P(f_i^k|\bar{c})$ via parametric density estimates together with given density distributions. However, the density function is unknown in our case and only nonparametric methods can be applied to estimate the distribution based on given samples in our dataset. For instance, find superpixels whose kth feature just equal to that of s_i and get the likelihood as 1 or 0 according to if such supperpixel belonging to class c or not. As you can see, this idea is not good enough to give confident estimates and it may not be possible to find such exact superpixel in many situations. To overcome this drawback, a window centered at f_i^k can be used to collect similar samples around f_i^k and smooth the unknown density curve. Thus, we try to find more similar samples in a reasonable neighbourhood instead

of single local sample, which gives reliable estimates from a window function. With the help of nonparametric density estimates of features from the required class(es) in $\mathcal{N}_{\rangle}^{\|}$, the neighbourhood of f_i^k, each likelihood ratio $P(f_i^k|c)/P(f_i^k|\bar{c})$ is computed as

$$\frac{P(f_i^k|c)}{P(f_i^k|\bar{c})} = \frac{(n(c,\mathcal{N}_{\rangle}^{\|}) + \epsilon)/n(c,\mathcal{D})}{(n(\bar{c},\mathcal{N}_{\rangle}^{\|}) + \epsilon)/n(\bar{c},\mathcal{D})} \tag{3.16}$$

$$= \frac{n(c,\mathcal{N}_{\rangle}^{\|}) + \epsilon}{n(\bar{c},\mathcal{N}_{\rangle}^{\|}) + \epsilon} \times \frac{n(\bar{c},\mathcal{D})}{n(c,\mathcal{D})} \tag{3.17}$$

where $n(c,\mathcal{S})$ is the number of superpixels in set $\mathcal{S}$ with class label c, respectively, $n(\bar{c},\mathcal{S})$ indicates not c, and ϵ is a constant added to prevent zero likelihoods and smooth the counts.

Note that the density estimates are normalized by counts over the entire training set $\mathcal{D}$ instead of over the retrieval set. This is because the likelihood estimates are expected to be proportional to the true likelihoods across the whole dataset as descripbed by Tighe.

At this point, for each query superpixel s_i, we have 33 log likelihood ratio scores from Equation 3.16 that represent the probabilities for s_i belonging to each class. Then, we can obtain a labeling of the image by simply assigning to each superpixel the class that maximizes Equation 3.16. As in Figure 3.4, an example image shows good and meaningful initial labeling results. Note that for the non-labeled area in the ground truth labels, even it is mislabeled as "Rock" instead of "Mountain" as we can tell in the original image, the shape (or the boundary for this area) is recognized very well. In fact, this works exactly as what we want for the higher goal of giving us a meaningful and clear segmentation.

In order to compute the fixed thresholds, we use the L2 distance for all features, and set each threshold t_k to the median distance to the Tth neighbour for the kth feature type over the dataset D. In our impelementation, we first randomly choose at most 2000 images and take all valid superpixels and their local features from the chosen images as a separate dataset A. Then 1000 superpixels and their corresponding local features are randomly chosen from A as a small dataset B. Thirdly, for each feature type f_k, we obtain a distance matrix M with 1000 rows by computing the Euclidean distance from

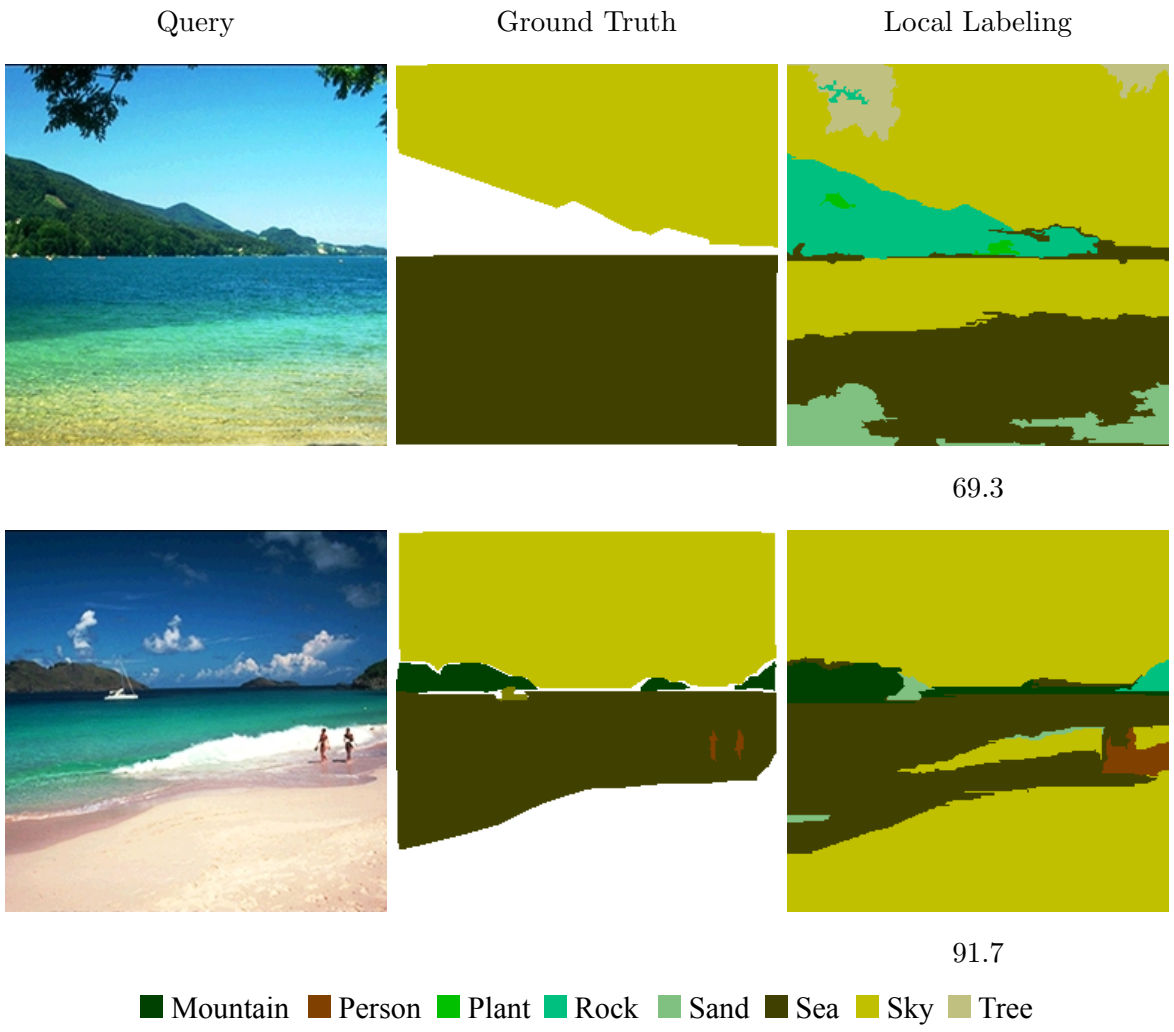

Figure 3.4: Local labeling compares to ground truth labels. The labeling is generated using graph-based segmentation. The number under local labeling is the percentage of pixels labeled correctly. Note white area means the ground truth label is empty.

B to A. Within this distance M, each entry (i, j) represents the Euclidean distance from ith superpixel in A to jth superpixel in B. In the next step, the distance for each feature type is ranked increasingly by row and the element in the Tth column of each sorted row indicates the Tth nearest-neighbours for each superpixel in A. Finally, the fixed threshold t_k will be assigned as the median distance to Tth nearest-neighbour for the kth feature, i.e. the median of elements in Tth column in the sorted distance matrix.

In our implementation, we compute the fixed threshold t_k using L2 distance. At first, we randomly choose 2000 images as a separate dataset consists of training superpixels and all their local features used to compute the thresholds. Then, 1000 superpixels are randomly chosen from this separate training dataset as a small query set. Thirdly, for each feature type f_k, we obtain a distance matrix with 1000 rows by computing the Euclidean distance between the queries and the training superpixels over the separate dataset. Within this distance matrix, each entry (i,j) represents the Euclidean distance from query i to training superpixel j. In the next step, the distance for each feature k is ranked increasingly by row, and the element in the Tth column of each sorted row indicates the Tth nearest-neighbours for each query. Finally, the fixed thresholdt_k will be assigned to the median distance to Tth nearest-neighbour for the kth feature, i.e. the median value of elements in Tth column of the distance matrix.

3.4 Summary

We have explained the details of the basic superparsing system and adapted it into a C++ environment. A nearest neighbour search algorithm, FLANN is also applied to speed up the retrieval set search. With the adapted framework, further extension and improvement will be performed in Chapter 4.

Chapter 4

Superparsing with more accurate boundaries

In Chapter 3, we introduced the basic image parsing system and our implementation of this system. We now extend this approach to obtain more accurate boundaries for our image segmentation. To achieve this goal, we make modification on both, segmentation method and contextual inference compared to Tighe and Lazebnik's work. We start by introducing in Section 4.1 the SLICO segmentation method (see Chapter 2) to the basic framework. Section 4.2 describes the background on random fields for contextual inference. We then discuss how Tighe and Lazebnik apply a MRF to optimize labeling results for an image and how we improve on this with a CRF.

4.1 Segmentation using SLICO method

As illustrated in Figure 3.4, the initial labeling gives meaningful segmented results. Although the graph-based segmentation algorithm has a single scale parameter that influences the segment size, the actual size and number of segments can vary greatly depending on local contrast. For this reason, the algorithm does not offer an explicit control on the number of superpixels or their compactness as we want. However, we believe that more superpixels with smaller regions will capture more precise boundaries

to distinguish different objects in an image.

Figure 4.1: SLICO segmentation results: the left are original images, the middle column shows segmented results with $N_{SP} = 43$, while the right for $N_{SP} = 100$

In attempt to prove our hypothesis, we use SLICO (a "zero" parameter version of SLIC) method [1] to do the segmentation. The compactness parameter is set to 10.0 as default, and then SLICO adaptively chooses the compactness parameter for each superpixel differently and generates regular shaped superpixels in both high-textured and non-textured regions alike. We just need to choose the number of superpixels we want to set parameter N_{SP}. Specifically, we use N_{SP} with 50, 100, 150, 300 and 500 separately on the SIFT Flow dataset in our experiments (for more details, see Chapter 5). However, to explain the methods in this chapter, we use the example labeling results from the Sample dataset as well as Chapter 3. In particular, N_{SP} is assigned to 43 to produce a similar number of total superpixels to that generated by the graph-based segmentation. As shown in the top row of Figure 4.1, segmentation results in the middle with $N_{SP} = 43$ puts some "mountain" and "sea" into same superpixels. However, in

superparsing, a superpixel can only be labeled as one object per superpixel, i.e. parts of such a superpixel can never be labeled correctly. Increasing the number of superpixels to 100 reduces the number of such errors resulting in a more accurate mountain shape.

Figure 4.2 will also serve as an initial labeling example of SLICO segmentation method compared to the graph-based segmentation approach in Figure 3.4.

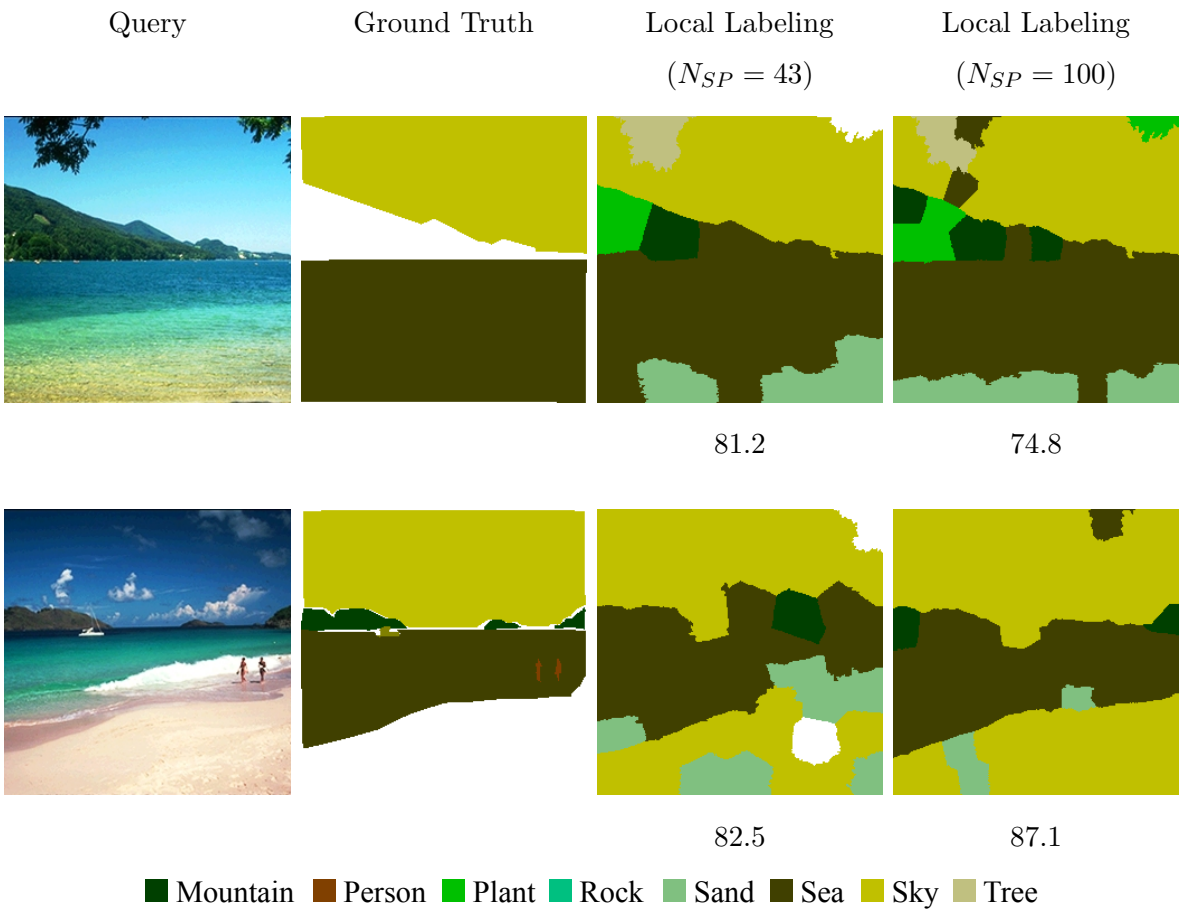

Figure 4.2: Example local labeling resutls using the SLICO segmentation with different number of superpixels. The number under the image is the percentage of correctly labeled pixels compared to the ground truth.

4.2 Contextual Inference

Both Figure 3.4 and Figure 4.2 show good and meaningful labeling results for superpixels, but these results still suffer from mislabeling which may lead to high probability of incorrect boundaries for image segmentation. Although smaller superpixels have the potential to give more accurate boundaries, small regions break up the internal consistency of larger areas and it becomes easier to mislabel these independent small regions. One reason why small regions are easier to mislabel is the lack of possible shape constraints because of the similar size of superpixels.. For instance, two separate small superpixels in the "sky" region are mislabeled as "sea". We observed similar failures in other images. Some of the most glaring labeling failures are "water" completely surrounded by "sky", a bunch of "cars" running on "sea" instead of "highway", etc. To overcome such problems, we would like to enforce semantic contextual constraints on image labeling. Even when the semantic contextual relations between objects' labels are acceptable, we can also incorporate the contextual constraints to achieve more precise segmentation boundaries by correcting mislabeled neighbouring superpixels. In order to do this, we transform the global image labeling problem into a superpixel-based random field as suggested by Tighe and Lazebnik..

In the next sub-sections of this chapter, we first explain the formulation of a random field model for image labeling problems and their standard optimization algorithms in Section 4.2.1. In Section 4.2.2, we show how Tighe and Lazebnik use efficient Markov Random Field (MRF) optimization to capture label co-occurrence context based on underlying feature representations. In Section 4.2.3, we propose a novel Conditional Random Field (CRF) model that incorporates more powerful neighbourhood context for labeling optimization and leads to more accurate boundaries.

4.2.1 Random Fields for Labeling Problems

In this section, we introduce the principles behind Markov and Conditional Random Fields and describe the graph-cut inference techniques, which will be used later in this

chapter.

The random field model [34] consists of an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ composed of a set of vertices or sites $\mathcal{V} = 1, 2, \ldots, N$ and a set of undirected edges $\mathcal{E}$ connecting neighbouring sites in $\mathcal{V}$. Consider the problem of assigning labels to all superpixels in an image, $\mathcal{V}$ is the set of superpixels in the image. A site q is a neighbour of p if they are connected by an edge $(p, q) \in \mathcal{E}$, and the set of neighbours of p is denoted by N_p. Furthermore, similarly to Ladichy et al.'s work [29], each site p in $\mathcal{V}$ associates with one discrete random variable, which may take a value from the set of labels $\mathcal{L} = l_1, l_2, \ldots, l_k$. We use $\mathbf{X} = X_1, X_2, \cdots, X_N$ to denote the set of random variables corresponding to the sites $p \in \mathcal{V}$. An edge $(p, q) \in \mathcal{E}$ connecting sites p and q indicates a dependency between X_p and X_q. In addition, a clique c is a set of sites, such that $\forall p, q \in c$, $p \in N_q$, and $q \in N_p$, and the associated random variables X_c are conditionally dependent on each other. These associated random variables can take any possible values from $\mathbf{L} = \mathcal{L}^N$, and we refer to such possible assignment as a labeling (denoted by $\mathbf{X}$).

Given an observation, i.e., the image data $\mathcal{D}$, and the posterior distribution $Pr(x|\mathcal{D})$ over the labeling of the random field (MRF or CRF) is a Gibbs distribution and the posterior can be written as:

$$Pr(x|\mathcal{D}) = \frac{1}{Z} \exp(-\sum_{c \in \mathcal{C}} \psi_c(x_c)), \tag{4.1}$$

where Z is a normalizing constant called the partition function, and $\mathcal{C}$ is the set of all cliques. For $x_c = x_i \in c$, the term $\psi_c(x_c)$ is known as the potential function of the clique $c \in \mathcal{V}$. The corresponding energy function is defined as:

$$E(x) = -\log Pr(x|\mathcal{D}) - \log Z = \sum_{c \in \mathcal{C}} \psi_c(x_c). \tag{4.2}$$

Then image labeling becomes an estimation or inference problem where the image $\mathcal{D}$ is an observation of some underlying random field $\mathbf{X}$, and the goal is to find the most

probable or Maximum a Posteriori (MAP) estimate labeling $\mathbf{x}^*$ of the random field, which is defined as:

$$\mathbf{x}^* = \arg\max_{x \in \mathbf{L}} Pr(x|\mathcal{D}) = \arg\min_{x \in \mathbf{L}} E(x). \tag{4.3}$$

Pairwise Random Fields

At this point, we need to find an efficient energy function to solve this optimization problem. Following this idea, most labeling problems in vision are formulated as a pairwise random field, whose energy can be written as the sum of unary and pairwise potentials:

$$E(x) = \sum_{i \in \mathcal{V}} \psi_i(x_i) + \sum_{i \in \mathcal{V}, j \in \mathcal{N}_i} \psi_{ij}(x_i, x_j). \tag{4.4}$$

The unary potentials $\psi_i(x_i)$ are typically defined as the negative log likelihood of variable X_i taking label x_i. To encourage neighbouring superpixels in the image to take on the same or a similar label, the pairwise potentials typically encode a smoothness term to establish the coherence between the labeling of these connected superpixels.

In a MRF, only unary potentials depend on the data. If both unary potentials and pairwise potentials rely on the data, then it becomes Conditional Random Fields (CRF). MRF and CRF follow the same principle. Further, the solution for both optimization problems are similar.

Graph-cut based Inference

Minimizing such energy function and find the MAP labeling is a challenge in the general case, as this problem is proven to be NP-hard. However, numerous optimization techniques were developed to solve this problem in specific cases. Here we use a combinatorial optimization technique, the graph-cut (also called st-mincut) algorithm [8, 7], to achieve a global optimum under certain conditions. In order to simplify the description,

let us consider a simple binary case where the label set consists of two labels.

The binary labeling problem is represented by a directed weighted graph where each node represents a variable, and edges connecting neighbouring nodes. These nodes are called internal nodes, and the corresponding edges are called internal links or n-links. The graph contains two terminal nodes, namely the source s and the sink t. The edges connecting each internal node with both terminal nodes are called terminal edges or t-links. Each edge is associated with a cost, or capacity. The cost of an n-link represents the penalty of assigning the corresponding two nodes to different classes, whereas the cost of a t-link defines the penalty of assigning the node to the corresponding terminal. For the case of the MAP estimate of a binary MRF or CRF, defined by 4.4, the n-link costs correspond to $\psi_{ij}(x_i, x_j)$ and the t-links to $\psi_i(x_i)$, where $\mathcal{L} = \{s, t\}$. An s/t cut in such graph partitions the set of nodes into two disjoint subset $\mathcal{S}$ and $\mathcal{T}$, such that $s \in \mathcal{S}$, and $t \in \mathcal{T}$, and the corresponding cost of the cut is defined as:

$$\mathcal{C}_{\mathcal{S},\mathcal{T}} = \sum_{i \in \mathcal{S}, j \in \mathcal{T}} c_{ij}. \tag{4.5}$$

Such a cut defines the configuration $\mathbf{x}$ of the random field, and the cost of the cut is the energy of $\mathbf{x}$. Form this equivalence, we can find a MAP binary labeling by computing the minimum cost cut [28]. By the Ford-Fulkerson theorem [13], the min-cut problem can be equivalently solved by finding the maximum flow from the source s to the sink t, through the capacitated edges. Figure 4.3 shows an example of the min-cut problem solved equally as a maximum flow problem. In this example, we are going to minimize the cost of the cut so that each node in the graph is assgined to the source $\mathcal{S}$ or sink $\mathcal{T}$, and the cost of the edge (i, j) is taken if $(i \in \mathcal{S})$ and $(j \in \mathcal{T})$.

The induction of the binary labeling problem proves that the binary pairwise submodular energy function can be solved exactly using graph-cut, where a binary energy function is submodular if:

$$\psi_{ij}(0,0) + \psi_{ij}(1,1) \leq \psi_{ij}(0,1) + \psi_{ij}(1,0), \forall i \in \mathcal{V}, j \in \mathcal{N}_i. \tag{4.6}$$

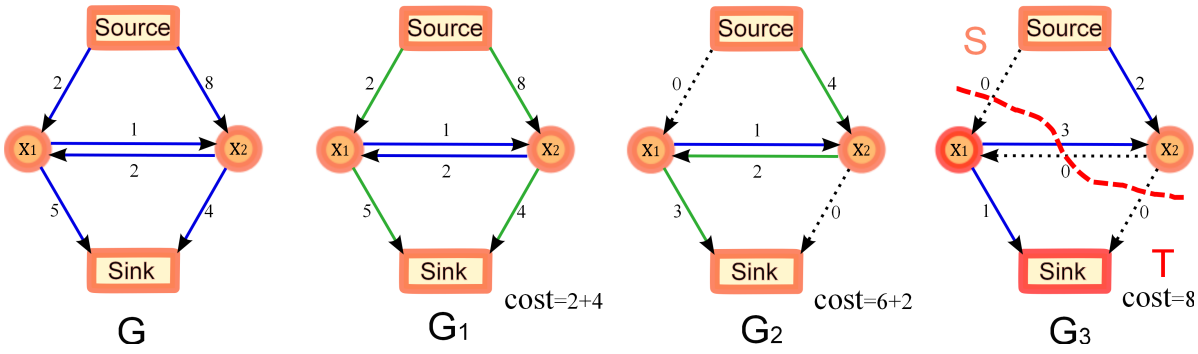

Figure 4.3: An example of the st-mincut algorithm. In G_1, two separate paths from the source s to sink t are marked as green lines, and the maximum capacities in both paths are added to the flow. In G_2, another path is found. In G_3, no more flows from s to t can be found. $\mathcal{S}$ is the set of reachable nodes from the source and $\mathcal{T}$ is the set of nodes that can reach the sink. Then the cost of the cut can be summed as the original costs of the dash edges in G, which is 8 (i.e. 2+2+4) and equals to the maximum flow we found here. Note that extra 2 cost is added to the existing reversed edge of (x_2, x_1).

Further, as discussed in [26], a global optimum is only guaranteed when the energy function is submodular.

However as we already mentioned, the optimization problem for finding the MAP labeling for many practical multi-label cases is NP-hard and approximation algorithms have to be applied. In our work, we choose move making algorithms proposed by Boykov et al. [8] to do the approximation of the optimal solution. Move making algorithms try to iteratively improve the current labeling by finding the move that locally minimizes the energy. Such optimal moves can guarantee that the energy decreases after each move and eventually converges. Next, we give some details about the swap and expansion move algorithms that we use later in our work.

An $\alpha\beta$-swap move allows a random variable x_i whose current label is α or β to take a new label of either α or β. Ladichy et al. also state that optimal $\alpha\beta$-swap moves cannot be efficiently found for all general energies in a random field. One sufficient condition is the semi-metricity of the pairwise potential. A pairwise potential is said to be semi-metric [8] if for all pairs of labels $l_a, l_b \in \mathcal{L}$:

$$\psi^p(l_a, l_a) = 0 \tag{4.7}$$

$$\psi^p(l_a, l_b) = \psi^p(l_b, l_a) \geq 0. \tag{4.8}$$

Trivially,

$$\psi^p(l_a, l_b) + \psi^p(l_b, l_a) - \psi^p(l_a, l_a) - \psi^p(l_b, l_b) = 2\psi^p(l_a, l_b) \geq 0. \tag{4.9}$$

Thus, under the semi-metricity the $\alpha\beta$-swap move energy is submodular and solvable using graph-cut.

In an α-expansion move, every node is only allowed to change its label to α or keep its old label. A sufficient condition that makes the projection submodular is the metricity of the pairwise potential. If pairwise potentials are semi-metric and for any $l_a, l_b, l_c \in \mathcal{L}$, they will be called metric if:

$$\psi^p(l_a, l_b) + \psi^p(l_b, l_c) \geq \psi^p(l_a, l_c). \tag{4.10}$$

4.2.2 Labeling optimization using MRF

With the background discussed in Section 4.2.1, we present next how Tighe and Lazebnik apply the contextual inference on image labeling using MRF model. As in Section 3.3, the annotation c_i is used to represent the class a superpixel s_i belongs to (or, in other words, which label s_i should be assigned to). Then the global image labeling problem is formulated as a minimization of the standard MRF energy function defined over the field of superpixel labels $\mathbf{c} = \{c_i\}$:

$$E(\mathbf{c}) = \sum_{s_i \in \mathcal{SP}} E_{data}(s_i, c_i) + \lambda \sum_{(s_i, s_j) \in \mathcal{A}} E_{smooth}(c_i, c_j), \tag{4.11}$$

where $\mathcal{SP}$ is the set of superpixels, $\mathcal{A}$ is the set of pairs of adjacent superpixels and λ is the smoothing constant. The data term (or the unary potential) is defined as:

$$E_{data}(s_i, c_i) = -\omega_i \sigma(L(s_i, c_i)), \tag{4.12}$$

where $L(s_i, c_i)$ is the likelihood ratio score from Equation 3.14, ω_i is the superpixel weight (the size of s_i in pixels divided by the mean superpixel size). The sigmoid function $\sigma(t) = \frac{1}{1+\exp(-\gamma t)}$ gives a limited range (from 0 to 1, exclusive) with the same monotonicity of t, and γ is set to 0.1 in the implementation. The smoothing term (or pairwise potential) E_{smooth} is defined based on probabilities of label co-occurrence:

$$E_{smooth}(c_i, c_j) = -\log\left[(P(c_i|c_j) + P(c_j|c_i))/2\right] \times \delta\left[c_i \neq c_j\right], \tag{4.13}$$

where $P(c|c')$ is the conditional probability of one superpixel having label c given that its neighbour has label c', estimated by counts from the training set. Instead of using the joint probability $P(c|c')$, the average of two conditionals not only has better numerical scaling, but one also obtains a symmetric quantity. Multiplication by $\delta\left[c_i \neq c_j\right]$ is necessary for graph-cut inference, which is used to ensure this energy term is semi-metric. Then with this energy definition, the MRF inference is performed using the efficient graph-cut optimization with the help of the $\alpha\beta$-swap move algorithm. The graph-cut code comes from [8], and 5 iterations are performed to reach convergence. Figure 4.4 shows some optimal labeling results produced by MRF inference, which successfully flags improbable boundaries. On the SiftFlow dataset, the results labeling improve the per-pixel accuracy by 2-4%.

Tighe and Lazebnik have also experimented with a per-pixel MRF similar to [35], but have found that the per-superpixel formulation achieves the same per-pixel and per-class performance with faster speed. As they state, one possible reason is that it is easier to converge to a better minimum by flipping labels over larger areas of the image other than processing pixel by pixel.

The experiments show that it is reasonable to use the superpixel-based MRF according to its reliable results and superior speed.

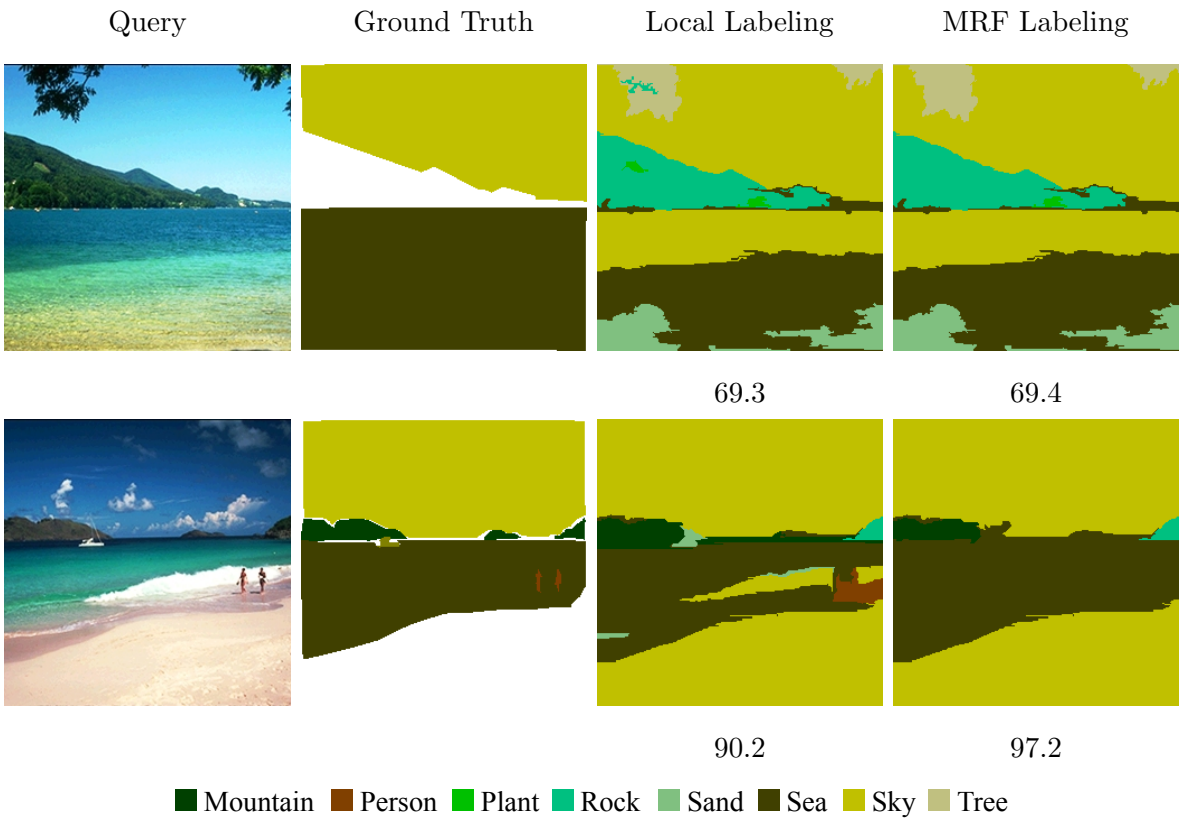

Figure 4.4: Example results of the MRF labeling on the Sample dataset using the graph-babsed segmentation. The per-pixel rate for the MRF labeling results improves over local labeling.

4.2.3 Labeling optimization using CRF

Although Tighe and Lazebnik choose to use MRF due to its low cost in terms of learning and inference, the MRF formulation with label co-occurence is not powerful enough (e.g. the mislabeling in the MRF labeling results in Figure 4.4). Besides, as the number of superpixels increases in SLICO segmentation method, the limitations of MRF formulation becomes more obvious. Thus, to reduce misclassification near the edges of objects in an image, we explore using a CRF to incorporate new constraints such as local features and learn context via connectedness of adjacent superpixels. Following a similar idea to the work of Fulkerson et al. [16], the conditional random fields incorporate such constraints by including them in the pairwise edge potential of the model. To make the annotations consistent with the previous MRF formulation, let $\mathbf{c} = \{c_i\}$ be the set of class label assignments for all superpixels, the energy function of a CRF for the image labeling problem can be defined as:

$$E(\mathbf{c}) = \sum_{s_i \in \mathcal{SP}} E_{data}(s_i, c_i) + \lambda \sum_{(s_i, s_j) \in \mathcal{A}} E_{smooth}(c_i, c_j | s_i, s_j), \tag{4.14}$$

where $\mathcal{SP}$ is the set of superpixels, $\mathcal{A}$ is the set of pairs of adjacent superpixels and λ is the smoothing constant. The unary potential is computed the same way as Equation 4.12 in MRF. Our pairwise potentials E_{smooth} are similar to those of [16], which represents penalties of assigning the same label to each pair of adjacent superpixels (i.e., the dissimilarity of two adjacent superpixels). Following the Bayes' classification method as in Section 3.2, E_{smooth} is defined as a log likelihood ratio score for each pair of adjacent test superpixels $\langle s_i, s_j \rangle$ and their corresponding classes $\langle c_i, c_j \rangle$ that are present in the retrieval set:

$$E_{smooth}(c_i, c_j | s_i, s_j) = \log \frac{P(s_i, s_j | c_i \neq c_j)}{P(s_i, s_j | c_i = c_j)} \times \delta\left[c_i \neq c_j\right]. \tag{4.15}$$

Under the Naive Bayes assumption that features f_i^k are independent of each other given the class, then for a pair of adjacent superpixels $\langle s_i, s_j \rangle$, their absolute feature difference $\left|f_i^k - f_j^k\right|$ are also independent of each other. Thus, we can define the log likelihood ratio as:

$$\log \frac{P(s_i, s_j|c_i \neq c_j)}{P(s_i, s_j|c_i = c_j)} = \log \prod_k \frac{P(\left|f_i^k - f_j^k\right| |c_i \neq c_j)}{P(\left|f_i^k - f_j^k\right| |c_i = c_j)} \tag{4.16}$$

$$= \sum_k \log \frac{P(\left|f_i^k - f_j^k\right| |c_i \neq c_j)}{P(\left|f_i^k - f_j^k\right| |c_i = c_j)}. \tag{4.17}$$

In Equation 3.14, each superpixel s_i and each class c has a likelihood ratio, but here pairwise potentials compute likelihood ratios for each pair of adjacent superpixels and each pair of classes that present in the retrieval set. In other words, for each adjacent pair $\langle s_i, s_j \rangle$, there will be a symmetrical matrix that represents the pairwise potentials between each pair of labels required in the retrieval set. Assume the set of such classes is $\{c_1, c_2, \ldots, c_n\}$, let us consider the case of $\langle c_1, c_2 \rangle$. We use $\mathcal{N}_{ij}^k$ to denote the neighbourhood of $\left|f_i^k - f_j^k\right|$ similarly to the neighbourhood of f_i^k, and $\mathcal{D}$ is for the set of all valid superpixels in the training set. With the help of nonparametric density estimates of feature differences from the required classes in $\mathcal{N}_{ij}^k$, the likelihood ratio for $\langle s_i, s_j \rangle$ in the case of $\langle c_1, c_2 \rangle$ is computed as:

$$\frac{P(\left|f_i^k - f_j^k\right| |c_i \neq c_j)}{P(\left|f_i^k - f_j^k\right| |c_i = c_j)} = \frac{(n(c_i \neq c_j, \mathcal{N}_{ij}^k) + \epsilon)/n(c_i \neq c_j, \mathcal{D})}{(n(c_i = c_j, \mathcal{N}_{ij}^k) + \epsilon)/n(c_i = c_j, \mathcal{D})} \tag{4.18}$$

$$= \frac{n(c_i \neq c_j, \mathcal{N}_{ij}^k) + \epsilon}{n(c_i = c_j, \mathcal{N}_{ij}^k) + \epsilon} \times \frac{n(c_i = c_j, \mathcal{D})}{n(c_i \neq c_j, \mathcal{D})}, \tag{4.19}$$

where $n(c_i \neq c_j, \mathcal{D})$ is the count of pairs of adjacent superpixels in $\mathcal{D}$ with $c_i \neq c_j$ (e.g. $c_i = c_1$ and $c_j = c_2$, or $c_i = c_2$ and $c_j = c_1$), respectively, $n(c_i = c_j, \mathcal{D})$ indicates the count when c_i and c_j are both equal to c_1 or c_2. Similar definitions are used for the count in $\mathcal{N}_{ij}^k$. In addition, a smoothing constant $\epsilon = 1.0$ is used to prevent zero likelihoods. For other cases of class combination, the same idea is applied.

To compute the neighbourhood $\mathcal{N}_{ij}^k$ for $\langle s_i, s_j \rangle$ under $\left|f_i^k - f_j^k\right|$, we first compute the absolute difference of the kth feature for all valid pairs of adjacent superpixels (i.e. both superpixels are valid with a nonempty label) in the retrieval set. After we get the threshold T using a similar approach as in Section 3.2, $\mathcal{N}_{ij}^k$ can be obtained by computing the distance from $\left|f_i^k - f_j^k\right|$ to each feature difference in the retrieval set. Note that

$\left|f_i^k - f_j^k\right|$ is regarded as a comparable value when computing the thresholds and distance.

With the above energy definitions for pairwise potentials, we choose to use graph-cut inference to solve our CRF model. As in the case of Tighe and Lazebnik's MRF formulation, multiplication by $\delta\left[c_i \neq c_j\right]$ in Equation 4.15 is required to ensure that this energy term is semi-metric for graph-cut inference. As in the MRF, the $\alpha\beta$-swap move algorithm successfully performs the CRF inference. Not like the smooth cost obtained from the the co-occurrence probabilities, it tend to be more costly to learn the pairwise cost in CRF. Since the CRF is defined on the superpixel graph, inference is efficient once we obtain the pairwise costs for each pair of neighbouring superpixels. We also have some results of our optimized labeling using the CRF inference shown in Figure 4.5.

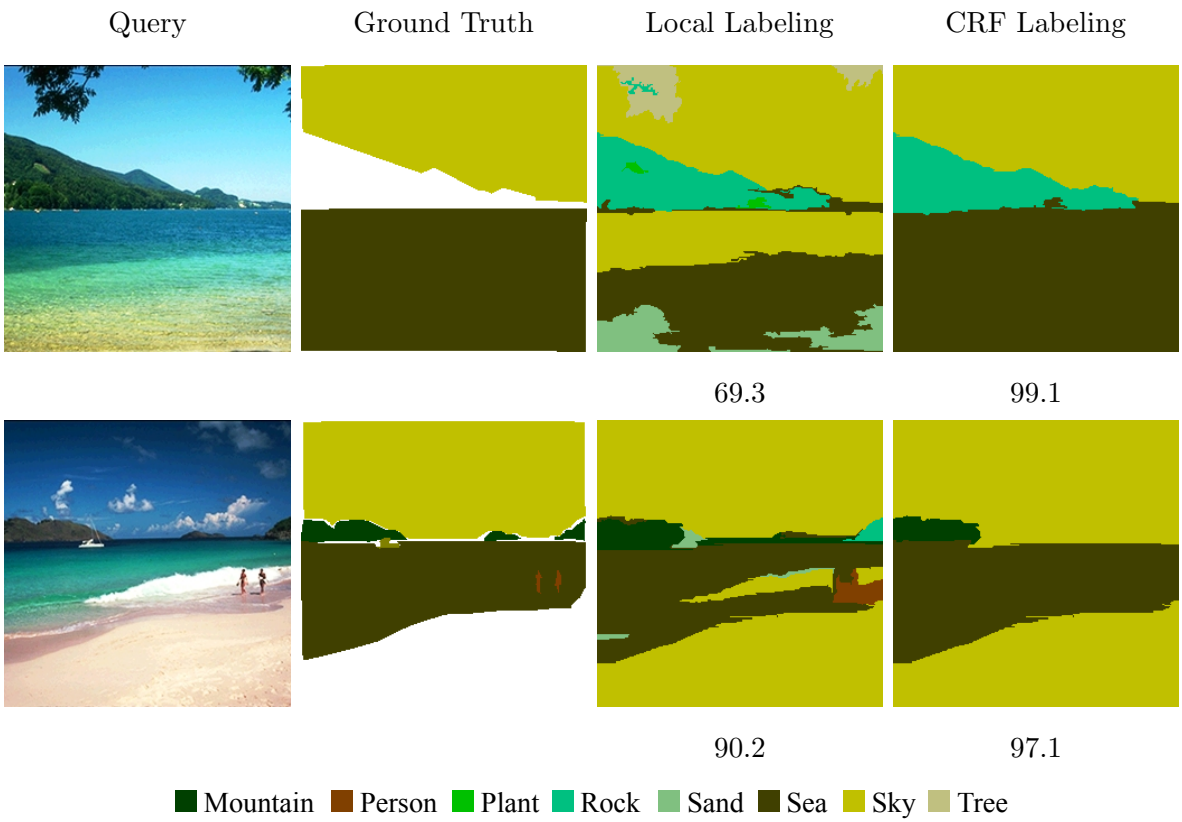

Figure 4.5: Comparing to ground truth labels (b), optimized CRF labeling (c) achieves per-pixel ratio equaling to 99.1%. Note the white area in (b) means its ground truth label is empty.

4.3 Summary

In this chapter, we have extended the superparing system to improve the segmentation boundaries. The SLICO segmentation method is applied first to obtain more superpixels with smaller regions. Then we have defined a pairwise cost depending on the similarity of neighbouring superpixels computed from local features, and have built a CRF model with contextual contraints provided by this pairwise cost. We have also explained the co-occurrece based MRF model proposed by Tighe and Lazebnik in this chapter. With these two approaches introduced, we will explain how they can affect image segmentation results using experiments in Chapter 5.

Chapter 5

Experimental Evaluation

In this chapter we present our results based on our method discussed Chapter 3 and Chapter 4 on the SIFT Flow dataset. For this dataset, we first give a basic description of this dataset and explain several measures used to evaluate our results in Section 5.1. Then we provide comparisons of results obtained with different segmentation approaches and different optimization methods in the next sections. In detail, Section 5.2 finds the best experimental results of local labeling though enumerating various parameters using the SLICO segmentation. In Section 5.3, we state and compare the optimized labeling results of our CRF method and the MRF optimization from Tighe and Lazebnik. Specially, the labeling results obtained using the graph-based segmentation method will be used as well as the best SLICO results obtained in Section 5.2. Section 5.4 dicusses the runtime of our modified system on the Sample Dataset we used in Chapter 3. We conclude with a discussion of our experimental results of our system in Section 5.5.

5.1 Dataset and Evaluation Measures

The "SIFT Flow dataset" [35] is a subset of the LabelMe database. The SIFT Flow dataset is composed of 2,688 images and all these images have been fully labeled by LabelMe users. Most of such annotated images are outdoor scenes such as beach, mountains, trees, street and buildings, but not covering indoor scenes or scenes mainly de-

scribed as people, animals and so on. Liu et al. have randomly split this dataset into 2488 "training" images and 200 test images with diffrenct types of scenes evenly. The top 33 object categories with the most labeled pixels have been chosen, leaving the pixels not labeled, or labeled as other object categories as the "unlabeled" category. These labeled images are regarded as ground truth. We use the same training/test split and 33 semantic labels as [35]. The top 33 object categories in the SIFT Flow dataset in order of label frequencies are: building, mountain, tree, sky, road, sea, car, field, window, plant, river, road, grass, sidewalk, sand, door, desert, bridge, person, balcony, fence, staircase awning, sign, crosswalk, boat, streetlight, bus, pole, sun, cow, bird, moon.

From the label frequencies of the training superpixels shown in [63], it is easy to tell that a few classes such as building, mountain, tree and sky are very common, but classes like boat, person, sign and pole are quite rare. To make it fair, Tighe and Lazebnik evaluate the labeling accuracy using both the per-pixel classification rate and the average of the per-class rates over all the classes. We keep these evaluation measures in our experiments. However, since we also aim to achieve meaningful segmentation results which can be used in, e.g., the haptic exploration system, we would like to use precision/recall values. Precision/recall are a quantitative measure of segmentation quality and we use the method of Estrada and Jepson [10] to evaluate the segmentation performance of our system.

Precision/Recall values compare two segmentations by characterizing the agreement between their region boundaries. Given a source segmentation S_1, and a target segmentation S_2, Estrada and Jepson define *precision* as the proportion of boundary pixels in S_1 for which a suitable matching boundary pixel was found in S_2 within a particular distance threshold:

$$Precision(S_1, S_2) = \frac{Matched(S_1, S_2)}{|S_1|} \tag{5.1}$$

where $|S_1|$ is the total number of boundary pixels in the source segmentation. Similarly, *recall* is defined as the proportion of pixels in S_2 for which a suitable match can be found in S_1:

$$Recall(S_1, S_2) = \frac{Matched(S_2, S_1)}{|S_2|} \tag{5.2}$$

As explained in [10], a low precision value is the result of significant over-segmentation,

or indicates that localization errors greater than the threshold occur to a large number of boundary pixels. To the contrary, recall is low when there is under-segmentation and it indicates a failure to capture salient image structure. An example of a segmentation boundary map and their corresponding precision/recall values are shown in Figure 5.1. The labeling results in the example are based on the graph-based segmentation method and from the Sample dataset introduced in Chapter 3. It is easy to see that there is an over-segmentation in the first image but the scene structure is captured well for the second. Note if all superpixels in an image are labeled as the same class, we will assign both precision and recall values for such image to 0. Thus, we use the precision/recall values to evaluate the segmentation accuracy of our system, together with the per-pixel rate and the average of the per-class rate for evaluating the labeling accuracy. We use two kinds of precision/recall measures for the complete test set, one is based on the median of precision while the other is based on the median of recall.

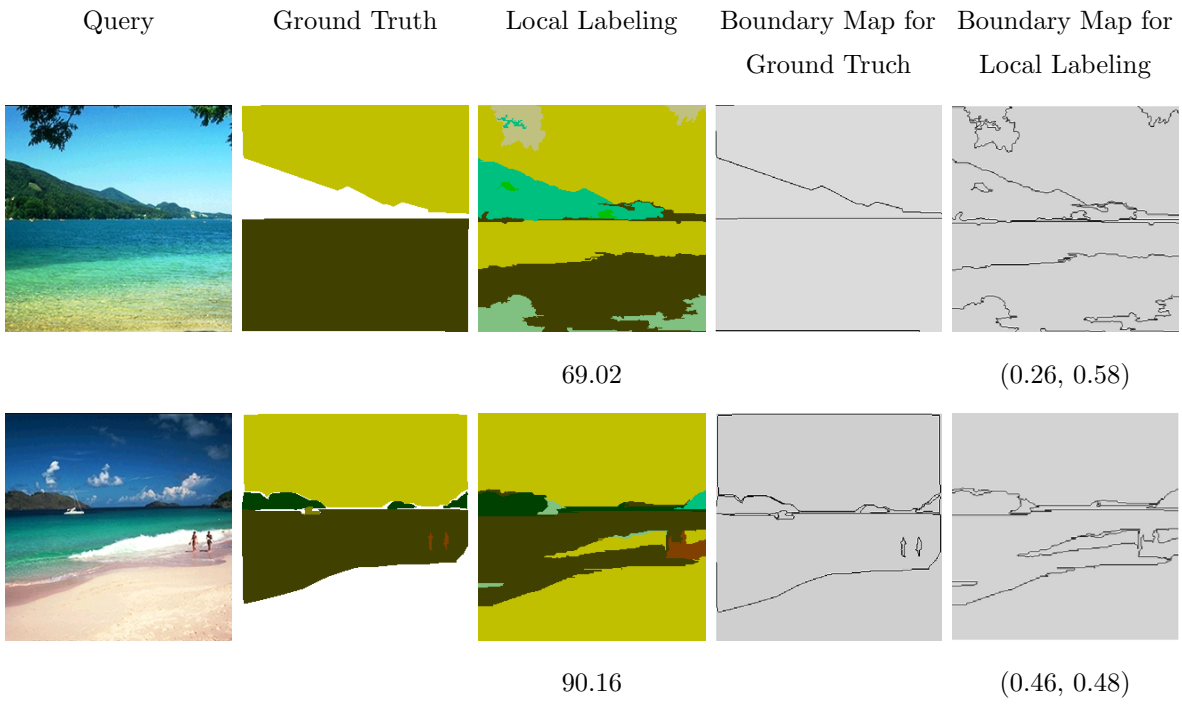

Figure 5.1: Comparison of boundary maps of local labeling and ones of the Ground Truth labels of the example images from the SampleDataset. The precision/recall value of the first image (0.26, 0.58) signals an over-segmentation, while (0.46, 0.48) in the second image indicates that the general scene structure is captured well

5.2 SLICO based Local Labeling Results

For the first modification of the basic superparsing system introduced in Section 4.1, we would like to know how the SLICO segmentation method can affect the labeling results through experiments. In our experiments, we choose to vary parameters of interest in the SLICO segmentation algorithm in order to produce appropriate superpixels for the local labeling step. The key parameter to do this is the number of superpixel, N_{SP}, which also determines the size of each superpixel. Increasing N_{SP} reduces superpixels' average area. When the average area are very large, it is easy to produce superpixels that include pixels from more than one class, so that some boundaries can never be recovered even if assigning them the same label in the superparsing. As N_{SP} is increased, SLICO generally captures boundaries more completely. However, if N_{SP} increased too far, it becomes more possible for small superpixels to be mislabeled because the internal consistency of larger area are broken and some superpixel features like shape don't work well any more. This will lead to a decrease of precision because of large number of small superpixels splitting large contiguous regions.

Here Figure 5.2 provides a visible comparison of results using different N_{SP} in SLICO segmentation for several images. The effect of N_{SP} can be directly seen. In Figure 5.2(a), the arches at the bottom of the building are plausibly classified as doors. And when the number of superpixels become larger, some of the windows are successfully labeled. However, as N_{SP} keeps increasing, many small regions are mislabeled as "sidewalk". On the other hand, big superpixels works well to avoid incorrect labels inside the "tree" category as shown in Figure 5.2(c). But it fails to classify "sky" between "tree" and "building" as well as the top part of "building". This is because superpixels are too large to accurately distinguish small regions of different categories. In Figure 5.2(d), some of "crosswalk" and "sidewalk" are successfully recovered as the area of each superpixel becomes smaller. This effect is more obvious for "crosswalk".

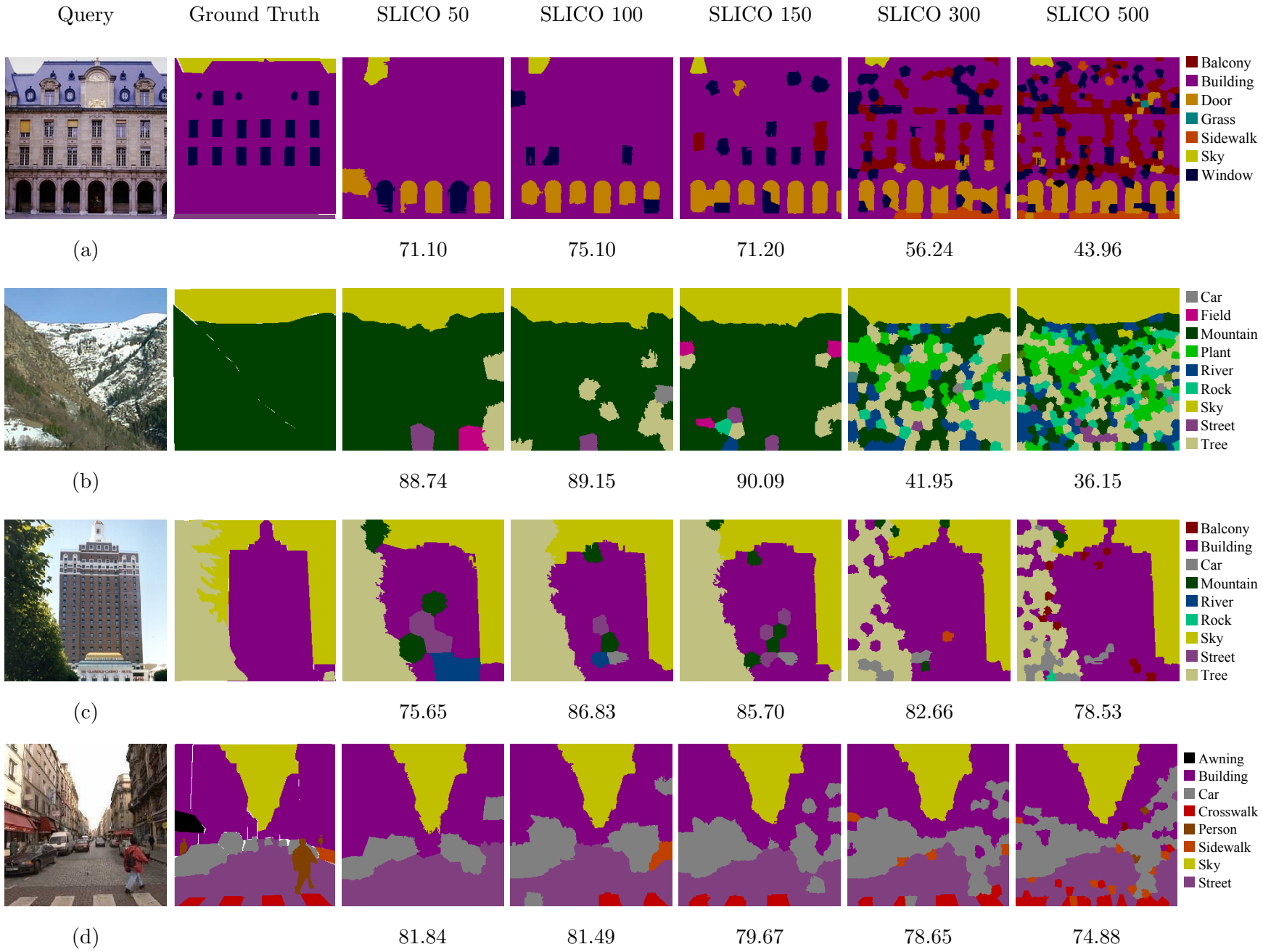

Figure 5.2: (©2015 IEEE) Example local labeling results from the SIFT Flow dataset using SLICO segmentation. Here we compare the effect of SLICO segmentation with different number of superpixels.

Except for parameters in the SLICO segmentation, the labeling results will also be hardly affected by the thresholds used to compute the log likelihood ratios defined in Section 3.3.2. In our implementation, we choose to use the same settings as Tighe and Lazebnik did in their work, the Tth nearest neighbour where $T = 80$ for all the experiments in this thesis. And we also have found that it gives very good results via examining various thresholds. We would like to point out, however, that the number of training superpixels has an influence in computing the thresholds. As the number of superpixels for each image increases, the total number of superpixels in the training dataset grows fast. The decision of how many superpixels to use in training becomes more important not only because of the computational cost, but also due to the very different performance of different choices. If the total number of superpixels used here is too large, the distance to the $80th$ nearest neighbour will become much smaller than the appropriate value which will result in a too small threashold obtained from the median distance. Then the threshold will be no longer effective to produce the expected results. Similarly, a too large threshold computed from a limited number of trainning superpixels will not work.

Table 5.1 examines the effect of the number of superpixels in SLICO segmentation on the local superpixel labeling with the per-pixel rate, average per-class rate and their corresponding precision/recall values over the SIFT Flow dataset. Interestingly, the performance drops drastically when increasing the number of superpixels. Thus, we quantitatively confirm that a larger number will not guarantee better results. Based on the comparisons of these results, we choose 100 as the number of superpixels in SLICO segmentation to illustrate the effect of CRF in the next section. As one can see, the average per-class rate increases as the number of superpixels becomes larger. We also observe that more rare classes have been classified due to smaller area of superpixels. However, similarly to the per-pixel rate, the per-pixel rate does not always get better when N_{SP} increases. It appears to slightly decrease with N_{SP} is 500, possibly because of too much mislabeling occurs.

We also report the precision/recall values for the local labeling results in Table 5.2. We have two pairs of such values for each case, one is selected based on the median of

N_{SP} in SLICO	Local Labeling
50	72.03 (22.97)
100	72.14 (26.66)
150	71.23 (29.99)
300	68.62 (30.83)
500	60.30 (29.24)

Table 5.1: (©2015 IEEE) Effect of the number of superpixels in SLICO segmentation on local superpixel labeling for the SIFT Flow dataset. Per-pixel classification rate is listed first, followed by the average per-class rate in parentheses.

N_{SP} in SLICO	Pre/Rec based on Median Precision	Pre/Rec based on Median Recall
50	(**0.30**, 0.15)	(0.40, **0.15**)
100	(**0.27**, 0.27)	(0.37, **0.19**)
150	(**0.26**, 0.17)	(0.18, **0.20**)
300	(**0.22**, 0.26)	(0.11, **0.25**)
500	(**0.19**, 0.27)	(0.15, **0.25**)

Table 5.2: (©2015 IEEE) The Precision/Recall Values for Local Labeling Results. The bold values show the trend for precision and recall, respectively. For an explanation, see Section 5.2 and 5.3.1.

precision and the other is selected based on the median of recall. The precision drops as the number of superpixels increases, resulting in more over-segmentation. But the increase of the recall indicates more boundaries are recovered in the labeling results. Note that the trend of the precision is primarily reflected by the median precision based values, while the median recall based values mainly shows the tendency of recall.

5.3 Comparisons of Optimized Labeling Results

The next important modification in our system is the optimization approach. To evaluate the effect of CRF inference we proposed in Section 4.2.3, we experiment with both

graph-based and SLICO segmentation methods for the SIFT Flow dataset. We also compare our CRF classification with the MRF formulation proposed by Tighe and Lazebnik to see how well both contextual inference methods can optimize the local labeling. To complete the contextual inference, an important parameter needed to be examined is the smoothing constant λ defined in Equation 4.11 and 4.14, which determines the contribution of the smooth cost. Usually the smooth cost will help improve the labeling results as λ increases at first. However, after a certain value, the effect of the data cost will become less domainent when λ becomes larger, causing the per-pixel rate to fall. Meanwhile, the average per-class rate will decrease by different degrees as a result of "smoothing away" of some small classes. In the MRF formulation, we use the same smoothing constant λ to balance the contribution of the smooth cost in all experiments. But for our CRF inference, we aim to find an appropriate λ to produce the best labeling results by balancing the per-pixel rate and the average per-class rate. In this section, we also provide a brief discussion as well as a demonstration of the labeling results.

5.3.1 Results based on Graph-based segmentation

At first, comparisons of labeling results from the CRF and MRF classifications using the graph-based segmentation are provided in Figure 5.3 and Figure 5.4. All the results here are depending on the local labeling produced by our reimplementation of the superparsing system explained in Chapter 3.

Figure 5.3 presents examples where contextual inference improves the initial semantic labeling based on Graph-based segmentation. In Figure 5.3(a), our CRF successfully smoothes "sky" on the top and corrects the middle region to "sea". Similarly, the CRF works better than the MRF in smoothing away small unexpected regions in Figure 5.3(d). Although"tree" and "mountain" are not able to be recovered, the primary countour of the scene structure keeps very well. In Figure 5.3(e), labeling based on Graph-based segmentation works well to distinguish "sky" between "tree" and "building" compared to results from SLICO shown in Figure 5.2. And for the optimization step, CRF works better than MRF for the "tree" category in this example.

λ/1000	Local	50	100	150	200	250
Per-pixel	71.04	72.92	73.63	73.72	73.67	73.30
Avg. per-class	29.62	28.22	26.50	24.36	22.91	22.49

Table 5.3: Performance of CRF based on graph-based segmentation by changing λ.

On the other hand, some other examples we would like to explain are shown in Figure 5.4. In the case of Figure 5.4(a) which primarily consisted of "tree", both CRF and MRF models perform well to correct the mislabeling. The CRF labeling gives a per-pixel rate of more than 99%, but it fails to capture the segmentation structure, resulting in serious under-segmentation (i.e., just a single segment with 0 precision/recall). For the segmentation measure, the choice of median precision/recall values helps with such cases.

As for Figure 5.4(b), the local labeling works very well to classify the rare "crosswalk" and "sidewalk" categories as well as the basic structure of the street view. However the contextual inference affects the labeling results somehow. The MRF smoothes "sidewalk" away unexpectedly and the CRF even removes all "crosswalk" resulting in a worse labeling results. This also gives an explanation to the drop of the average per-class rate reported in Table 5.4 while the per-pixel rate increases after the labeling optimization. That is, some of the smaller classes are "smoothed away" while the semantic CRF or MRF improves the labeling results.

Interestingly in Figure 5.4(c), both our CRF and the MRF based on our reimplementation remove the spurious classification of the suns reflection in the water compared to results reported in [63]. However, "mountain" is still not labeled out and "sun" disappears after the CRF inference.

The effect of the smoothing constant λ defined in the CRF formulation is shown in Table 5.3. Through experimenting with different λ values, the labeling results produced when λ equal to 150/1000 give the best per-pixel rate with an acceptable average per-class rate. Thus, λ is set to 150/1000 in the CRF classification for the graph-based

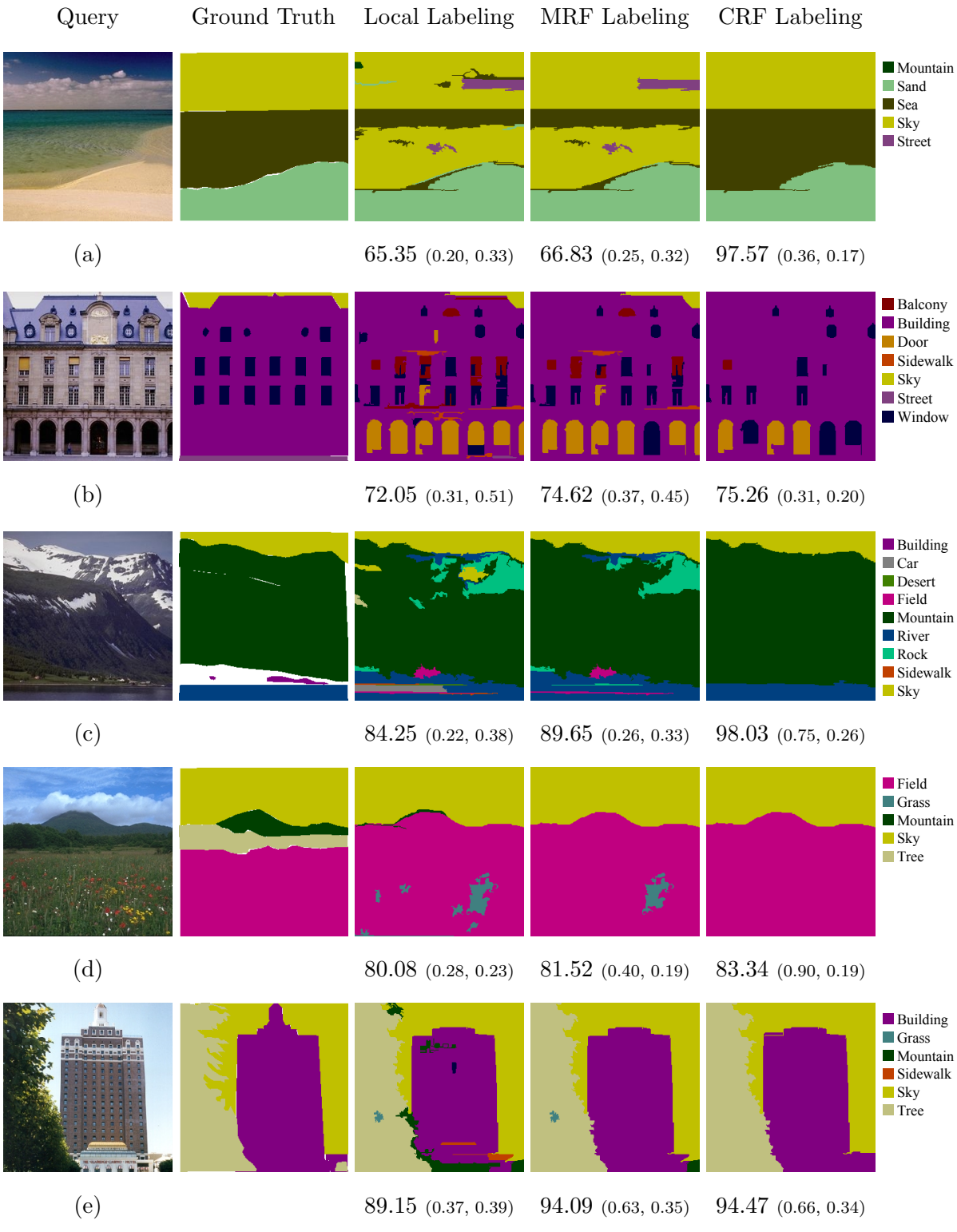

Figure 5.3: Example results from the SIFT Flow dataset for which Graph-based segmentation is successful. The number under each result image is the percentage of pixels labeled correctly and the precision/recall values of the segmentation.

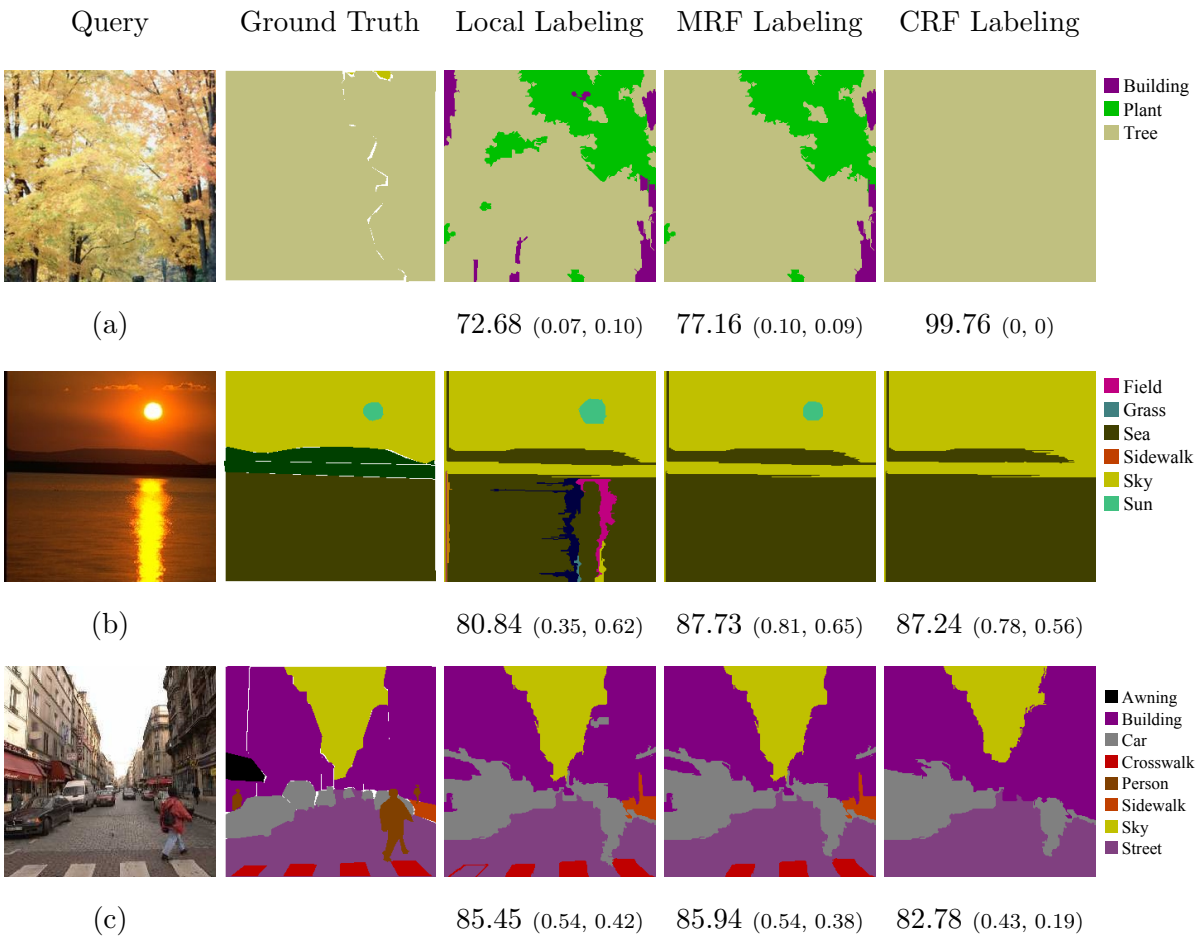

Figure 5.4: Some other example results from the SIFT Flow dataset based Graph-based segmentation.

	Per-pixel rate	Average per-class rate	Pre/Rec based on Median Precision	Pre/Rec based on Median Recall
Local labeling	71.04	29.62	(**0.31**, 0.35)	(0.48, **0.24**)
MRF	72.60	28.97	(**0.34**, 0.38)	(0.28, **0.22**)
CRF	73.72	24.36	(**0.36**, 0.19)	(0.46, **0.14**)

Table 5.4: Performance on the SIFT Flow Dataset Depending on Graph-based Segmentation. The bold values in precision/recall show the trend for precision and recall, respectively.

segmentation in our experiments. The CRF labeling results in Figure 5.3 and Figure 5.4 are generated using this value.

Table 5.4 shows the per-pixel and average per-class rates for local superpixel labeling, re-implemented MRF and our novel CRF on the SIFT Flow dataset. As compared to the local baseline, the contextual CRF improves overall per-pixel rates by about 2.7 percentage points while MRF improves by about 1.6 percentage points. The average per-class rate drops after contextual inference due to smoothing away some smaller classes. This is more obvious in CRF labeling when its per-pixel rate gets higher. As some rare classes are smoothed away and some of smaller superpixels are classified to labels of their neighbouring superpixels, the boundary maps of the optimized labeling (such as Figure 5.1) become smoother. This is also reflected by the drop of recall value. Hence, after the optimization step, under-segmentation gets worse while over-segmentation becomes less obvious as reflected in the increase of the precision values. The trend is more obvious for CRF compared to MRF.

5.3.2 Results based on SLICO segmentation

We now report a comparison of labeling results from the MRF and CRF inference based on SLICO segmentation method. Figure 5.5 presents the labeling results on the SIFT Flow dataset produced by our modified superparsing system which combines the SLICO

λ/1000	Local	50	100	150	200	250
Per-pixel	72.14	74.76	75.61	75.54	76.18	76.31
Avg. per-class	22.66	24.48	23.79	23.29	23.59	23.26

Table 5.5: (©2015 IEEE) Performance of CRF based on SLICO segmentation by changing λ.

segmentation and the optimization classification methods. As explained in Section 5.2, the local labeling based on SLICO where the number of superpxiels $N_{SP} = 100$ will be used here for the comparison because of its good performance. On the other hand, Table 5.5 examines the smoothing constant λ similar to Table 5.3. Although 250/1000 gives the maximum per-pixel rate, we use 200/1000 as the value of λ which provide a higher average per-class rate as well as a very high per-pixel rate. Based on above configuration, the labeling results are produced and demonstrated in Figure 5.5.

Most query images in Figure 5.5 are the same examples from previous sections to help the direct comparison of the performance of our system under different situations using the same examples. Figure 5.5 shows the effect and difference of both contextual inference over local labeling with the SLICO segmentation. The examples in Figure 5.5 are quite representative and describe the characteristics of SLICO based classification well.

In Figure 5.5(a), we see that many small superpixels are mislabeled in the regions of "sky" and "sea". Not like its graph-based local labeling in Figure 5.4, these small areas are easy to be broken. Since the MRF smoothes labeling using the label co-occurrence of neighbouring superpixels, the "field", "door" and "bridge" are successfully corrected because of their low co-occurrence with "sea" in the SIFT Flow dataset. But the dislocated "sky" and "sea" keep unchanged as well as "building" under this classification. In contrast, our CRF analyzes the similarity of neighbouring superpixels depending on their local features. Hence, the mislabeling are smoothed easily by CRF while the sun is unfortunately removed.

For Figure 5.5(b), the per-pixel rate slightly drops after the MRF smoothes away a

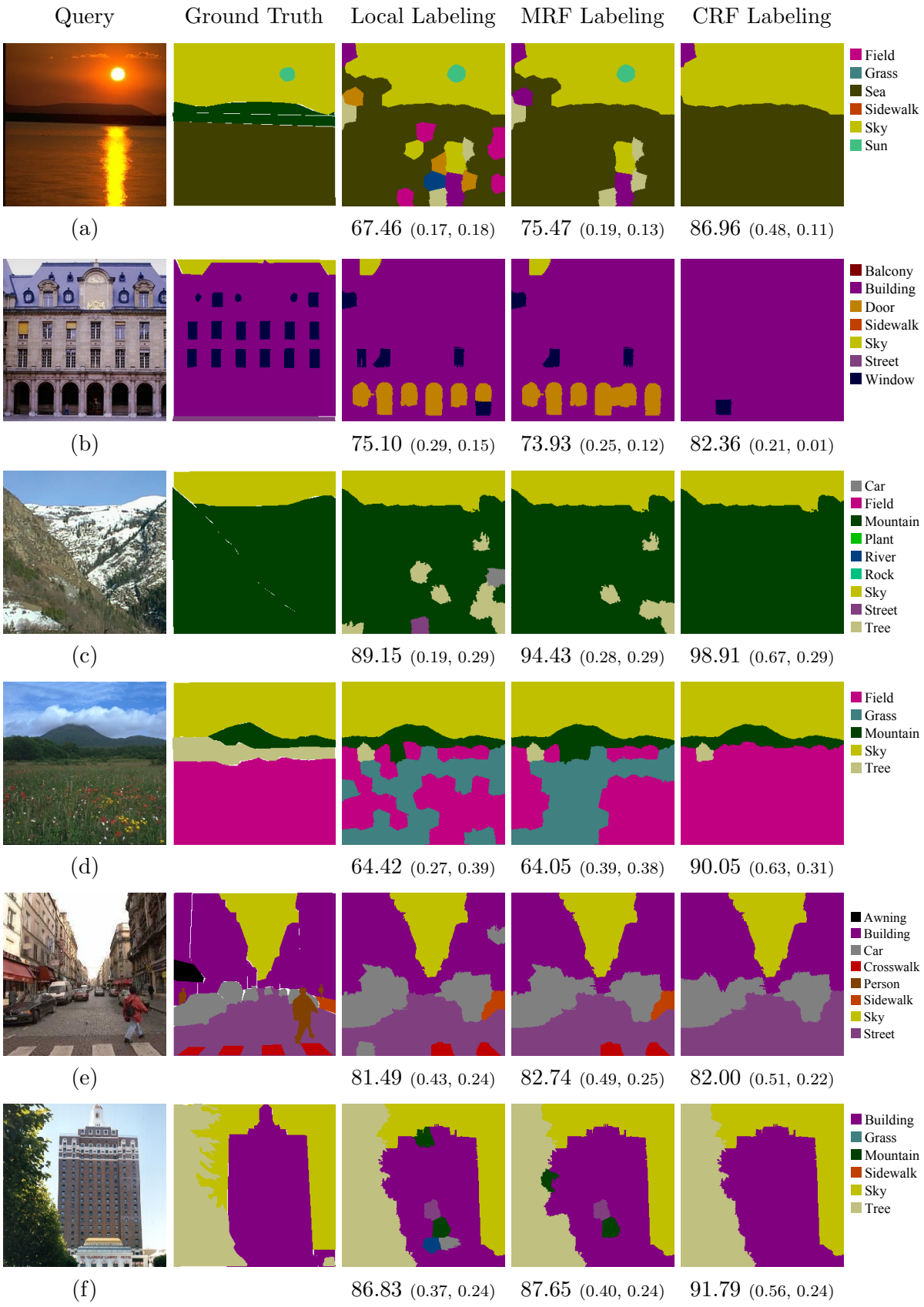

Figure 5.5: (©2015 IEEE) Results based on SLICO 100.

necessary "window" in the middle left. However in the CRF classification, all relatively small classes in the scene are removed, leaving an empty "building" together with an unexpected "window". Even the per-pixel rate grows, it is over-smoothed by the CRF and such segmentation results are very implausible. Similar to the first example in Figure 5.4.

As explained in Section 5.2, the local labeling of Figure 5.5(e) fails to distinguish "sky" between "tree" and "building" because superpixels in that area cover multiple labels. The contextual inference works to correct small mislabeled regions inside "building", but still cannot correct for the falsely labeled sky regions.

According to the per-pixel rate used as the primary measure to evaluate the performance of our system, the CRF labeling improves by some degree for all examples in Figure 5.5 compared to the local labeling. In many of the remaining test images we can observe the same situation. But from a classification perspective, it is not good enough to classify relatively rare classes. On the contrary, a lot of small regions labeled with rare classes are assigned to classes of neighbouring superpixels and smoothed away by the CRF inference. As for segmentation, the serious under-segmentation in some results due to over-smoothing is extremely implausible. In general, a labeling result with only one class cannot be regarded as a meaningful segmentation.

The problems above seem to be the disadvantages of aiming to find the possibly highest per-pixel rate. In order to do this, we need to improve the "smoothing" ability of the CRF to correct unexpected mislabeling. The λ defined in Section 4.2.3 is used to control the contribution of the smoothing term. However, increasing λ also has direct impact on many of the correctly classified small classes. The drop of the average per-class rate illustrated in Table 5.6 also reflects the "smoothing away" of smaller classes.

Similarly to Table 5.4, the performance of our system on the SIFT Flow dataset using SLICO segmentation is shown in Table 5.6. The per-pixel rate and the average per-class rate of local labeling, re-implemented MRF and our CRF are illustrated separately. The per-pixel rate using the MRF compared to the local baseline is improved by

SLICO ($N_{SP} = 100$)	Per-pixel rate	Average per-class rate	Pre/Rec based on Median Precision	Pre/Rec based on Median Recall
Local labeling	72.14	26.66	(**0.27**, 0.27)	(0.37, **0.19**)
MRF	73.94	26.48	(**0.32**, 0.20)	(0.30, **0.17**)
CRF	76.18	23.59	(**0.39**, 0.15)	(0.34, **0.12**)

Table 5.6: (©2015 IEEE) Performance on the SIFT Flow dataset depending on SLICO segmentation with number of superpixels equal to 100. Note that the bold values in precision/recall show the trend for precision and recall, respectively.

about 1.8 percentage points. In contrast, this per-pixel rate has been improved about 4.0 percentage points by our CRF inference. Compared to the performance based on the graph-based segmentation, the effect of the CRF optimization is more apparent here. However, there are still many of the small classes being smoothed away as one can tell from the drop of the average per-class rate. But the gap between the local labeling and the CRF becomes smaller, which relieves the effect of "smoothing away" of rare classes a little. As the median precision value keeps increasing, there are more boundaries covered after the contextual inference. Of course, the median recall value drops because of similar reasons. Some of the correct boundaries are removed as the disappearance of some smaller classes and incorrect labeling.

For the comparison of the labeling results from our modified system and the original superparsing approach of Tighe and Lazibnik, Figure 5.6 to Figure 5.8 provide some extra example results based on our reimplementation. We would like to discuss more about how the ground truth may affect the labeling resutls. Below is summary of effects of errors and ambiguities in the ground truth on our labelling resutls.

- Missing labels will impact the labeling results in the nonparametric density estimates, especially, if a query superpixel matches well with a non-labeled superpixel in the retrieval set.
- The incorrect ground truth may also give false per-pixel rates even if regions are correctly indentified. For example, the first example in Figure 3.4 shows that the

Type	N_{SP} in SLICO	50	100	150	300	500	Graph-based
(1)	Feature extraction	30.04	36.35	38.15	43.63	49.87	34.82
	Indexing	7.26	15.00	21.23	41.56	67.78	6.87
(2)	Flann tree construction	3.45	3.40	3.42	3.42	3.44	3.46
	Thresholds (Local)	114.3	237.1	342.1	683.3	1061.4	86.2
	Thresholds (CRF)	216.1	494.1	746.8	1571.8	2650.6	137.6
(3)	Retrieval set search	0.40	0.40	0.39	0.39	0.40	0.40
	Local labeling	4.73	14.62	33.04	131.07	413.29	4.88
	CRF/MRF solver	13.74/2.13	59.61/2.02	128.34/2.62	535.17/5.00	1673.02/7.94	9.99/2.12
	Total (CRF/MRF)	18.87/7.26	74.63/22.06	161.77/36.05	666.63/136.46	2086.71/421.63	15.27/7.4

Table 5.7: (©2015 IEEE) Comparison of the run time of different stages in seconds on the Sample dataset using different N_{SP} in SLICO segmentation (including file I/O). Note Type (1) is the average for the whole dataset (i.e. 100 images), while Type (3) is the average of the query images (i.e. 5 images). Type (2) is the total time for computing the thresholds over the whole data set. Type (1) and (2) only need to be computed once.

tree in the sky area are ignored in the ground truth.

- Sometimes multiple labels may be arguably equally correct. For example, in Figure 5.5(c), some of mountain area could also be classified as "tree" or "rock", but only "mountain" is accepted according to the ground truth.

- Superpixels may also be identified as different categories under different scales. As shown in Figure 5.7, our labeling resutls successfully lebel out some windows on the building. Again, only "building" is correct when compared with the ground truth.

5.4 Runtime Discussion

We currently evaluate the runtime of our system on the Sample data set used in Chapter 3 and Chapter 4, which is part of the SIFT Flow dataset and consists of 95 training images and 5 test images. All the tests here are executed single-threaded on a PC with Intel Core i7 CPU running at 2.67 GHz and 12 GB RAM. Table 5.7 shows a breakdown of the main stages of the computation.

There are three types of computation in our system and they are clearly categorized

in Table 5.7. The first two types only need to be computed once for each configuration of the system. In Type (1), the computing time is illustrated as the average for the whole dataset (i.e. 100 images), and the stage of feature extraction includes both global and local features. Type (2) is the total time for building a flann tree to search the retrieval set and computing the thresholds over the whole dataset respectively. Having the training data prepared by stages in the first two types, Type (3) shows the average time for a test image to complete the query process in the image parsing system, while the "Total" just represents the total time of stages in this type without feature extraction for the query image. We also show the time cost for the MRF inference to make a comparison.

The thresholds computation in local labeling is necessary for both MRF and CRF solvers. The main difference comes from the threshold preparation for the pairwise cost as well as the CRF inference. As the number of superpixels increases, the computation in CRF increases more and the difference becomes more obvious. However, by limiting the total number of superpixels in the training set used to compute the thresholds, it helps limit the computation. This constraint will be more obvious and helpful for a larger training dataset.

On the other hand, there are only 95 training images in this dataset and all of them are included in the retrieval set of each test images. However, we usually use 200 as the size of the retrieval set in a larger dataset, which indicates a more complex computation for searching the nearest neighbours. Besides, more training superpixels in the retrieval set will make it longer to match superpixels and compute likelihoods for the CRF inference.

5.5 Summary

Based on the results represented in this chapter, we find that the proposed CRF inference works very well in improving the overall per-pixel rate of the labeling results, compared to the MRF method from Tighe and Lazibnik. But it is also easy to see that the average

N_{SP} SLICO	50	**100**	150	300	500	**Graph-based**
Local	72.03(22.97)	**72.14(26.66)**	71.23(29.99)	68.62(30.83)	60.30(29.24)	**71.04(29.62)**
MRF	73.31(22.91)	**73.94(26.38)**	73.34(30.34)	71.13(31.13)	62.57(29.49)	**72.60(28.97)**
CRF	74.61(21.36)	**76.18(23.59)**	75.49(23.48)	73.81(25.29)	N/A	**73.72(24.36)**

Table 5.8: (©2015 IEEE) Comparisons of performance on the SIFT Flow dataset between the graph-based and SLICO segmentation at various number of superpixels. The per-pixel rate is followed by the average per-class rate in the parentheses. Note that the bold values are the best labeling results in each case, and the CRF labeling for $N_{SP} = 500$ is too costly for our implementation setup to compute.

per-class rate will drop as the local labeling being more smoothed by the CRF as well as the MRF, which indicates the "smoothing away" of some small classes.

Compared with the graph-based segmentation method, the performance of our system using SLICO segmentation with various number of superpixels is reported in Table 5.8. The best per-pixel rate is achieved with an acceptable average per-class rate when the number of superpixels equal to 100. It also explains our decision taken in the design of our method, which discussed in Section 5.2 and 5.3.2. The precision/recall values also reflect the trend in the aspect of segmentation evaluation. Furthermore, the variation tendency of the per-class rate indicates a potential to keep more rare classes after labeling optimization when N_{SP} becomes larger.

On the other hand, this improvement made by our novel CRF comes at a computational cost higher than that of the MRF due to computing the similarity of neighbouring superpixels based on their local features.

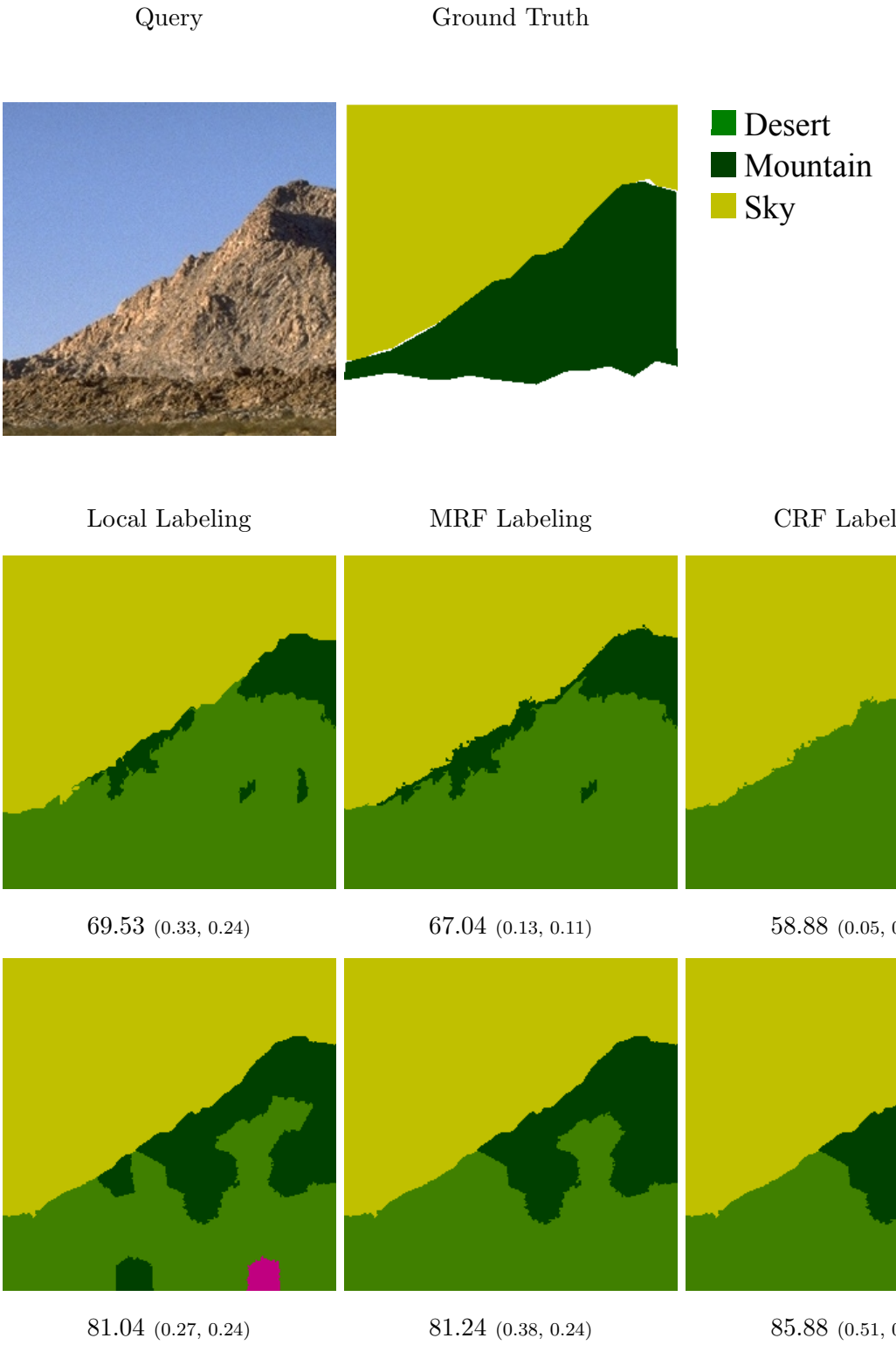

Figure 5.6: More example results 1, from the SIFT Flow dataset.

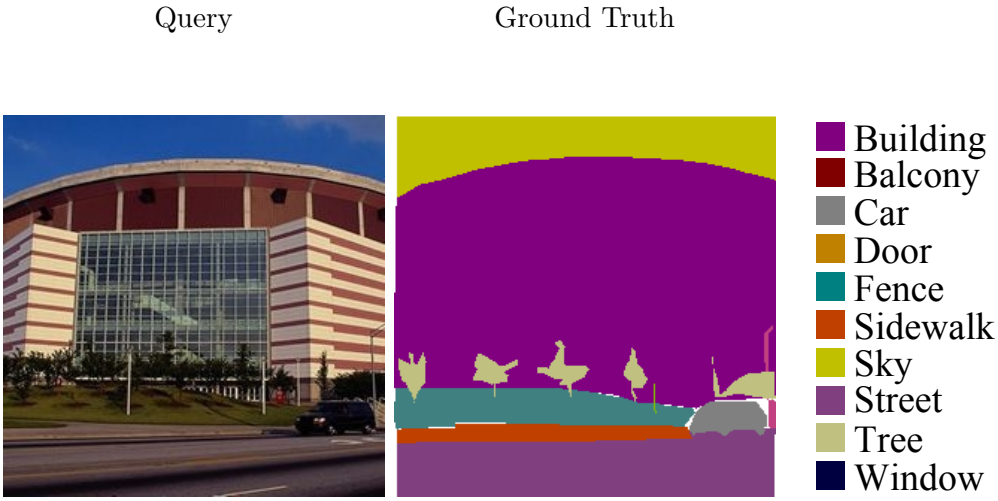

Figure 5.7: More example results 2, from the SIFT Flow dataset.

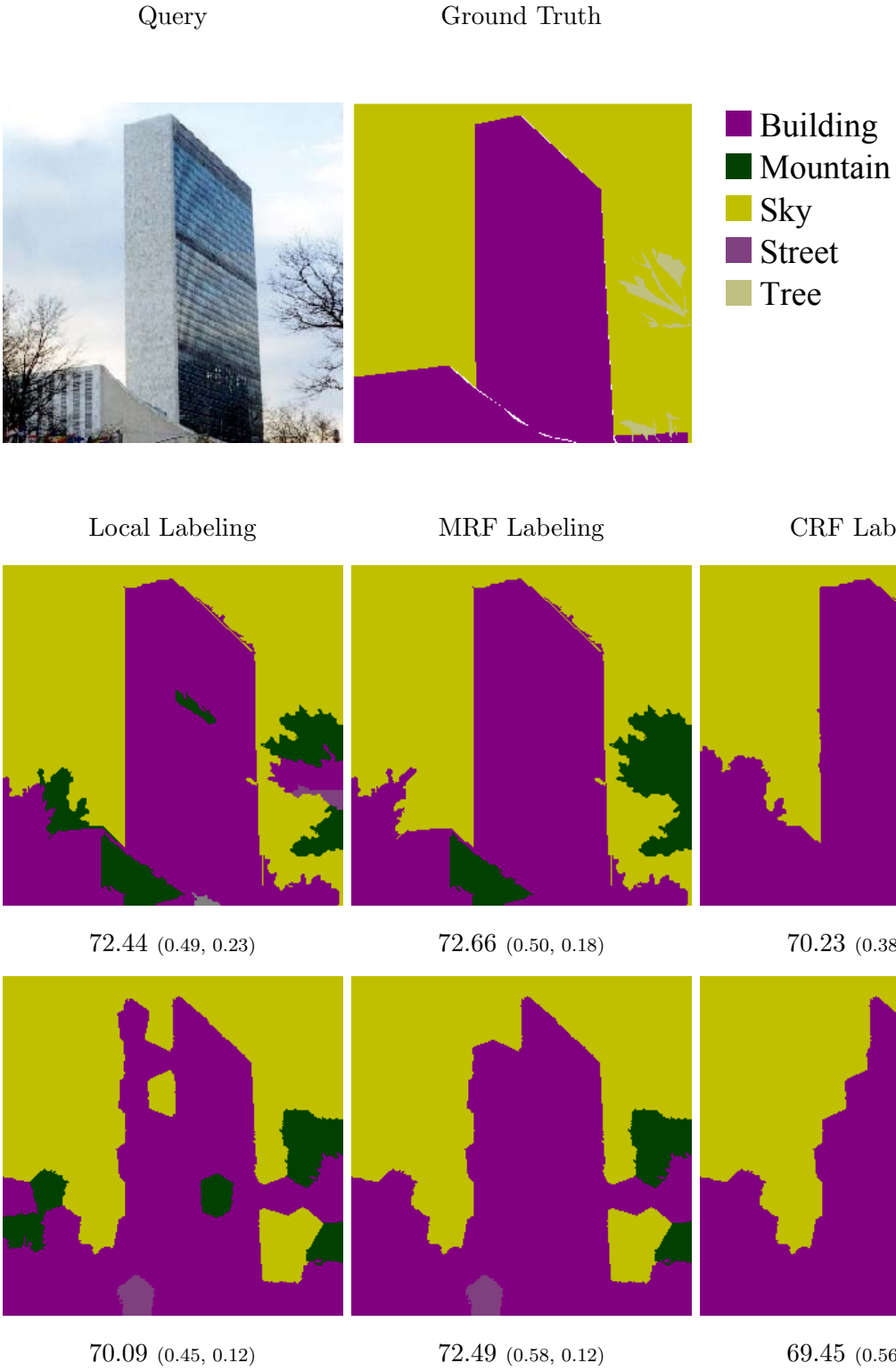

Figure 5.8: More example results 3, from the SIFT Flow dataset.

Chapter 6

Conclusion

From the outset of this work, our goal has been to produce meaningful object-level segmentations through the image parsing approach, and to improve labeling results at the same time. We have extened the retrieval-based superparsing system proposed by Tighe and Lazebnik [63] using the SLICO segmentation method, and proposed a new CRF inference to improve the local labeling results. We have shown the success of our approach, which improves the labeling results as well as the segmentation quality at the object level over the SIFT Flow dataset. Our novel CRF inference produces better results than the MRF formulation in optimizing initial labeling in terms of per-pixel rate. In this chapter, we provide a summary of our work, discuss the limitation of the proposed method and the original superparsing approach in general, and propose future work to further enhance our system.

6.1 Summary

Our goal in extending the basic superparsing system is to improve the systems ability produce segmentations with more meaningful and accurate boundaries as well as to produce better labeling results. To this end, we modify both segmentation method and contextual inference. Since different contextual inferences will improve the labeling results by different degrees, we devote significant effort to our proposed CRF classification

method.

We review the existing relevant literature, including various categories of image segmentation methods, several algorithms used to extract global and local features, other image parsing methods based on random fields, different contextual contraints in image parsing and a brief review of evaluation measures for segmentation quality. The third chapter describes the framework of the basic superparsing system of Tighe and Lazebnik, and pointes out the general modifications we have made to adjust the system to the C++ environment, including the use of the FLANN algorithm [45] in the retrieval set search.

In Chapter 4, we extend the basic superparsing system by modifying the segmentation method. In order to capture more precise boundaries of the query images, we use SLICO segmentation to generate more superpixels with smaller regions. This follows a reasoned consideration of the different characteristics between graph-based and SLICO segmentations. We introduce a novel CRF inference for the labeling optimization process based on the information of adjacent superpixels. Compared to the probabilities of label co-occurrence used in the MRF classification, our CRF inference corrects and smoothes the local labeling by computing the similarity of neighbouring supperpixels with the help of training their local features. The solution of the CRF using the graph-cut algorithm is also adjusted to any possible combination of labels in the dataset other than that of the used labels in the initial labeling only. With this modification, the correction of mislabeling of small regions like “sea” completely surrounded by “sky” improves a lot.

Finally, we demonstrate the labeling results of our system for the SIFT Flow dataset, and compare the performance of our proposed approach with the superparsing pipeline from Tighe and Lazebnik based on our reimplementation in various aspects. Our modified method is able to produce better results with a higher per-pixel rate and an acceptable average per-class rate. The trend of precision/recall also indicates an improvement in segmentation after contextual inference. We also evaluate the runtime of our system.

6.2 Conclusions

We have achieved an improvement in segmentaion boundaries by adapting and extending the superparsing system of Tighe and Lazebnik in our thesis. The primary contributions of our work cover two aspects. For the superpixels generation, the SLICO method is used to achieve more accurate boundaries of segmentations by controlling the size, shape and compactness of superpxiels. In the contextual inference, a Conditional Random Field (CRF) though a novel nonparametric feature-based pairwise constraint is proposed to optimize and smooth the local labeling very well. In this way, the segmentations are grouped and corrected with improved boundaries as well.

However, there are still limitations of the proposed approach. Through the demonstration and discussion of the labeling results in Chapter 5, we have identified three major limitations of our modified system. Some of them cannot be avoided in the current framework, but some of them are candidates for further work.

The first limitation is regarding superpixels produced by automatic segmentation algorithms and the recognition of small classes. If the region of a superpixel is too large and contains pixels with multiple labels, there will be no way to fix the label of such a superpixel by assigning it to any single class. Hence, we choose to use the SLICO segmentation to generate more superpixels with smaller areas, instead of the graph-based segmentation with uncontrolled irregular superpixels. However, since the size of superpixels are similar to each other in the SLICO segmentation, the local features such as "shape" and "size" will be no longer effective in classification decision. In addition, there still exist superpixels covering more than one class even in SLICO segmentation. This will occur less often when the superpixels become smaller and smaller. But as we keep increasing the number of superpixels in the SLICO segmentation, mislabeling of smaller regions will be more serious due to loss of constraints from local features. To recover such mislabelings, our CRF plays a significant role in removing small incorrect labels because of its powerful smoothing ability. However, it also results in the failure to recognize small and important classes in the original images.

Second, the classes in the SIFT Flow dataset are limited. It only covers some outdoor categories like "building", "mountain", "tree", "sky" and so on, which results in a limited evaluation not general enough to recover more complex environments. In other words, the dataset limits the ability to recognize object categories and causes incorrect labels.

Next as discussed in Section 5.4, the speed of our system really suffers from computing the thresholds, and the time for training the pairwise cost in the CRF labeling as the number of superpixels in each image increases and as the number of images in the training dataset increases. Thats why we didnt apply the CRF labeling for the case when $N_{SP} = 500$ in SLICO. This imposes limits on experiments with larger data sets and increasing the number of superpixels and images will have to be addressed in future work.

6.3 Future Work

The following are several directions of interest for improvements or enhancement to our modified superparsing system:

- **Parameter Optimization:** Our proposed method has no parameters exposed to end users, but there are multiple parameters involved in our approach, which will influence the performance of our system. Thus, an adjustment is necessary for some of them in our design. As such, we would like to perform an optimization of our parameters not only for the number of super pixels, N_{SP}, in the SLICO segmentation and the smoothing constant, λ, in the CRF inference (as discussed in Section 5.3). Furthermore, the thresholds used to compute the data cost and pairwise cost for the CRF could be adjusted via experiments in various cases. And we could also change the retrieval set size to optimize the labeling results based on the SLICO segmentation method.

- **Local Feature Adjustment:** Currently, we describe the appearance of superpixels using 20 different local features including "shape", "size", "location", "color",

"texture" and so on. These features work very well in matching irregular superpixels with various sizes produced by the graph-based segmentation method. However, in SLICO segmentation, superpixels usually have similar shapes and sizes, which make the "shape" and "size" features not effective in superpixel search. A possible solution is to adapt local features for SLICO segmentation by reducing local features related to "shape" and "size" categories and replacing them with features better suitable, e.g. increasing the weight of "texture". Or we could also just simply remove such features. This would also be helpful to overcome the major bottleneck in our system, the resources required to complete the CRF labeling.

- **Superpixel Generation:** The graph-based and SLICO segmentation methods have been used in our implementation so far. Each of them has their pros and cons in producing superpixels used in the labeling process. Hence, we would be interested in exploring the effect of some other segmentation algorithms on our system. Various segmentation methods have been proposed as we reviewed in Section 2.4.

- **CRF Inference with Spatial Information:** Our current CRF inference work focuses on neighbourhood relationships which may be considered as semantic context. It corrects mislabeling depending on the similarity of neighbouring superpxiels, which is computed based on the local features. However, there are still some obvious mislabels, e.g., "sky" under "sea" in the optimized results. A useful enhancement of the proposed CRF would be the integration of the spatial information of superpixels, providing further disambiguation of relative object locations. As work by Galleguillos et al. [19], we could incorporate spatial relationships above, below, inside and around for the database to enforce spatial contextual constraints on the image labeling. On the other hand, we could also consider the idea from Tighe and Lazibnik [63], combining inference over semantic and geometric labels to improve the accuracy of the semantic labeling. But we need to notice that there are only three geometric labels ("sky", "horizontal" and "vertical") used in their method, which will result in an unclear description of some spatial relationships like inside and around. Besides, it may be too restrictive that each semantic class is assumed to associate with a unique geometric class (e.g., "rock" is forced to be

"vertical").

- **Evaluation on other Databases:** Now we evaluate the performance of our system on the SIFT Flow dataset. It would be worthwhile to extend our experiments to some other database to obtain more general results. Two good candidates would be MSRC-21 [60] and PASCAL VOC 2008 [11]. The MSRC segmentation dataset contains 591 images of resolution 320 x 213 pixels, accompanied with a hand labeled object segmentation of 21 object class including "animal", "house", "person", "tree", "vehicle" and so on. Pixels on the boundaries of objects are unlabeled in these segmentations. The PASCAL VOC 2008 was used for the the PASCAL Visual Object Class Challenge 2008. It contains 511 training, 512 validation, 512 test images of 20 foreground and 1 background classes. Since the test data annotation in PASCAL VOC 2008 is not public, we could change to use the validation images with ground truth segmentation as the test images for our system. In addition, it would also be possible to consider the "LM+SUN" dataset [63] with more classes for our future experiments, once we find a good solution for the bottleneck in the computation.

- **Further Applications:** So far, we have obtained meaningful object-level segmentations with our modified superparsing system. We could adapt these image segmentation results for further applications, e.g., building an object-level segmentation hierarchy for the haptic exploration system.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

[2] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. From contours to regions: An empirical evaluation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2294–2301. IEEE, 2009.

[3] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[4] Pablo Arbelaez, Jordi Pont-Tuset, Jon Barron, Ferran Marques, and Jitendra Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 328–335. IEEE, 2014.

[5] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[6] Christopher M Bishop. Neural networks for pattern recognition. 1995.

[7] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1124–1137, 2004.

[8] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[9] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

[10] Francisco J Estrada and Allan D Jepson. Quantitative evaluation of a novel image segmentation algorithm. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 1132–1139. IEEE, 2005.

[11] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html.

[12] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[13] Lester Randolph Ford and Delbert Ray Fulkerson. *Flows in networks*, volume 1962. Princeton Princeton University Press, 1962.

[14] Jordi Freixenet, Xavier Muñoz, David Raba, Joan Martí, and Xavier Cufí. Yet another survey on image segmentation: Region and boundary information integration. In *Computer Vision ECCV 2002*, pages 408–422. Springer, 2002.

[15] Jordi Freixenet, Xavier Muñoz, David Raba, Joan Martí, and Xavier Cufí. Yet another survey on image segmentation: Region and boundary information integration. In *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002, Proceedings, Part III*, pages 408–422, 2002.

[16] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670–677. IEEE, 2009.

[17] Carolina Galleguillos and Serge Belongie. Context based object categorization: A critical survey. *Computer Vision and Image Understanding*, 114(6):712–722, 2010.

[18] Carolina Galleguillos, Brian McFee, Serge Belongie, and Gert Lanckriet. Multi-class object localization by combining local contextual interactions. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 113–120. IEEE, 2010.

[19] Carolina Galleguillos, Andrew Rabinovich, and Serge Belongie. Object categorization using co-occurrence, location and appearance. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[20] Robert M Haralick and Linda G Shapiro. Image segmentation techniques. In *1985 Technical Symposium East*, pages 2–9. International Society for Optics and Photonics, 1985.

[21] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*, pages 297–312. Springer, 2014.

[22] James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[23] Xuming He, Richard S Zemel, and MA Carreira-Perpindn. Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2, pages II–695. IEEE, 2004.

[24] Geremy Heitz and Daphne Koller. Learning spatial context: Using stuff to find things. In *Computer Vision–ECCV 2008*, pages 30–43. Springer, 2008.

[25] Derek Hoiem, Alexei A Efros, and Martial Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.

[26] Peter L Ivănescu. Some network flow problems solved with pseudo-boolean programming. *Operations Research*, 13(3):388–399, 1965.

[27] Anil K Jain and Aditya Vailaya. Image retrieval using color and shape. *Pattern recognition*, 29(8):1233–1244, 1996.

[28] Pushmeet Kohli and Philip HS Torr. Efficiently solving dynamic markov random fields using graph cuts. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, pages 922–929. IEEE, 2005.

[29] Lubor Ladicky, Christopher Russell, Pushmeet Kohli, and Philip HS Torr. Associative hierarchical crfs for object class image segmentation. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 739–746. IEEE, 2009.

[30] Lubor Ladicky, Paul Sturgess, Karteek Alahari, Christopher Russell, and Philip H. S. Torr. What, where and how many? combining object detectors and crfs. In *ECCV (4)'10*, pages 424–437, 2010.

[31] David Lareau and Jochen Lang. Instrument for haptic image exploration. *IEEE T. Instrumentation and Measurement*, 63(1):35–45, 2014.

[32] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[33] Alex Levinshtein, Adrian Stere, Kiriakos N Kutulakos, David J Fleet, Sven J Dickinson, and Kaleem Siddiqi. Turbopixels: Fast superpixels using geometric flows. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(12):2290–2297, 2009.

[34] Stan Z Li. Markov random field models in computer vision. In *Computer Vision?ECCV'94*, pages 361–370. Springer, 1994.

[35] Ce Liu, Jenny Yuen, and Antonio Torralba. Nonparametric scene parsing via label transfer. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2368–2382, 2011.

[36] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, 2011.

[37] Ming-Yu Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 2097–2104, Washington, DC, USA, 2011. IEEE Computer Society.

[38] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[39] Tomasz Malisiewicz and Alexei A Efros. Recognition by association via learning per-exemplar distances. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[40] David R. Martin, Charless C. Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, may 2004.

[41] David Royal Martin. *An Empirical Approach to Grouping and Segmentation*. PhD thesis, EECS Department, University of California, Berkeley, Aug 2003.

[42] Fernando Martinelli. Scene layout segmentation of traffic environments using a conditional random field. Master's thesis, Girona, 2010.

[43] Marina Meila. Comparing clusterings: an axiomatic view. In *Proceedings of the 22nd international conference on Machine Learning (ICML-05)*, pages 577–584, 2005.

[44] Alastair P Moore, Simon Prince, Jonathan Warrell, Umar Mohammed, and Graham Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[45] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

[46] Peer Neubert and Peter Protzel. Superpixel benchmark and comparison. In *Proc. Forum Bildverarbeitung*, 2012.

[47] Richard Nock and Frank Nielsen. Statistical region merging. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1452–1458, 2004.

[48] Aude Oliva and Antonio Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

[49] Christopher A Peters and Faramarz Valafar. Comparison of three nonparametric density estimation techniques using bayes' classifiers applied to microarray data analysis. In *METMBS*, pages 119–125, 2003.

[50] Andrew Rabinovich, Andrea Vedaldi, and Serge J Belongie. *Does Image Segmentation Improve Object Categorization?* Citeseer, 2007.

[51] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *ICCV*, Rio de Janeiro, 2007.

[52] Andrew Rabinovich, Andrea Vedaldi, Carolina Galleguillos, Eric Wiewiora, and Serge Belongie. Objects in context. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, pages 1–8. IEEE, 2007.

[53] William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[54] Vladimir Risojevic, Snjezana Momic, and Zdenka Babic. Gabor descriptors for aerial image classification. In Andrej Dobnikar, Uros Lotric, and Branko Ster, editors, *ICANNGA (2)*, volume 6594 of *Lecture Notes in Computer Science*, pages 51–60. Springer, 2011.

[55] Martin Rotard, Christiane Taras, and Thomas Ertl. Tactile web browsing for blind people. *Multimedia Tools Appl.*, 37(1):53–69, mar 2008.

[56] Bryan Russell, Antonio Torralba, Ce Liu, Rob Fergus, and William T Freeman. Object recognition by scene alignment. In *Advances in Neural Information Processing Systems*, pages 1241–1248, 2007.

[57] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[58] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[59] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[60] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Computer Vision–ECCV 2006*, pages 1–15. Springer, 2006.

[61] Chanop Silpa-Anan and Richard Hartley. Optimised kd-trees for fast image descriptor matching. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[62] David Stutz, Alexander Hermans, and Bastian Leibe. Superpixel segmentation using depth information. http://davidstutz.de/, September 2014.

[63] Joseph Tighe and Svetlana Lazebnik. Superparsing. *International Journal of Computer Vision*, 101(2):329–349, 2013.

[64] Joseph P. Tighe. *Towards Open-universe Image Parsing with Broad Coverage*. PhD thesis, Chapel Hill, NC, USA, 2013. AAI3606774.

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

[66] Zhuowen Tu and Xiang Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(10):1744–1757, 2010.

[67] Ranjith Unnikrishnan and Martial Hebert. Measures of similarity. In *Application of Computer Vision, 2005. WACV/MOTIONS'05 Volume 1. Seventh IEEE Workshops on*, volume 1, pages 394–394. IEEE, 2005.

[68] Ranjith Unnikrishnan, Caroline Pantofaru, and Martial Hebert. A measure for objective evaluation of image segmentation algorithms. In *Computer Vision and Pattern Recognition-Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, pages 34–34. IEEE, 2005.

[69] Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin de Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VII*, ECCV'12, pages 13–26, Berlin, Heidelberg, 2012. Springer-Verlag.

[70] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision: Special Issue on Texture Analysis and Synthesis*, 62(1–2):61–81, April 2005.

[71] Andrea Vedaldi and Stefano Soatto. Quick shift and kernel methods for mode seeking. In *Computer Vision–ECCV 2008*, pages 705–718. Springer, 2008.

[72] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 211–224, Berlin, Heidelberg, 2010. Springer-Verlag.