Library Declaration Form

University of Otago Library

Author's full name and year of birth: Xianbin Gu,
(for cataloguing purposes) 29/10/1979

Title of thesis: Image Segmentation Using Superpixel Ensembles

Degree: Doctor of Philosophy

Department: Department of Information Science

Permanent Address: 111 North Road, Dunedin, NZ

I agree that this thesis may be consulted for research and study purposes and that reasonable quotation may be made from it, provided that proper acknowledgement of its use is made.

I consent to this thesis being copied in part or in whole for

i) a library

ii) an individual

at the discretion of the University of Otago.

Signature:

Date:

Image Segmentation Using Superpixel Ensembles

Xianbin Gu

a thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand.

February 27, 2017

Abstract

Recently there has been an increasing interest in image segmentation due to the needs of locating objects with high segmentation accuracy as required by many computer vision and image processing tasks. While image segmentation remains a research challenge, "superpixel" as perceptual meaningful grouping of pixels has become a popular concept and a number of superpixel-based image segmentation algorithms have been proposed. The goal of this thesis is to examine the state-of-the-art superpixel algorithms and introduce new methods for achieving better image segmentation outcome.

To improve the accuracy of superpixel-based segmentation, we propose a color covariance matrix based segmentation algorithm (CCM). This algorithm employs a novel color covariance descriptor and a corresponding similarity measure method. Moreover, based on the CCM algorithm, we propose a multi-layer bipartite graph model (MBG-CCM) and a low-rank representation technique based algorithm (LRR-BGM). In MBG-CCM, different superpixel descriptors are fused by a multi-layer bipartite graph, and in LRR-CCM, the similarities of the covariance descriptors of the superpixel are measured by the subspace structure. Besides, we develop a new over-segmentation, called superpixel association, and propose a novel segmentation algorithm (SHST) which is able to generate hierarchical segmentation from superpixel associations.

In addition to those unsupervised segmentation algorithms, we also explore the algorithms for supervised segmentation. We propose a model for semantic segmentation, named "generalized puzzle game", by which the segmentation information containing in the superpixels can be integrated into the supervised segmentation.

Acknowledgements

The Ph.D. process is an intellectually stimulating and emotionally draining journey to me, but I am privileged to have received the support from many people. First of all, I'd like to thank my supervisors. To my primary supervisor Jeremiah D. Deng, thank you for your patient guidance and constant encouragement which helped me to go through the many ups and downs over the years. To my co-supervisor Martin K. Purvis, thank you for your academic advice and constructive comments which inspired me greatly in the research. And, I am also grateful to Dr. Brendon J. Woodford who gave me lots of advice and tips that are helpful in research.

I would like to thank the former and current postgraduate members in our Intelligent Computing and Networking (ICoN) lab, especially, Ahmad Shahi, Feng Zhou, Hanhe Lin, Joyce Zhang, Juan Zhang, Sean Lee, and Sepideh Zareei. The professional discussions and feedbacks from you undoubtedly contributed to my research. I also appreciate the staff in the Department of Information Science, University of Otago. To Associate Professor Peter Whigham, thank you for the feedbacks in the progress report meetings. To Gail Mercer, Heather Cooper and Stephen Hall-Jones, thank you for the liberal assistance. To the Technical Support Group (TSG), thanks for the efficient IT support.

Finally, I want to thank my parents and Jing Lu, for their support and understanding during my whole Ph.D. study period. To my beloved wife, thank you most of all.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Image segmentation

Image segmentation is a process to partition the image into several independent regions that are supposed to be meaningful and semantically related (Wang *et al.*, 2013). For human, it is more like an inherent skill: the light comes into the eyes and the brain instantly perceives objects, such as people, buildings, cars, or other things which makes up our real world. But in computer vision, things are completely different: the lights go through the camera and what the computer "perceives" are some dots of color, namely, pixels. Unfortunately, such difference leads to a "semantic gap" (Liu *et al.*, 2007; Wu *et al.*, 2012) between the human and computer visual experiences, which has long frustrated the field of computer vision. Figure 1.1 is an example of semantic gap in image processing. Converting the pixels into meaningful objects is still a great challenge for researchers in this domain.

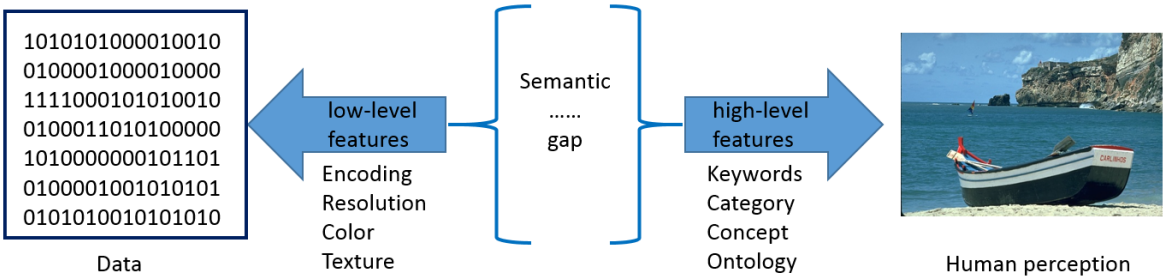


Figure 1.1: The semantic gap in computer vision.

Since the Gestalt movement in psychology (Wertheimer, 1938) has pointed out that perceptual grouping plays a critical role in human visual perception, researchers turn

to mathematical models that simulate this perceptual grouping behavior. Accordingly, the digital representations of an image, such as color and texture, are regarded as low-level features since they are similar to the machine language, which is designed for describing elementary and reproducible operations. And those semantic image representations, such as contents in the image, or object-ontology (Liu *et al.*, 2007), are considered to be high-level features because they are more abstract and close to the natural language. To that extent, the image segmentation is actually a clustering problem in the view of data mining, which tends to group the points represented by low-level features into a set of clusters that are associated with the high-level features.

Many segmentation techniques have been developed (Zhu *et al.*, 2016). They are usually divided into two categories: supervised segmentation and unsupervised segmentation, based on the difference in their modeling approaches. For supervised segmentation, the algorithm is designed in a top-down manner, which means, when given a number of labeled images, the algorithm is able to learn the object descriptors from the pixels belonging to the same object (Kontschieder *et al.*, 2015). For the unsupervised segmentation, the algorithm is designed under a bottom-up style, by which the pixels are considered as one object if they are locally coherent (Ren and Malik, 2003). However, these two kind of image segmentation methods are not mutually exclusive. A few works show that the unsupervised segmentation is able to improve the performance of supervised segmentation (Malisiewicz and Efros, 2007; Kohli *et al.*, 2009), because image cues containing in the unsupervised segmentations are informative. And, the unsupervised segmentations can also get merits from the supervised segmentations (Borenstein *et al.*, 2004; Fidler *et al.*, 2013), because the supervised segmentations provide the prior knowledge about the objects in the image.

1.2 Superpixel

The concept of superpixel is first proposed by Ren and Malik (2003), as a preprocessing stage for a two-class classification segmentation model. Essentially, superpixel is a group of pixels, in which the pixels are close to each other in some given feature space. Usually, the superpixels can be obtained from the bottom-up segmentation algorithms which are regarded as superpixel algorithms in some literatures. And, the output of a superpixel algorithm is called superpixel segmentation, or superpixel representation. A number of works have shown that superpixel segmentation is an effective and efficient representation of the image (Ding and Yilmaz, 2008; Wang *et al.*, 2013; Fulkerson *et al.*,

2009; Li *et al.*, 2012; Gould *et al.*, 2014; Huang *et al.*, 2016).

There are a few reasons for using superpixels instead of pixels in computer vision applications. Firstly, even for an image at moderate resolutions, the number of pixels is tremendous, which makes the pixel-level operation intractable. But if the image is represented by superpixels, the computational cost can drop down without too much loss of image information because they are local and inherent, preserving the structure needed for further segmentation.

Secondly, the pixels are not natural entities of an image but a consequence of the discrete representation of an image. So, in the view of the perceptual grouping theory, partition the image on a superpixel-level should be more likely to happen in real human vision than directly using pixels.

Thirdly, from the pixel grouping principles (Wertheimer, 1938; Palmer, 1999), it has been pointed out that proximity, similarity and continuation are critical for generating proper segments. So, it is more appropriate to think the image segmentation as a hierarchical structure. However, such a structure should be built by combining different low-level cues. Apparently, a superpixel is a richer source than a single pixel when extracting such combined features.

Generally, the superpixel representation of an image is obtained via an unsupervised segmentation, and as a pre-processing step in many real practice, the superpixel representation is always set to over segment the image. Figure 1.2 demonstrates an example of superpixel segmentations.

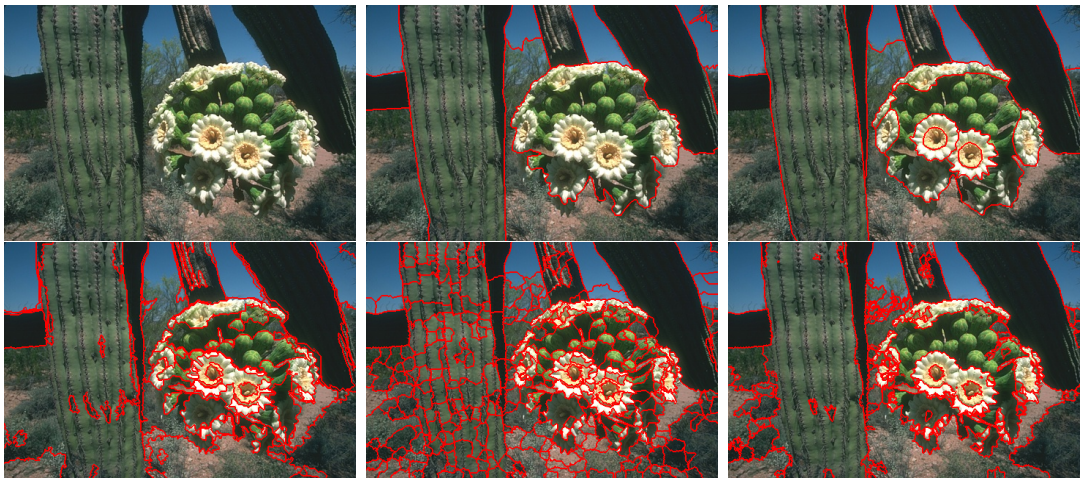


Figure 1.2: An example of superpixel segmentations: the upper row is the original image and two and human-annotated ground truth segmentations, the lower row is the example of suerpixel segmentations.

1.3 Challenges

The research in image segmentation has been carried out for years, and, the researchers have indeed gone a long way toward achieving robust, high quality segmentations. However, there are still several challenges facing the state of the art in this field.

The first challenge is to link the semantic gap between low-level features and high-level semantic effectively. Hundreds of segmentation algorithms have been proposed to implement pixel clustering and classification, which contains supervised and unsupervised models, but it remains difficult to ensure the segmentation result with meaningful partitions, especially for those unsupervised segmentation algorithms. Because even with small variations in brightness, lighting and view, the low-level appearance of an object will change drastically in different images. Although many descriptors have been developed for extracting robust features, such as SIFT (Ng and Henikoff, 2003) or HoG (Dalal and Triggs, 2005), there is still a long way to go for reducing the semantic gap.

The second challenge is to produce accurate segmentation for images. As an application oriented task, accurate segmentation results may not be necessary in some cases. But the demand for accurate segmentation is rising. For example, those image/video processing applications that can automatically recognize the objects in photos are highly required due to the popularity of mobile devices nowadays. Moreover, a robust and accurate segmentation will also improve many traditional applications such as object detection or content based video coding (Liu *et al.*, 2007; Huang *et al.*, 2011). The research in deep learning has push the accurate segmentation a big step forward (LeCun *et al.*, 2010), but to obtain a well-trained convolutional neural network needs tremendous training samples, which may not applicable in some cases. And for unsupervised segmentation, researchers employed ensemble techniques to generate robust and accurate segmentations (Li *et al.*, 2012). Although many of them are able to improve the accuracy of segmentations, we still lack of knowledge about what kind of feature is necessary for ensemble and how to effectively combine the features from different feature spaces.

Finally, computational efficiency is another issue of concern. In the process of segmentation, it is quite common to process large affinity matrices. A segmentation procedure can become intractable because of the requirement on extremely large amounts of memory or loops. This may impose a limitation on the size of images that can be processed by the segmentation algorithms. Although the limitations may be solved

gradually by the persistent increase in computation power and storage capacity of modern computers, the demand for efficient segmentation algorithms remains high. For instance, in most mobile devices, the computational ability of the processors is still limited.

1.4 Research objective

The research objective of the thesis is to develop superpixel-based techniques for image segmentation, which is able to cope with the challenges mentioned above. More specifically, the goals are to study relevant issues and propose new methods for integrating the superpixel segmentations into image segmentation process. The research mainly focuses on the following questions:

- Can we develop an efficient descriptor for superpixels which is able to improve the existing segmentation algorithms?
- Can we find some methods for combining the superpixel descriptors extracted from different feature spaces?
- Is there any method that can improve the performance of the handcrafted superpixel descriptors?
- Is there new method that can generate image segmentation with superpixels more effective than the state of the art?
- Can we make use of the image cues in superpixel segmentations to improve the supervised segmentation?

1.5 Contributions

The main contributions of this dissertation include:

- Improving a state-of-the-art superpixel-based image segmentation algorithm by
  - proposing a novel descriptor for superpixel which provides a strong texture representation for the superpixels, and
  - finding a proper method for similarity measuring among the superpixels.

- Proposing a multi-layer bipartite graph model for combining superpixel descriptors extracted from different feature spaces. This includes:
  - developing a multi-layer bipartite graph model, and
  - proposing a algorithm for partitioning the multi-layer bipartite graph.
- Proposing a low-rank representation method for the covariance descriptors of superpixel, which can improve the robustness of the algorithms that run with covariance descriptors.
- Proposing a new superpixel-based image segmentation method by first proposing a new type of primitives for image processing, namely superpixel association, and then developing a segmentation algorithm based on superpixel association.
- Developing a novel framework for integrating unsupervised segmentation into supervised segmentation.

1.6 Organization of the thesis

The rest of the thesis is organized as follows:

- **Chapter 2 The Fundamentals**

  This chapter introduces the fundamentals of superpixel-based image segmentation. Details of algorithms for superpixel generation and ensemble segmentation are discussed. And, the methods and data sets for evaluation are also elaborated here. Furthermore, a general framework of our research is given.

- **Chapter 3 Superpixel-based Segmentation with Covariance Matrix**

  In this chapter, we propose a novel covariance descriptor for superpixel and develop an superpixel-based segmentation algorithm named CCM by integrating the covariance descriptor into an ensemble segmentation method. Some parts in this chapter have been published in (Gu *et al.*, 2014a).

- **Chapter 4 Improving the Color Covariance Martrix base Segmentation with Subspace Representation**

  In this chapter, we proposed a method for improving the performance of the CCM algorithm, named MBG-CCM. In the MBG-CCM, we employs a multi-layer bipartite graph to model the superpixel descriptors from different feature

spaces, and then a novel method is proposed for merging different features. Parts of this chapter have been published in (Gu *et al.*, 2014b).

- **Chapter 5 Low-Rank Representation for Covariance Descriptor**

  In this chapter, we proposed a algorithm, called LRR-CCM. In LRR-CCM, a low-rank representation method is developed for the covariance descriptors of superpixel, which is able to remove the noises in the covariance descriptor set and improve the robustness of the segmentation. Parts of this chapter have been published in (Gu and Purvis, 2016).

- **Chapter 6 Superpixel Association**

  In this chapter we proposed a new concept of over-segmentation, named superpixel association. Some properties of superpixel association are discussed and we demonstrate that, the superpixel association is more suitable to be the primitive for further image processing. Besides, a segmentation algorithm based on the superpixel associations is also proposed, which is able to produce hierarchical segmentations in a tree structure. Parts of this chapter have been published in (Gu *et al.*, 2016).

- **Chapter 7 Semantic Segmentation with Unsupervised Segmentation**

  In this chapter, we propose a semantic segmentation framework, called generalized puzzle game, by which the unsupervised segmentations can be integrated into the labeling process.

- **Chapter 8 Conclusion and Future work**

  This chapter contains the conclusion drawn for the research carried out in this dissertation. And some possible research directions for future work are also included.

Chapter 2

The Fundamentals

Image segmentation involves a wide range of disciplines in a broad sense, including mathematics, psychology, computer science, and machine learning etc., some of which are far beyond the scope of this thesis. In this chapter, we concentrate on the fundamental techniques for this thesis, especially the algorithms for superpixel generation and methods of ensemble clustering.

The chapter is organized as follows. Section 2.1, Section 2.2, and Section 2.3 are the reviews of the image segmentation, superpixel algorithms, and ensemble segmentation respectively. Section 2.4 and Section 2.5 are the introductions to the data sets and the evaluation methods used in this thesis. In Section 2.6, we present our research framework, and Section 2.7 is a summary of this chapter.

2.1 Basic approaches for image segmentation

The research about the working mechanisms of the perceptual grouping ability in human vision has been carried out for about eighty years. The study began with the work by the scientists in cognitive science (Wertheimer, 1938), and the computer scientists joined them in the late 1960s (Boden, 2006). Researchers are interested in simulating the perceptual grouping ability by computers and keen on developing applications with it, which makes image segmentation a classical topic in the field of computer vision. As one of the basic operations in computer vision, image segmentation is considered to be a process that partition a natural image into some independent meaningful regions, for example, some particular objects or parts. However, the definition of the "meaningful object" is ambiguous; it can be the things, like a person or a car, or, sky or sea. More interestingly, even a combination of "objects", sometimes, is also considered as one

"object". Figure 2.1 demonstrates an example. In the example, it is easy to notice that the ways for partitioning mountain and sky are obviously different between the human subjects, which means the perception of different people is not same. Thus, a "correct" segmentation is hard to defined, which makes the image segmentation not a well-defined problem. Another difficulty in image segmentation is the methods for rep-

Figure 2.1: An example of blurring definitions of "object". The upper left is the original image, and the rest are human-annotated ground truth segmentations.

resenting "object". In human vision, the "objects" are perceived by the brain as words, but in computers, they are a few sets of low-level features. To bridge this semantic gap is still a challenge nowadays.

Fortunately, some pathways for developing segmentation algorithms can be found from the research in human perception. The Gestalt theory and studies in cognition (Wertheimer, 1938; Hoffman and Singh, 1997) proposed a few principles of human perception. For example, in human vision, elements similar in color, shape, or spatial position, are tended to be grouped together. A lot of research has been launched in the field of image segmentation. The existing methods can be categorized into two major categories: unsupervised methods and supervised methods. And for those supervised methods, they can again be divided into semi-supervised and fully-supervised methods based on how much supervision involving.

For unsupervised segmentation, the pixels are grouped into non-overlapped regions by their similarity over the low-level features (e.g. colors, textures) without any prior knowledge about the image, i.e., there are no training examples. Therefore, they carry out image segmentation by clustering the pixels with mixture models, mode shifting,

or graph partitioning. We divide the conventional unsupervised methods into two categories: the graph based methods and the clustering based methods (Zhu *et al.*, 2016).

The graph based methods formulate an image as a graph $G(V, E)$, where V is set to be the pixels (or regions), E is the set of edges linking the vertices. Each edge is associated with a weight, which reflects the similarities between the pixels (or regions). The image is partitioned according to the optimization function defined on the graph. There are a few graph based methods that are commonly used in unsupervised segmentation, which includes the F-H method (Felzenszwalb and Huttenlocher, 2004), Normalize Cut (Shi and Malik, 2000), and Watershed (Vincent and Soille, 1991; Couprie *et al.*, 2009) etc.

The clustering based segmentation methods are developed on the real analysis techniques in data mining. Generally, they encode the pixels into a feature vector space, and then run clustering in that space. These methods tend to partition image into small regions due to the fact that most low-level features are actually local statistics. There are some popular clustering based segmentation algorithms, which includes, K-means, Mixture of Gaussian (Rao *et al.*, 2009) and Mean Shift (Comaniciu and Meer, 2002; Vedaldi and Soatto, 2008) etc.

It is worth noting that most of the superpixel algorithms come from unsupervised segmentation methods. In most cases, the superpixels can be generated directly by tuning the parameters in the unsupervised algorithms, for examples, adjusting the cluster number in Normalized Cut. But there are also a few unsupervised algorithms proposed especially for superpixel generation, such as TurboPixel (Levinshtein *et al.*, 2009), Superpixel lattices (Moore *et al.*, 2008) and SLIC (Achanta *et al.*, 2012) etc.

For supervised segmentation, the algorithms are designed for incorporating high-level information as prior knowledge so that the ill-posed segmentation problem can get a better definition.

In semi-supervised methods, the prior knowledge is obtained under a framework of interaction between human and machine, in which a few pixels in the given image are labeled manually and then the algorithms adopt a self-training procedure to learn the model parameters and conduct segmentation. The popular methods of semi-supervised segmentation include GrabCut (Rother *et al.*, 2004) and OneCut (Tang *et al.*, 2013) etc.

In fully-supervised methods, the algorithms train a segmentation model by extracting knowledge about the objects from the given training samples, which generally are

well labeled, i.e. all pixels are assigned to some object class labels. Based on different applications, there are two major tasks in fully-supervised segmentation: "object proposals" and "semantic segmentation" (Zhu *et al.*, 2016).

For the first one, it seeks to locate the objects with regions that have high probabilities to cover the objects. And, the algorithms generally often employ a few bounding box detectors for generating the region candidates and select the optimal by some trained classifiers, such as SVM etc (Tsai *et al.*, 2015; Felzenszwalb *et al.*, 2010). Moreover, salient object detection methods are adopted to replace the bounding box detectors for region pooling in some works (Hosang *et al.*, 2014; Zhu *et al.*, 2015; Borji *et al.*, 2014).

For semantic segmentation, the goal is to develop the algorithms that can partition an image into independent regions and associate them with some object classes predefined, e.g. people, cat, and sheep etc. Since the task of semantic segmentation is not simply producing the possible regions that objects may locate in but parsing the whole image into different "things" and "stuff", Markov random field (MRF) and conditional random field (CRF) are often employed for modeling the neighborhood relations displayed in the training samples (Zheng *et al.*, 2015; Ladicky *et al.*, 2010; Shotton *et al.*, 2006; Gould *et al.*, 2008).

Figure 2.2 shows a categorization of the image segmentation methods.

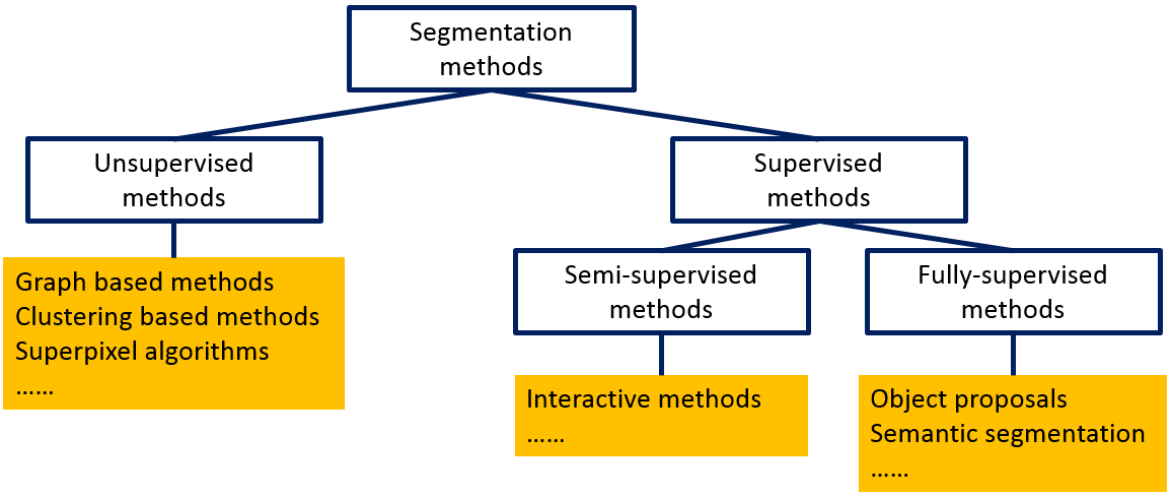


Figure 2.2: A categorization of existing image segmentation methods.

Image segmentation is one basic process of many computer vision applications, and the purpose of a segmentation algorithm varies based on the applications it belongs to. Thus, there are various ideas for designing a new segmentation algorithm. But

among them, there is a notable trend in developing superpixel-based algorithms. The motivations are obvious: for unsupervised segmentation, superpixel provides a format for extracting more complex and discriminative features; for supervised segmentation, using superpixel can reduce the time for training and inference. However, the superpixel is not a perfect replacement of pixel. For example, superpixel wrecks the regular grid structure of pixels which may bring problems to the definition of neighborhood. And, the inappropriate parameters in superpixel algorithms will introduce structure errors into superpixel representation. All these problems are worthy of further investigation.

2.2 Algorithms for superpixel generation

2.2.1 Overview

Since superpixels are in fact perceptual groupings of pixels, and naturally, most of the unsupervised segmentation algorithms can be used for superpixel generation. But in real practice, because superpixels always serve as primitives for further computation, the algorithms for superpixel generation are supposed to have a few distinctive properties, including boundary coherency, computational efficiency, hierarchy, and topology preserving (Wei *et al.*, 2016; Achanta *et al.*, 2012). In the past few years, there have been considerable achievements in superpixel segmentation, and most of the state-of-the-art methods possess one or more of the properties mentioned. Liu *et al.* (2011) proposed a graph based method which is able to produce segmentation with good accuracy. Van den Bergh *et al.* (2012) proposed the SEEDS algorithm that achieves a compromise between accuracy and efficiency for superpixel generation. In Moore *et al.* (2008, 2010), the proposed algorithms are able to generate superpixels that conform to a grid topology which can be integrating into many vision algorithms conveniently. And, some superpixel algorithms (Felzenszwalb and Huttenlocher, 2004; Mei *et al.*, 2013) use a tree structure of regions to represent the image, which can characterize the hierarchical structure of the image with a relative low computational complexity. In addition, a few unsupervised clustering algorithms are also widely used for generating superpixels, such as Normalized Cut (Shi and Malik, 2000) and Mean Shift (Comaniciu and Meer, 2002; Vedaldi and Soatto, 2008).

Frankly speaking, all superpixel generation approaches have their own advantages and drawbacks that may be better content to a particular application. In this thesis, the superpixel generation task is done by two popular superpixel algorithms: the

efficient graph based F-H method (Felzenszwalb and Huttenlocher, 2004) and Mean Shift (Comaniciu and Meer, 2002). This choice is based on two facts.

First, these two algorithms are based on the data-driven models by which the intrinsic structure of the data can be easily investigated with a scale parameter. For example, the F-H method merges the pixels into superpixels according to a predefined minimum difference controlled by a threshold, and, the Mean Shift is a nonparametric gradient-based method which seeks the modes along the surface of the data distribution with a given step length. So, different superpixel segmentations can be obtained by simply adjusting the threshold or step length.

Second, these two superpixel algorithms are complementary and practically efficient (Li *et al.*, 2012). The graph based and gradient based methods are motivated by different goal functions, which means the pixel data structure can be explored by different clustering procedures. And in the view of ensemble clustering, this may contribute to the robustness of the final clustering (Zhou, 2012; Zhou *et al.*, 2015). Actually, in the existing works on superpixel-based image segmentation, this combination is widely used (Li *et al.*, 2012; Wang *et al.*, 2013, 2015). However, we have to mention that the algorithms we proposed in this thesis are also feasible with other choices on superpixel algorithms.

For completeness, we elaborate the F-H method and Mean Shift in the following.

2.2.2 The efficient graph based image segmentation

The F-H method is proposed by Felzenszwalb and Huttenlocher (2004). Let $G(V, E)$ be an undirected graph, where vertices $v_i \in V$ represent the set of elements to be segmented, and edges $e(v_i, v_j) \in E$ correspond to pairs of neighboring vertices. Moreover, each edge $e(v_i, v_j) \in E$ has a weight $w(v_i, v_j)$, which is a non-negative value measured by the dissimilarity between v_i and v_j. And, a segmentation $S = \{C_i\}$ is a partition of V, and, $\forall i, j \in \{1, 2, ..., n\}$, we have $C_i \cap C_j = \emptyset$ and $C_i \in S$.

For a given image I, the pixels are set to be the elements in V and the weight on an edge is some measure of the dissimilarity between two pixels connected by the edge, which could be the difference in intensity, color, or some other attributions.

The basic idea of F-H method is that the weights on the edges connecting vertices in the same component should be relatively smaller while those on the edges connecting vertices in different components should be larger. Thus, three indices are defined for describing this idea. The first is the *internal difference* of a component C, which is

defined as,

$$\mathrm{Int}(C) = \max_{e \in \mathrm{MST}(C,E)} w(e), \tag{2.1}$$

where $w(e)$ is the weight on the edge e, and $\mathrm{MST}(C, E)$ represents the edges in the minimum spanning tree of the component C. The second is the *minimum internal difference*,

$$\mathrm{MInt}(C_i, C_j) = \min(\mathrm{Int}(C_i) + \tau(C_j), \mathrm{Int}(C_i) + \tau(C_j)), \tag{2.2}$$

where $\tau(C) = k/|C|$ is a threshold function controls the degree of difference between two components. The third is the *difference* between two components C_i and C_j,

$$\mathrm{Dif}(C_i, C_j) = \min_{v_m \in C_i, v_n \in C_j, (v_m, v_n) \in E} w(v_m, v_n), \tag{2.3}$$

Then, the *pairwise comparison predicate* is defined as,

$$D(C_i, C_j) = \begin{cases} \text{true} & \text{if } \mathrm{Dif}(C_i, C_j) > \mathrm{MInt}(C_i, C_j), \\ \text{false} & \text{otherwise.} \end{cases} \tag{2.4}$$

If $D(C_i, C_J) = \text{true}$, then C_i and C_j will be merged. The details of F-H method are as shown in Algorithm 2.1.

The computational complexity is $O(m \log m)$, where m is the number of edges in the graph (Felzenszwalb and Huttenlocher, 2004).

Algorithm 2.1 F-H method

Input: A graph $G = (V, E)$

Output: A segmentation $S = (C_1, ..., C_r)$ of V

1: Sort E into $\pi = (e_1, ..., e_m)$, where $\forall i < j$, $e_i \leq e_j$;
2: Initializing S by setting $S^0 = V$
3: **for** $q = 1, \cdots, m$ **do**
4: Let v_i, v_j be the vertices connected by e_q, and, C_i^{q-1} and C_j^{q-1} be the components of S^{q-1} containing v_i and v_j respectively,
5: **if** $C_i^{q-1} \neq C_j^{q-1}$ and $w(e_q) \leq MInt(C_i^{q-1}, C_j^{q-1})$ **then**
6: merging C_i^{q-1} and C_j^{q-1} to get S^q
7: **else**
8: $S^q = S^{q-1}$
9: **end if**
10: **end for**
11: $S = S^m$

2.2.3 Mean Shift segmentation

The mean shift algorithm is proposed by Fukunaga and Hostetler (1975) and Cheng (1995); Comaniciu and Meer (2002) introduced Mean Shift into image segmenation. The algorithm is essentially a mode seeking procedure, which is based on the density estimation.

Given a dataset $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\} \subset R^d$, where $\mathbf{x}_i$ is a point in a d-dimension space and predefined kernel $K(\mathbf{x})$. And, let the diagonal $H = h^2 I$ be the bandwidth matrix, where h is a fixed bandwidth for all dimensions. Then a multivariate kernel density estimator is defined as

$$\hat{f(\mathbf{x})} = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}), \tag{2.5}$$

where $K_H(\mathbf{X})$ is defined as

$$K_H(\mathbf{x}) = |H|^{-\frac{1}{2}} K(|H|^{-\frac{1}{2}}\mathbf{x}). \tag{2.6}$$

And, let $k(||\mathrm{x}||^2)$ be the *profile* of the kernel $K(\mathbf{x})$, which satisfies $K(\mathrm{x}) = c_{k,d}k(||\mathrm{x}||^2)$, where $c_{k,d} > 0$ is the normalization constant which makes $K_{\mathbf{x}}$ integrate to 1. Then Eq. 2.5, can be rewritten as

$$\hat{f(\mathbf{x})} = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2). \tag{2.7}$$

Since the modes are located in the place where the gradient $\bigtriangledown f(\mathrm{x}) = 0$, a mean shift procedure is designed to locate the zeros without estimating the density. From Eq. 2.7, we have

$$\bigtriangledown \hat{f}_{h,K}(\mathbf{x}) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x} - \mathbf{x}_i) k'(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2). \tag{2.8}$$

Let $g(x) = -k(x)$, and introduce it into Eq. 2.8, which yields

$$\begin{aligned} \bigtriangledown \hat{f}_{h,K}(\mathbf{x}) &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}) g(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} g\left(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2 \right) \right] \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2 \right)}{\sum_{i=1}^{n} g\left(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2 \right)} - \mathbf{x} \right]. \end{aligned} \tag{2.9}$$

The second term is defined as the *mean shift*, i.e.

$$\mathbf{m}_{h,G}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2 \right)}{\sum_{i=1}^{n} g\left(||\frac{\mathbf{x} - \mathbf{x}_i}{h}||^2 \right)} - \mathbf{x}. \tag{2.10}$$

Let

$$y_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(||\frac{\mathbf{x}-\mathbf{x}_i}{h}||^2\right)}{\sum_{i=1}^{n} g\left(||\frac{\mathbf{x}-\mathbf{x}_i}{h}||^2\right)}, \tag{2.11}$$

It has been proved that $\{y_j\}_{j=,1,2,\ldots}$ will converge if the kernel $K(\mathbf{x})$ has a convex and monotonically decreasing profile (Cheng, 1995). So, the modes can be found by the mean shift procedure, i.e. updating $\{y_j\}$ by $y_{j+1} = \mathbf{m}_{h,G}(y_j) + y_j$ until it converges.

In image segmentation, every pixel is associated to a mode via a mean shift procedure, and those pixels connected to the same mode are grouped as a cluster. The computational complexity of the Mean Shift is $O(n^2)$ (Vedaldi and Soatto, 2008), and Algorithm 2.2 shows its details.

2.3 Ensemble clustering in image segmentation

2.3.1 Overview

The ensemble clustering technique aims to combine the results of different clustering methods into a more robust and better clustering (Huang *et al.*, 2016), and it is also called consensus clustering. In the past few years, a number of ensemble segmentation approaches have been developed by employing a variety of ensemble clustering techniques into image segmentation. The early research concentrates on developing the frameworks. Franek *et al.* (2010) proposed a framework for adapting ensemble clustering methods into image segmentation. In their algorithm, the superpixels are used as primitive objects and the general ensemble clustering methods are applied on suprpixel-level. The weakness of their framework is lack of the ability to use the multiple image cues, such as color, and lightness etc. Mignotte (2008) developed a relabeling-based ensemble segmentation method, in which the local histogram of class labels is used to control the fusing of different segmentations. But this method assumes that every input over-segmentation should be partitioned into a fixed number of clusters, which is not applicable in some cases. Kim *et al.* (2014) proposed an algorithm which generates the final segmentation by using the hierarchical segmentations. And in some works, the ensemble segmentation is formulated into an optimization problem, by which the final segmentation is obtained by maximizing some predefined similarities between the segmentations (Vega-Pons *et al.*, 2011; Alush and Goldberger, 2012; Mignotte, 2014; Wang *et al.*, 2014). Besides, Wang *et al.* (2013) and Ding and Yilmaz (2008) introduced the hypergrah into ensemble segmentation, their models also take single superpixel segmentation as primitives and use multiple features for clustering.

Algorithm 2.2 Mean Shift

Input: An image $I = \{p_1, ..., p_n\}$, bandwidth h, stop-threshold τ, merge-threshold ϵ

Output: A segmentation $S = (C_1, ..., C_k)$

/* mean shift procedure */

$M = \{m_1, \cdots, m_n\}$
for $i = 1, \cdots, n$ **do**
  $y_0 = p_i$, $j = 0$, $m_i = 0$
  Compute y_1 by Eq. 2.11
  while $||y_{j+1} - y_j|| > \epsilon$ **do**
    $j = j + 1$
    $y_j = y_{j-1}$
    Compute y_{j+1} by Eq. 2.11
  end while
  $m_i = y_{j+1}$
end for

/* segmentation */

$k = 1$, $C_k = \emptyset$
while $M \neq \emptyset$ **do**
  $m_i = M\{1\}$
  $M = M - m_i$
  $C_k = C_k \cup p_i$
  for $m_j \in M$ **do**
    if $||m_i - m_j|| < \tau$ **then**
      $C_k = C_k \cup p_j$
      $M = M - m_j$
    end if
  end for
  $k = k + 1$
end while

However, the quality of their final segmentations may be affected by the superpixel segmenations they used.

Unfortunately, the image data is generally made up by a tremendous amount of pixels, which results in high computational cost for measuring the pixel-wise similarity globally. So, most of the existing ensemble segmentation algorithms are proposed to approach segmentation at the superpixel-level, and the robustness of the output is inevitably effected by the quality of the superpixel segmentation they adopted. Alternatively, Li *et al.* (2012) proposed an ensemble segmentation algorithm, which takes multiple superpixel segmentations as segmentation cues and can generate robust final segmentation. This algorithm employs a bipartite graph to represent the pixel-superpixel relations from the input superpixel segmentations and obtains the final segmentation by an modified normalized cuts. Wang *et al.* (2013, 2015) improved this method by proposing a novel method for measuring the similarity between the superpixels in the bipartite graph construction.

The bipartite graph model employed in (Li *et al.*, 2012; Wang *et al.*, 2013, 2015) is first proposed by Fern and Brodley (2004), named Hybrid Bipartite Graph Formulation (HBGF). In this thesis, we also use this model for superpixel-based segmentation.

2.3.2 The HBGF algorithm

The HBGF (Fern and Brodley, 2004) is a graph based ensemble clustering method. This algorithm contains two parts: the first one is the construction of the bipartite graph which integrates the clustering information, and the second one is spectral clustering by which the final clustering is obtained.

Given a data set $X = \{x_1, \cdots, x_m\}$, let $C^i = \{c_1^i, \cdots, c_{K_i}^i\}$ be a clustering which partitions X into K_i disjoint clusters, and $\mathcal{C} = \{C^1, \cdots, C^N\}$ be a collection of clusterings of X, where N is the number of clusterings, and $K = \{K_1, \cdots, K_N\}$ represents the set of cluster numbers of each clustering.

For a bipartite graph $G(V, E)$, the vertex set V can be divided into two parts, i.e. $V = V^I \cup V^C$, and for each edge $e \in E$, it connects a vertex in V^C to one in V^I, i.e. $\forall e_{ic} \in E$, $e_{ic} = (v_i, v_c)$, where $v_i \in V^I$ and $v_c \in V^C$. And, let each edge is associated to a weight w, we have, $w(e_{ic}) = w_{ic}$. And, the bipartite graph can be rewritten as $G(V^I, V^C, W)$, when it needs to emphasize that G is weighted.

In HBGF, the bipartite graph G is constructed by setting the elements in X as the vertices in V^I, i.e. $V^I = \{x_1, \cdots, x_n\}$, and all the entries in $\mathcal{C}$ as the vertices in V^C,

i.e. $V^C = \cup_{i=1}^{N} C^i$; and for the weights edges, we set,

$$w_{ic} = \begin{cases} 1 \text{ or, other positive number}, & \text{if } v_i \in v_c, \\ 0, & \text{otherwise.} \end{cases} \tag{2.12}$$

Let $n = \sum_{i=1}^{N} K_i$, then, $W = [w_{ic}]$ is a $m \times n$ matrix, which is called cross-adjacency matrix (Liu *et al.*, 2010). Algorithm 2.3 shows the details of the graph construction of HBGF.

Algorithm 2.3 Graph construction of HBGF

Input: A data set $X = \{x_1, \cdots, x_m\}$, a collection of base clusterings $\mathcal{C} = \{C^1, \cdots, C^N\}$, the cluster number k of the final clustering

Output: Final clustering $C = (c_1, ..., c_k)$

```
Set V^I = X, V^C = ∪_{i=1}^{N} C^i, n = Σ_{i=1}^{N} K_i;
W = ∅;
for i = 1, ··· , m do
    for c = 1, ··· , n do
        if v_i ∈_c then
            W = W ∪ w_ij
        end if
    end for
end for
```

The spectral clustering on G can be done by directly extending $W_{m\times n}$ into $W^{ext} = \begin{bmatrix} 0 & W \\ W^T & 0 \end{bmatrix}$, where W^{ext} is a $(m+n) \times (m+n)$ symmetric matrix. However, Dhillon (2001) proved that the normalized cut algorithm on a bipartite graph can be realized via SVD (i.e. singular value decomposition) on the cross-adjacency matrix W, and the clustering information of X is contained in the right singular vectors.

Li *et al.* (2012) proposed a more efficient algorithm for computing the singular vectors, which is called *T-cut*. This algorithm is proposed based on the truth that W can be converted into the probility of a two-step transition on the vertices of bipartite graph G. And, the *T-cut* algorithm makes use of the equivalence in the two-step transition and delivers the clustering of V^X from the clustering of V^Y without loss. The details of *T-cut* algorithm is given in Algorithm 2.4.

Algorithm 2.4 *T-cut*

Input: A cross-adjacency matrix W, the cluster number k of the final clustering

Output: Final clustering $S = (s_1, ..., s_k)$

1: Compute $D_X(i,i) = \sum_i w_{ij}$, $D_Y(j,j) = \sum_j w_{ij}$;
2: Compute $W_Y = W^T D_X^{-1} W$;
3: Compute $L_Y = D_Y - W_Y$;
4: Compute the buttom k eigenpairs $\{(\lambda_i, \mathbf{v_i})\}_{\mathbf{i=1}}^{\mathbf{k}}$ of $L_Y\mathbf{v} = \lambda D_Y\mathbf{v}$;
5: **for** $i = 1, \cdots, k$ **do**
6: Compute γ_i such that $0 \leq \gamma_i < 1$ and $\gamma_i(2 - \gamma_i) = \lambda_i$;
7: Compute $\mathbf{u_i} = \frac{1}{1-\gamma_i} D_X^{-1} W \mathbf{v_i}$;
8: **end for**
9: Cluster $\mathbf{u}$ into k clusters via k-means algorithm and obtain S.

2.4 Data sets

For evaluation the performance of the proposed approaches, there are three data sets used throughout this thesis, which includes the Berkeley Segmentation Data Set 300 ("BSDS300") (Martin *et al.*, 2001), the Berkeley Segmentation Data Set 500 ("BSDS500") (Arbelaez *et al.*, 2011), and the Microsoft Research Cambridge 21-Class Data Set ("MSRC21") (Shotton *et al.*, 2008).

BSDS300 is a public image segmentation database which is widely used in evaluating the performance of unsupervised image segmentation. This database contains 300 natural images of diverse scene categories, and each image has a number of ground truth segmentations which are manually segmented by different human subjects. There are at least 4 human annotations for each image, and all images are in the size of 481×321. BSDS500 is an update of BSDS300, which contains some more 200 images. The same as BSDS300, every image has a few human-annotated ground truth segmentations, and the image size is set to be 481×321. Figure 2.3 shows a few examples in the BSDS 300 and 500 data set.

MSRC21 is a classic dataset for semantic segmentation. This database consists of 591 images which are labeled with 21 classes: building, grass, tree, cow, sheep, sky, airplane, water, face, car, bicycle, flower, sign, bird, book, chair, road, cat, dog, body, boat. Normally, the dataset is split into 276 training samples, 256 test samples, and the rest for validation, which is same to Shotton *et al.* (2008). One difficulty for experiments on this data set is that the ground truth labeling is approximate, and many of the pixels on the object boundaries have void labels, which make the training

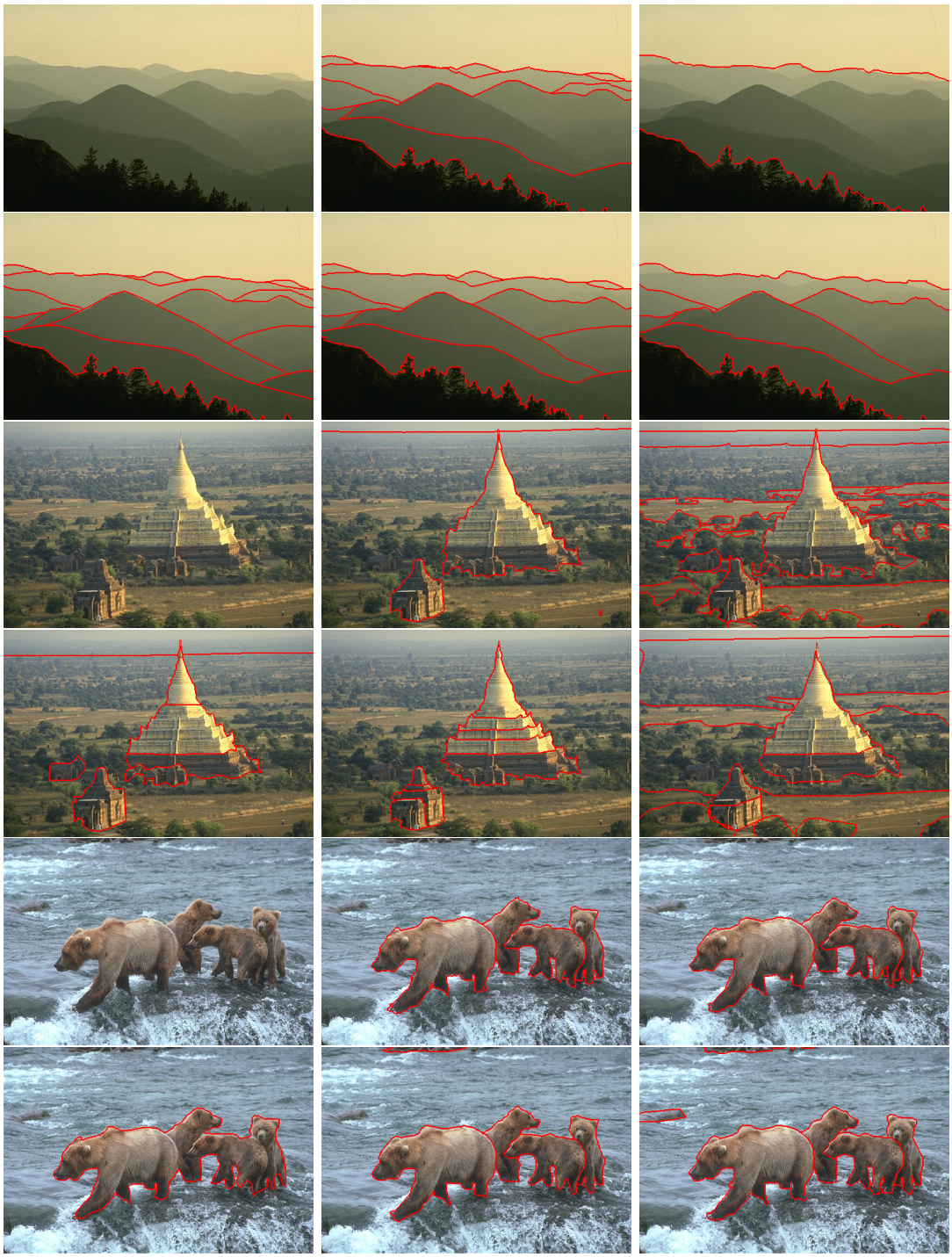


Figure 2.3: A few images and their ground truth segmentations from BSDS300 and 500 data sets. In every two rows: the upper left is the original image and the rest five are the ground truth segmentations made by different human subjects.

is difficult. Figure 2.4 demonstrates some images from MSRC21.

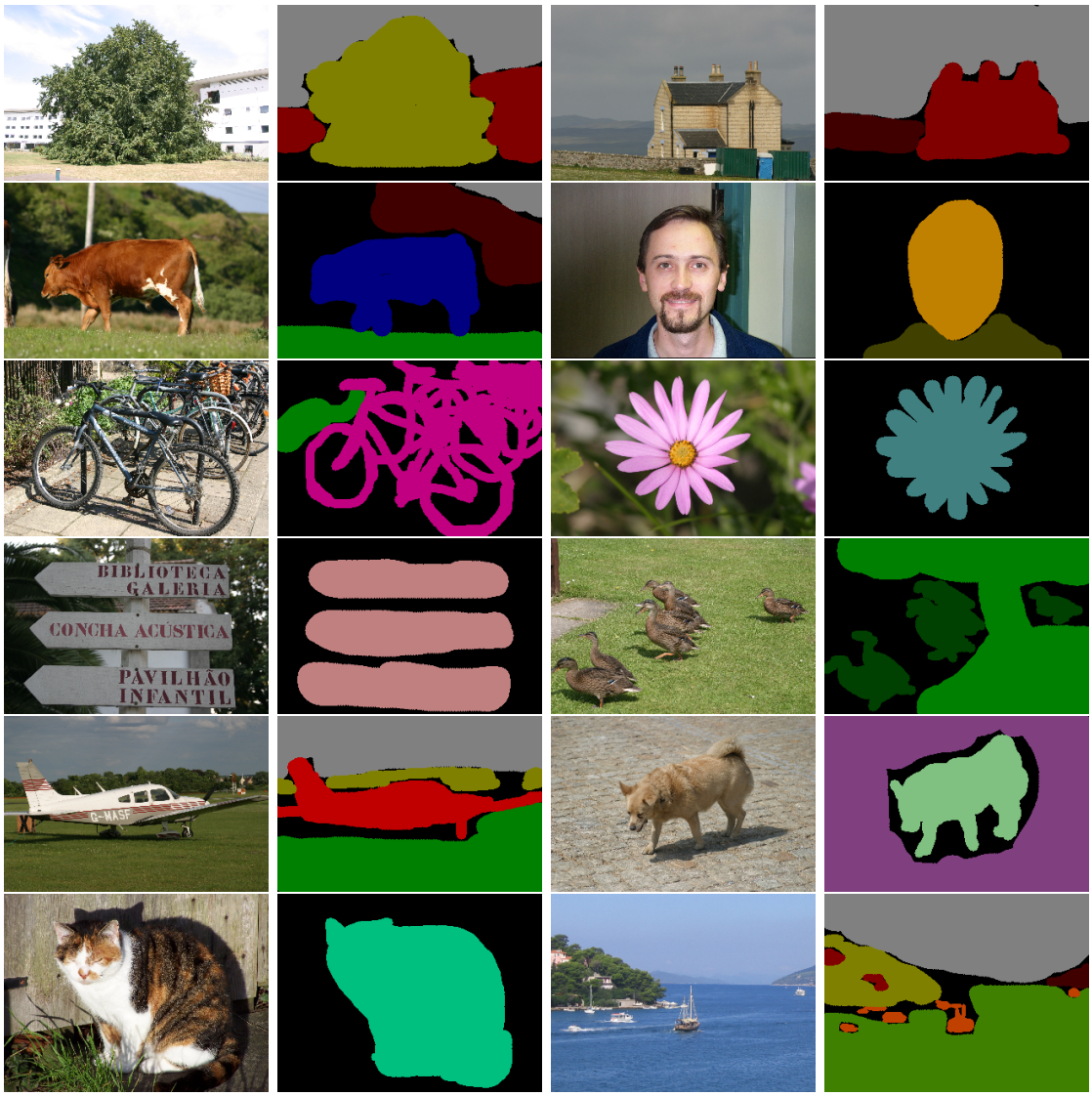


Figure 2.4: A few images and their ground truth segmentations from MSRC21 data sets. The original images are shown in odd columns, and the respective ground truth segmentations are listed in the even columns with class labels shown in color (the void label is in black).

2.5 Evaluation

Evaluating the quality of segmentation is commonly referred to as cluster validity analysis (Zhou, 2012). For comparison purpose, we employ the evaluation methods that are widely used in the image segmentation society.

2.5.1 Evaluations for unsupervised segmentation

Evaluation of an unsupervised segmentation algorithm is in fact largely subjective, mainly because there is no unique ground-truth segmentation of an image against which the outputs may be compared. However, there are four popular segmentation evaluation methods that are widely used in qualifying the segmentation result, which includes the Probabilistic Rand Index (PRI) (Unnikrishnan *et al.*, 2007), the Variation of Information (VoI) (Meilă, 2005), the Global Consistency Error (GCE) (Martin *et al.*, 2001), and the Boundary Displacement Error (BDE) (Freixenet *et al.*, 2002).

PRI is a generalization to the rand index, which measures the probability of an arbitrary pair of samples being labeled consistently in the two segmentations. In image segmentation, it can compare the segmentation result with a set of ground truth. Let S_t be the segmentation result and $\{S_g\}$ be a set of ground-truth, the PRI is defined as follows,

$$PRI(S_t, \{S_g\}) = \frac{1}{\binom{N}{2}} \sum_{i<j} [c_{ij} p_{ij} + (1 - c_{ij})(1 - p_{ij})], \tag{2.13}$$

where N is the number of pixels in the image, $c_{ij} \in [0, 1]$ is the event that pixel i and pixel j have the same label in S_t, and p_{ij} is the corresponding probability estimated with the sample mean. A higher PRI value means a better segmentation.

The VoI is a metric that relates to the conditional entropies between the class label distribution. It measures the sum of information loss and information gain between the two partitions, and it is defined as

$$VoI(S_g, S_t) = H(S_g) + H(S_t) - 2I(S_g, S_t), \tag{2.14}$$

where H and I are the respective entropies of the mutual information between two clusterings. A lower VoI value indicates better segmentation result.

The GCE measures the difference between two regions that contain the same pixel in different segmentations. Particularly, this metric compensates for the difference in granularity. Let $R(S, p_i)$ be the set of pixels within the regions in segmentation S that contains pixel p_i, and $'-'$ denotes the set difference. The GCE is defined as

$$GCE(S_t, S_g) = \frac{1}{N} \min \sum_i E(S_t, S_g, p_i), \sum_i E(S_g, S_t, p_i), \tag{2.15}$$

where $E(S_t, S_g, p_i) = \frac{|R(S_t, p_i) - R(S_g, p_i)|}{|R(S_t, p_i)|}$ is the local refinement error. Obviously, for the GCE values, being close to 0 implies a good segmentation.

The BDE measures the average displacement error of boundary pixels between two segmentations. The error of one boundary pixel is defined as the distance between the pixel and the closest pixel in the other boundary image.

For $p_i \in B_1$, we define $d(p_i, B_2) = \min_{p \in B_2} ||p_i - p||$ as the distance of a boundary point $p_i \in B_1$ to the boundary set B_2; let N_{B_1} and N_{B_2} denote the number of pixels in B_1 and B_2, the BDE is defined as

$$BDE(B_1, B_2) = \frac{\sum_i^{N_{B_1}} \frac{d(p_i, B_2)}{N_{B_1}} + \sum_i^{N_{B_2}} \frac{d(p_i, B_1)}{N_{B_2}}}{2}. \tag{2.16}$$

A lower BDE value means less deviation between the segmentation and ground truth.

In addition, we use the average rank to compare the overall performance of different algorithms because the performance of each algorithm is represented by a few indices. Let $R = r_1, \cdots, r_n$ be the set of ranks on n evaluation indices of an algorithm, the average rank is defined as

$$\text{Avg.R} = \frac{\sum_{i=1}^{n} r_i}{n}. \tag{2.17}$$

2.5.2 Evaluations for semantic segmentation

The performance of supervised segmentation is always evaluated based on the recall and precision criteria. Many researchers evaluated their algorithm via both the category average accuracy and the global accuracy (Shotton *et al.*, 2008; Gould *et al.*, 2014; Yao *et al.*, 2012). We follow their modus and use '*Global*' to refer the percentage of all pixels that were correctly classified and '*Avg(Class)*' for the average recall over all classes. Specifically, the recall of each class is computed by

$$Recall(Class_i) = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}, \tag{2.18}$$

and *Avg(Class)* is defined as

$$Avg(Class) = \frac{1}{N} \sum_{i=1}^{N} Recall(Class_i), \tag{2.19}$$

where N is the number of classes. Respectively, *Global* is computed by

$$Global = \frac{\sum_{i=1}^{N} \text{True Positive}(Class_i)}{\sum_{i=1}^{N} \text{True Positive}(Class_i) + \text{False Negative}(Class_i)}. \tag{2.20}$$

2.6 General framework

The research in image segmentation is often connected with some particular applications. So, the frameworks may vary across different applications. However, our research concentrates on developing image segmentation approaches based on superpixels, consequently, we have a general framework which dominates our research.

Generally, the work begins with generating superpixel segmentations and followed by a model construction procedure, where different feature extraction and similarity measure methods are proposed. And then, image segmentation is carried out, which could be unsupervised or supervised. We have to mention that the unsupervised segmentation carried out in this thesis, i.e. semantic segmentation, actually involves both unsupervised and supervised methods. Finally, there is an evaluation procedure. Those unsupervised segmentation algorithms are evaluated by PRI, VoI, GCE and BDE on BSDS 300 and BSDS 500. For the unsupervised algorithms, the experiments are conducted on MSRC21 and evaluated by *Avg(Class)* and *Global*. Figure 2.5 demonstrates the flowchart of the research framework.

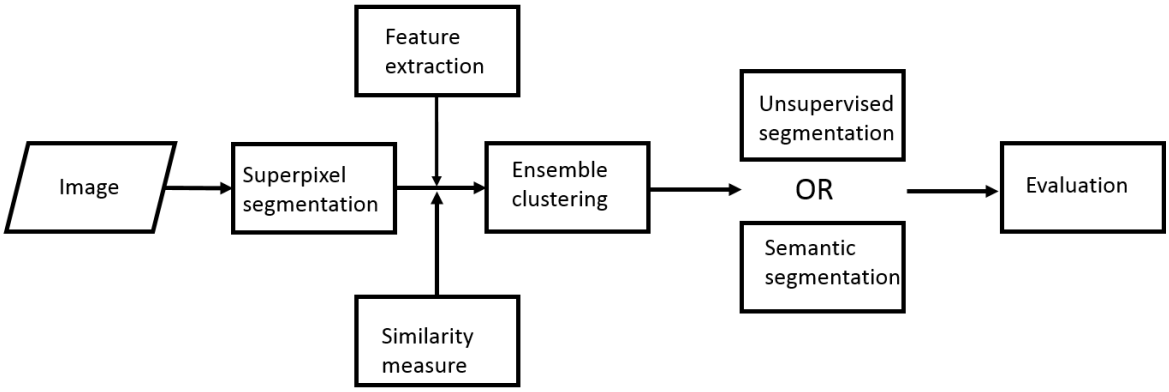


Figure 2.5: The flowchart of the research framework.

2.7 Summary

This chapter elaborates the fundamentals of our research, which includes methods for generating superpixels, models for ensemble segmentation and a few popular segmentation evaluation methods. However, there is still one thing need to be addressed, that is the necessity of ensemble segmentation.

The superpixel algorithms are actually bottom-up segmentation algorithms, and most of them are proposed based on the perceptual grouping theory. But in real practice, there are no such algorithms that can produce segmentation as good as human vision does. Figure 2.6 shows a few segmentations made by some popular superpixel algorithms. It is easy to notices that they all tend to over-segment the objects. One possible reason for this phenomenon is that the perceptual grouping in human vision is occurred by composing multiple features but most of the existing bottom-up segmen-

Figure 2.6: Segmentations from different superpixel algorithms. From left to right: the original image, Normalized Cut, F-H method, Mean Shift, and SLIC.

tation algorithms lack the ability to group pixels in multiple feature spaces. Therefore, the ensemble techniques in image segmentation are worthy of investigation.

Chapter 3

Superpixel-Based Segmentation with Color Covariance Matrix

3.1 Introduction

A number of clustering algorithms can be used to segment an image, e.g., the gradient-based algorithms such as Mean Shift (Comaniciu and Meer, 2002) and SLIC (Achanta *et al.*, 2012), and graph-based methods such as Ncut (Shi and Malik, 2000), F-H algorithm (Felzenszwalb and Huttenlocher, 2004) and Power Watersheds (Couprie *et al.*, 2009). Unfortunately, most of them have limited performance in practical because the visual patterns in the real-world images are broadly diverse and ambiguous while the algorithms are developed under some particular motivations. Actually, as is shown in the Chapter 2, it is much easier for those algorithms to generate over segmentations, i.e. superpixels. However, in order to get a good image segmentation, some further treatment is required for the superpixels to be formed as the segmentation outcome (Panagiotakis *et al.*, 2013).

Notably, there is a growing trend in treating superpixels as cues to be merged through the clustering ensemble techniques. In Kim *et al.* (2010), a superpixel-based segmentation algorithm is proposed, in which the superpixel segmentations are fused by a graph model; Li *et al.* (2012) developed an efficient graph partition method, named *T-cut*, which can effectively reduce the computation complexity of bipartite-graph-based image segmentation. More recently, Wang *et al.* (2013) applied a sparse coding method to represent the superpixels in a ℓ_0 space, and achieved some impressive results by using a modified cross-adjacency matrix with the *T-Cut* algorithm.

In a wider context, it is found that the fusion of multiple cues can lead to better

segmentation, e.g., by combining color histograms, local binary patterns feature, and Bag of Words (Cheng *et al.*, 2010). Apparently a suitable representation of superpixels may improve the quality of superpixel-based image segmentation.

In this chapter, we proposed a method for improving the superpixel-based image segmentation algorithm. The experiments show that our algorithm is competitive to the state of the art, and performs better especially in the foreground-background segmentation. Figure 3.1 gives a quick comparison of our algorithm and SAS (Li *et al.*, 2012). Note that the tiger is split into different chunks by SAS, but not by our method.

(a) (b) (c)

Figure 3.1: Visual comparison of the best segmentation: (a) original image; (b) SAS; (c) our method.

The main contributions are as follows:

- We first propose a color covariance matrix as a kind of feature for superpixel, and, since the covariance matrix is a kind of tensor lying on a Riemannian manifold, we find a proper distance metric for it.
- We then propose several ways of fusing the similarity of the superpixels measured in two different feature spaces and adopt a few empirical tests for them.

For the rest of this chapter, Section 3.2 is a breif introduction of ensemble segmentation with superpixels; Section 3.3 introduces our CCM algorithm; Section 3.4 gives the details about the experiments, and Section 3.5 is the summary of the chapter.

3.2 Superpixel ensemble

The HBGF algorithm (i.e. Algorithm 2.3) is employed to model the structure of the pixel data by the given superpixel segmentations.

Given an image $I = \{p_i, \cdots, p_m\}$ and a collection of superpixel segmentations $\mathcal{S} = \{S^1, \cdots, S^N\}$, where, $S^i = \{s_1^i, \cdots, s_{K_i}^i\}$ is a superpixel segmentation that contains K_i superpixels. Obviously $\forall s_k^i, s_l^i, (k, l = 1, \cdots, K_i)$, we have, $s_k^i \cap s_l^i = \emptyset$ for $k \neq l$ and $\cup_{k=1}^{K_i} s_k^i = I$.

Let $G(V^X, V^Y, W)$ be a bipartite graph for superpixel ensemble, where, V^X and V^Y are two subset of the vertices which satisfy $V^X \cup V^Y = V$ and $V^X \cap V^Y = \emptyset$; W is the weighted cross-adjacency matrix. Similar to the original HBGF model, we set $V^Y = \cup_{i=1}^N S^i$, but for V^X, we set it as an union of pixels and superpixels, i.e. $V^X = I \cup (\cup_{i=1}^N S^i)$. With this setting, the bipartite graph is not only influenced by the relations between pixel and superpixel but also among the superpixels. Let v_i^X and v_j^Y represent vertices in V^X and V^Y respectively, then the weight w_{ij} on the edge between v_i^X and v_j^Y is defined as Eq. 3.1,

$$w_{ij} = \begin{cases} \alpha, & \text{if } v_i^X \in I \text{ and } v_i^X \in v_j^Y, \\ sim(v_i^X, v_j^Y), & \text{if } v_i^X \in S^i \text{ and } v_j^Y \in S^i \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

where α is a constant, and $sim(x, y)$ is a function that returns the similarity of input x and y. And from Eq. 3.1, W can be considered as a concatenation of two matrices, i.e.

$$W = \begin{bmatrix} W^{ps} \\ W^{ss} \end{bmatrix}, \tag{3.2}$$

where W^{ps} represents the similarities between pixel and superpixel, and W^{ss} is the superpixel-wise similarity matrix. Moreover, it always holds $|V^X| >> |V^Y|$ since the number of pixels are far more than that of superpixels. So, the spectral clustering on G can be applied via *T-cut* (i.e. Algorithm 2.4), which is efficient than SVD (Li *et al.*, 2012).

3.3 The CCM algorithm

Figure 3.2 shows the framework of the CCM algorithm. The algorithm is named as CCM because the it employs color covariance matrix of the superpixels as one feature for similarity measuring among the superpixels.

In the first step, the input image is partitioned into a few over segmentations by the superpixel algorithms. And then, the covariance descriptors are extracted from the superpixels. Thirdly, a bipartite graph is constructed based on the extracted covariance

feature. Finally, the ensemble segmentation is run by the *T-cut* algorithm. The details of feature extraction and graph construction are elaborated in the following subsections.

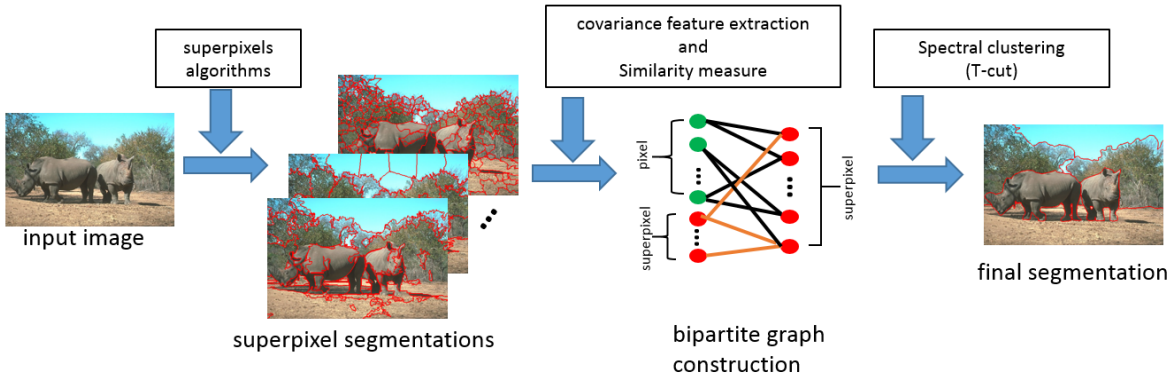


Figure 3.2: The framework of CCM.

3.3.1 Feature extraction

One of the key issues in superpixel-based segmentation is what kind of features can be extracted from superpixels. Intuitively, color is the most important cue for human to identify different objects, and in computer vision, the color space is one of the most natural ways for representing an image. Particularly, it has been shown that the *Lab* color can provide a good approximation of the color difference in human vision (Jain, 1989). Therefore in CCM, the method for extracting feature descriptors is delivered based on the *Lab* color space, but actually for other color spaces, it is also adaptive.

Let $s_i = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_R\}$ be a superpixel of R pixels in the *Lab* color space, where, $\mathbf{x}_i = (l_i, a_i, b_i)^T$ is a 3-dimension vector. The first feature we used to represent a superpixel is simply the average value of the color vectors inside the superpixel. Given a superpixel s_i, the color feature is defined as

$$\mathbf{c}_i = E(\mathbf{x}_r), \tag{3.3}$$

where $\mathbf{x}_r \in s_i$.

However, using color information alone may not be enough for generating good segmentation because the high variations of lights and contrast in the real world always make the color values unstable in the digital images. In fact, many researchers incorporate the color cue with some other cues for getting a better image segmentation. For example, Li *et al.* (2012) use the spatial cues, i.e. the neighborhoods of the superpixel. But such kind of spatial cues always fail to catch the long-range relations between the superpixels.

Different to others, we consider to use the color covariance matrix as a feature for the superpixel. The color covariance matrix of superpixel s_i is defined as

$$\Sigma_i = E((\mathbf{x}_r - \mathbf{c}_i)(\mathbf{x}_r - \mathbf{c}_i)^T), \tag{3.4}$$

where $\mathbf{x}_r \in s_i$.

Covariance matrix is a kind of tensor that lies on a smooth manifold, hence requiring a non-Euclidean distance metric. Since they are symmetric and positive semi-definite, we can use the Förstner & Moonen metric (Förstner and Moonen, 2003) given as

$$d(\Sigma_A, \Sigma_B) = \sqrt{\sum_{r=1}^{n} \ln^2 \lambda_r}, \tag{3.5}$$

where Σ_A, Σ_B are two covariance matrices of dimension $n \times n$, and $\lambda_r (r = 1, \cdots, n)$ are eigenvalues from the generalized eigenvalue problem $|\lambda\Sigma_A - \Sigma_B| = 0$.

3.3.2 The similarity measure

Another difference between the CCM and other related works is in the similarity measure of the superpixels. In (Li *et al.*, 2012), each superpixel is connected with the nearest neighborhood among its spatially adjacent superpixels, which fails to catch the relationship of those vertices that are separated by spacial distance but close in the feature space. In (Wang *et al.*, 2013), this weakness was overcome by measuring the similarity of the superpixels with their ℓ_0 sparse coding representation. However, this problem can also be solved in another way. In the CCM approach, the color covariance matrices are employed to strengthen the color representations of superpixels so that the spatial constrains can be removed.

Let d_{ij} denotes the distance between superpixel s_i and s_j. The similarity function $sim(s_i, s_j)$ between the two superpixels then is defined as follows:

$$sim(s_i, s_j) = \begin{cases} e^{-\beta \min(d_{ij}, d_{ji})} & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases} \tag{3.6}$$

where β is a coefficient of the Gaussian-like kernel, and d_{ij} is normalized into [0,1].

Because the superpixels are represented by two features and they are in two different feature spaces, one is Euclidean and the other is non-Euclidean, it is more desirable to compute the similarity separately than to concatenate them as a vector.

Let sim^C denote the similarity in Lab color space and sim^Σ be the similarity in the covariance manifold; $Sim = [sim_{ij}]$ denotes the similarity matrix over all superpixels.

There are three algorithms for fusing the similarity matrices. The first is by means of the entry-wise product (*aka* Hadamard product):

$$Sim^{\mathrm{HP}} = Sim^{C} \circ Sim^{\Sigma}. \tag{3.7}$$

The second approach is proposed by de Sa (2005), which adopts the direct matrix product to similarity matrices:

$$Sim^{\mathrm{DP}} = Sim^{c} \times Sim^{\Sigma}. \tag{3.8}$$

The third one is to combine two individual modalities by simply adding them together, which is proposed by Joachims (2003):

$$Sim^{\mathrm{AD}} = Sim^{c} + Sim^{\Sigma}. \tag{3.9}$$

In real practice, we try all three approaches and choose the one that gives the best performance as the fusing method. The overall algorithm of CCM is given in Algorithm 3.1

Algorithm 3.1 Superpixel-based Segmentation via Color Covariance Matrix

Input: An image $I = \{p_1, \cdots, p_m\}$, a collection of superpixel segmentations $\mathcal{S} = \{S^1, \cdots, S^N\}$, the cluster number k of the final clustering

Output: Final clustering $S = (s_1, ..., s_k)$

Set $V^X = I \cup (\cup_{i=1}^{N} S^i)$, $V^Y = \cup_{i=1}^{N} C^i$, $n = \sum_{i=1}^{N} K_i$;
$W = \emptyset$;
for $i = 1, \cdots, m$ **do**
  for $j = 1, \cdots, n$ **do**
    Compute the similarity of the superpixels via Eq. 3.6;
    Fuse similarities via Eq. 3.7, Eq. 3.8, or Eq 3.9;
    Compute w_{ij} via Eq. 3.1;
    $W = W \cup w_{ij}$;
  end for
end for
Apply *T-cut* to obtain S.

3.4 Experiments

3.4.1 Data sets and settings

The experiments are conducted on two public image segmentation datasets: the Berkeley Segmentation Dataset 300 (BSDS300), and its update, the Berkeley Segmentation Dataset 500 (BSDS500) (Martin *et al.*, 2001; Arbelaez *et al.*, 2011).

In order to compare the performance of the CCM and the state of the art, the parameters are set as same as those in (Li *et al.*, 2012; Wang *et al.*, 2013). Specifically, the superpixel segmentations are created by Mean Shift and F-H algorithm. There are three superpixel segmentations generated by Mean Shift with the parameters $(h_s, h_r, M) \in \{(7, 7, 100), (7, 9, 100), (7, 11, 100)\}$ where, h_s and h_r are the bandwidth parameters, and M represents the minimum size of the superpixel, and two or three superpixel segmentations produced by the F-H algorithm based on the image variance in the *Lab* color space with a given threshold; the parameters are set to be $(\sigma, c, M) \in \{(0.5, 100, 50), (0.8, 200, 100)\}$ for the two-segmentation case, or $(\sigma, c, M) \in \{(0.8, 150, 50), (0.8, 200, 100), (0.8, 300, 100)\}$ for the three-segmentation case, where σ and c are the parameters for smoothing and scale, M is the minimum size of the superpixel.

For the edge weights in Eq. 3.1, α is set to be 1×10^{-3}, and the parameter β in Eq. 3.6, we set $\beta = 20$ for all feature spaces. We also adopt a nearest-neighbor filter on the similarity matrix of the superpixels, so each superpixel is only connected to its closest neighbor in the final similarity matrix, which is the same as in (**?**Wang *et al.*, 2013).

The experiments are conducted in two parts. First, we manually set the number of segmentations K of every image to find the best performance of the algorithms for comparison. Second, we fix the segment number $K = 2$, which is considered as an foreground and background segmentation.

3.4.2 Results

We compare the CCM with the SAS(Li *et al.*, 2012) and ℓ_0-sparse (Wang *et al.*, 2013). And the evaluation is based on four popular methods, i.e. PRI (Unnikrishnan *et al.*, 2007), VoI (Meilă, 2005), GCE (Martin *et al.*, 2001), and BDE (Freixenet *et al.*, 2002). The overall performance is represented by Avg.R, i.e. the average rank. Moreover, we would like to mention that, for PRI a higher value means better, and for VoI, GCE

and BDE, the lower value is better.

Table 3.1 and Table 3.2 show the scores of the four evaluation indices with the K manually adjusted on BSDS300 and BSDS500. And, the results of $K = 2$ are listed in Table 3.3 and Table 3.4 separately. Here, we note that some of the scores of the SAS algorithm and ℓ_0-sparse representation methods are directly obtained from the reports in (Li *et al.*, 2012; Wang *et al.*, 2013).

Table 3.1: Performance over BSDS300 with K adjusted manually

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.8319	1.6849	**0.1779**	11.2900	2.5
ℓ_0-sparse	0.8355	1.9935	0.2297	**11.1955**	2.5
CCM(W_{HP})	**0.8495**	**1.6260**	0.1785	12.3034	**2.25**
CCM(W_{DP})	0.8345	2.1169	0.2341	12.0008	4.5
CCM(W_{AD})	0.8397	2.0359	0.2308	11.8868	3.25

Table 3.2: Performance over BSDS500 with K adjusted manually

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.8372	**1.6914**	**0.1813**	12.6599	2
ℓ_0-sparse	-	-	-	-	-
CCM(W_{HP})	**0.8407**	2.0399	0.2359	**10.7800**	**1**
CCM(W_{DP})	0.8275	2.5169	0.2541	11.5002	3.5
CCM(W_{AD})	0.8370	2.0490	0.2503	10.8868	2.75

Table 3.3: Performance over BSDS300 with K fixed to 2

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.6179	2.0110	0.1106	42.2877	4.25
ℓ_0-sparse	0.6270	2.0299	0.1050	**23.1298**	3
CCM(W_{HP})	**0.6312**	**1.9350**	**0.0820**	35.8760	**1.75**
CCM(W_{DP})	0.5998	2.0336	0.0892	29.1803	3.5
CCM(W_{AD})	0.6284	1.997	0.0940	24.6991	2.25

In both scenarios our method gives competitive performance. Our method ranks the first place with PRI and VoI when the cluster number K is manually set, and when K is fixed to 2, it gets the best scores in PRI, VoI, and GCE. We also examine the

Table 3.4: Performance over BSDS500 with K fixed to 2

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.6094	2.0701	0.1130	43.7731	3.5
ℓ_0-sparse	-	-	-	-	-
CCM(W_{HP})	**0.6234**	**1.9961**	**0.0870**	36.4631	**1.5**
CCM(W_{DP})	0.5961	2.0841	0.0923	28.6858	3
CCM(W_{AD})	0.6203	2.0461	0.0972	25.3790	2

performance of all three fusion methods in the experiments. The Hadamard product seems to performs the best among the three fusing schemes with the best Avg.R, but the difference is mostly marginal.

Moreover, the effects of different color spaces are also investigated. As shown in Table 3.5, the choice of the color space does not seem to be critical, since the utilization of the color covariance matrices seems to boost the performance significantly to a competitive level even for RGB and HSV. SAS, on the other hand, reports worse results in VoI and GCE when using these two color spaces compared with using *Lab*.

Table 3.5: Performance in different color spaces (on BSD300, $K = 2$)

Algorithms	PRI	VoI	GCE	BDE	Avg.R
Lab (SAS)	0.6179	2.011	0.1106	42.2877	5
RBG (SAS)	0.6189	2.0224	0.1138	42.5141	4.5
HSV (SAS)	0.6182	2.0450	0.1203	42.0903	5.5
Lab (CCM(W_{HP}))	0.6312	**1.9350**	0.0820	35.8760	2
RBG (CCM(W_{HP}))	0.6289	1.9426	**0.0815**	33.9353	2
HSV (CCM(W_{HP}))	**0.6317**	1.9549	0.0838	**30.2480**	2

The experiment results on the BSDS data sets show that the new superpixel feature extracted by a covariance matrix apparently improves the average performance of the bipartite graph based algorithm when combined with color cues. By removing the spatial constraints, our method seems to handle long-range homogeneity well, forming superpixels well aligned with object contours. Figure 3.3 shows more experimental results of our algorithm when K is set to 2. The method seems to be quite effective in foreground-background separation.

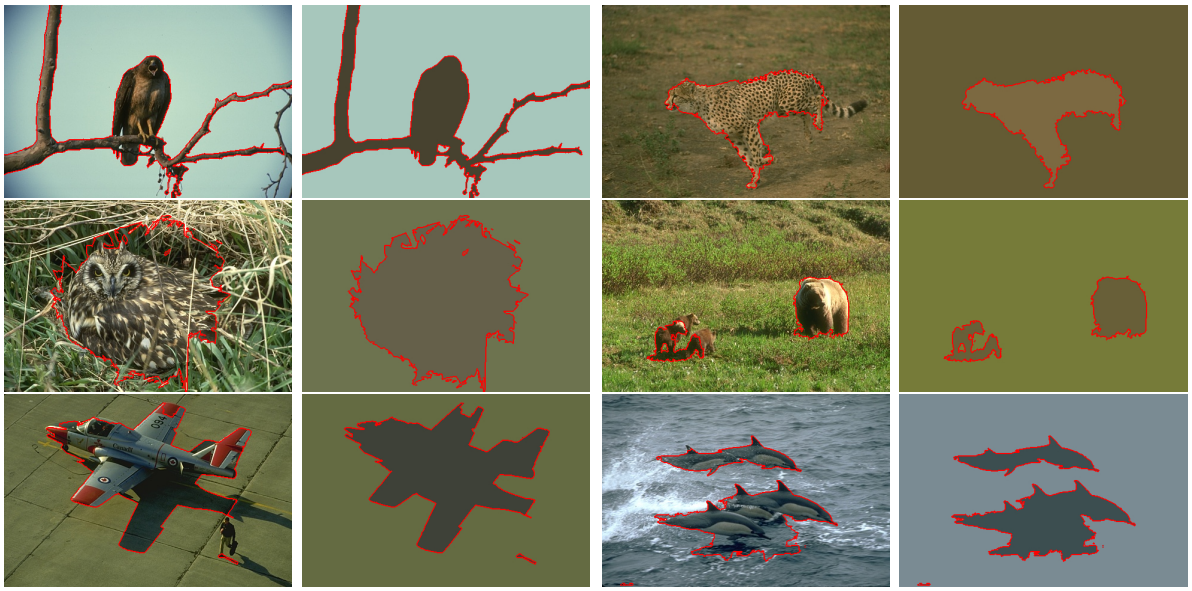


Figure 3.3: Some segmentation results of CCM ($K = 2$).

3.5 Conclusion

In this chapter, we present a superpixel-based segmentation approach that utilizes color covariance matrices to boost the performance of graph-based image segmentation. A non-Euclidean metric is employed for the covariance matrix space, and the new feature is then integrated with color information to form the affinity graph for segmentation. The empirical results show that the new approach produces better or competitive segmentation results compared with the state-of-the-art approaches. It is not sensitive to the choice of color space, different from the previous work (Li *et al.*, 2012).

But, there are two issues need to be concerned further. The first one is the methods for merging the similarity matrix of the superpixels. We would like to explore some other information fusing approaches rather than the three intuitive methods used in this chapter. The second one is about the covariance matrix itself, i.e. what kind of covariance matrix is better for superpixel representation. These issues will be discussed in the next two chapters.

Chapter 4

Improving the Color Covariance Matrix based Segmentation with Subspace Representation

4.1 Introduction

In the CCM algorithm proposed in Chapter 3, the color covariance matrix is employed to represent the superpixels, by which the similarities between the superpixels can be measured on a Riemannian manifold. The segmentation quality is improved by fusing the superpixel-wise similarities measured in the color space and the Riemannian manifold. Although the CCM algorithm shows that the covariance descriptor is a useful representation for superpixels, one thing still need to be concerned. Because different features may have different data structure, for example, the covariance descriptors of the superpixels are lying on an manifold while the *Lab* color features are points in a 3-D Euclidean space, it may not be appropriate to fuse the superpixel descriptors from different feature spaces directly in a Euclidean space. A new feature fusing method is needed for improving the CCM algorithm.

The research in subspace representation has been blooming in recent years, and some works in that region shed lights on the problem mentioned above. Actually, this problem is also called feature embedding in literature (Zhang *et al.*, 2015). Most of the proposed solutions are based on the dimensionality reduction technologies, by which the redundancy among features can be reduced while preserving the important discriminative information. Tang *et al.* (2009) modeled the relations from different features with different graphs and the common factors of the multiple graphs are extracted by

linked matrix factorization. Dong *et al.* (2014) formulated the multiple features by a multi-layer graph, but differently, they merge the different layers via the regularization on a Grassmann manifold. Zhou and Burges (2007) propose a multiple-graph merging models based on the graph random walk, and the kernel methods are also employed for fusing the information from multiple sources (Wang *et al.*, 2012; Nguyen *et al.*, 2015).

In this chapter we propose one method for fusing superpixel descriptors extracted from different feature spaces, and this method improves the performance of the CCM. Our contributions are as follows,

- we propose a multi-layer bipartite graph to formulate structure information provided by the color and the covariance descriptors of the superpixels;
- we develop an algorithm for clustering multi-layer bipartite graph.

In the rest of the chapter, Section 4.2 contains the introductions of the necessary background knowledge for elaborating our algorithms. Section 4.3 is the multi-layer bipartite graph based CCM algorithm (MBG-CCM) and Section 4.4 shows the results of the experiments. In Section 4.5, we give the conclusion.

4.2 Preliminary

4.2.1 The subspace representation

Given a graph $G(V, E)$ and let W be the adjacency matrix of G. We set the degree matrix $D = \text{diag}(W\mathbf{1})$, where $\mathbf{1}$ is a vector of ones of appropriate size. Then, the graph Laplacian of G is defined as

$$L = D - W. \tag{4.1}$$

For a graph, L is a representation of its structure, i.e the relations between the vertices. And, the spectral decomposition of L can map the graph into a Euclidean space with the structure information preserved (Von Luxburg, 2007). Moreover, if we replace the graph Laplacian with $L = D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, the eigenvectors of the k largest eigenvalues of L will still contain most structure information of the graph (Dong *et al.*, 2014).

Let $U = [u_1, \cdots, u_k]$ be a set of the first k eigenvectors of a graph Laplacian L. Since the eigenvectors are orthogonal to each other and k is smaller than the rank of L, we say U is a subspace representation of L .

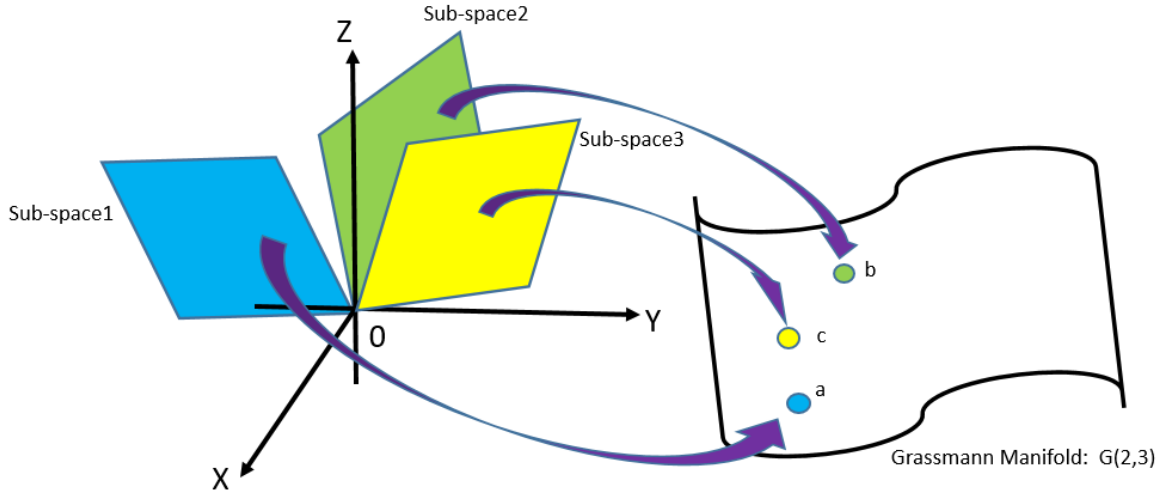


Figure 4.1: A example of Grassmann manifold $\mathcal{G}(2,3)$. The subspace representations are points on $\mathcal{G}(2,3)$.

4.2.2 Grassmann manifold

A Grassmann manifold $\mathcal{G}(k,n)$ is defined as the set of k-dimensional linear subspace of $\mathbb{R}^n$ (Hamm and Lee, 2008). Figure 4.1 demonstrates an example of Grassmann manifold $\mathcal{G}(2,3)$.

Obviously, the subspace representation of L can be considered as a point on a Grassmann manifold.

4.3 Multi-Layer graph based CCM

There are a number of low-level features can be extracted from superpixels, such as colors, covariance descriptors, etc., each of which is a source of segmentation information. And they can be formulated by a multi-layer bipartite graph, with which the structure information from each feature space is represented by a single graph layer independently.

We first introduce the clustering method of a normal multi-layer graph, then propose the clustering algorithm for the bipartite case.

4.3.1 The multi-layer graph

A multi-layer graph is used to model a graph that processes multiple views. Given a dataset $X = \{x_1, \cdots, x_n\}$; let $G(V,E)$ be a graph built on X, where V is the vertex

set, each vertex represents a point in X, and E is the set of edges, representing the relationships between the vertices.

Suppose the dataset X has M different properties which lead to M different relationships among the vertices; naturally, they can be represented by a set of weighted edges on the common set of vertices. So, a multi-layer graph G with M-layer is defined as

$$G = \{G_i\}_{i=1}^{M}, \tag{4.2}$$

where G_i is a single layer built on the i-th property and defined as

$$G_i = G(V, W_i), \tag{4.3}$$

where W_i is the weight associated to the i-th edge set E_i.

4.3.2 Clustering on multi-layer graph

The clustering on the multi-layer graph G is actually an ensemble of the clusterings on all single graph G_i. Since a graph can be represented by its subspace representation, we achieve the ensemble via a Grassmann manifold.

Let U_i be the subspace representation of $G_i \in G$, the subspace representation of a M-layer graph is written as

$$\mathcal{U} = \{U_i\}_{i=1}^{M}. \tag{4.4}$$

Since each U_i is a point on a Grassmann manifold, the fusion of the $U_i \in \mathcal{U}$ can be straightforwardly modeled as a minimization problem (Dong *et al.*, 2014),

$$U = \arg\min_{U} \sum_{i=1}^{M} f(U_i, U) \tag{4.5}$$

where U is the final representation of $\mathcal{U}$, and, $f(\cdot, \cdot)$ is the cost function. Same to Dong et al., we use the squared projection distance as the cost function, and Eq. 4.5 can be rewritten as

$$U = \arg\min_{U} \sum_{i=1}^{M} d_{proj}^2(U_i, U), \tag{4.6}$$

where $d_{proj}(\cdot, \cdot)$ is the projection distance.

Using Eq. 4.6 is based on two facts. Firstly, the spectral clustering is equal to a trace minimization problem (Dhillon *et al.*, 2004), i.e.

$$\begin{aligned} &\min_{U \in R^{n \times k}} \operatorname{tr}(U^T L U), \\ &\text{s.t. } U^T U = I. \end{aligned} \tag{4.7}$$

where n, k are the numbers of vertices and clusters respectively; $\mathrm{tr}(\cdot)$ returns the trace of the input. Secondly, projection distance is a measurement on the Grassmann manifolds which is related to trace. The defined of projection distance is

$$d_{proj}(X_1, X_2) = (\sum_{i=1}^{k} sin^2\theta_i)^{1/2} \tag{4.8}$$

where X_1, X_2 are the orthonormal matrices representing two subspaces; $\{\theta_i\}_{i=1}^{k}$ is the set of principal angles between two subspaces. So, for the squared projection distance, we have (Hamm and Lee, 2008)

$$\begin{aligned} d^2_{proj}(X_1, X_2) &= \sum_{i=1}^{k} sin^2\theta_i \\ &= k - \sum_{i=1}^{k} cos^2\theta_i \\ &= k - \mathrm{tr}(X_1 X_1^T X_2 X_2^T). \end{aligned} \tag{4.9}$$

And so, Eq. 4.6 can be written as

$$U = \arg\min_{U} \left[kM - \sum_{i=1}^{M} \mathrm{tr}(UU^T U_i, U_i^T) \right]. \tag{4.10}$$

Because U needs to satisfy both Eq. 4.7 and Eq. 4.10, we can combine them together as follows

$$\begin{aligned} &\min_{U \in R^{n\times k}} \sum_{i=1}^{M} \mathrm{tr}(U^T L_i U) + \gamma \left[kM - \sum_{i=1}^{M} \mathrm{tr}(UU^T U_i U_i^T) \right], \\ &\text{s.t. } UU^T = I, \end{aligned} \tag{4.11}$$

where L_i is the i-th Laplacian of G and U_i is the respective subspace representation, and γ is a weight parameter that balances the effects of two terms in the equation. Moreover, by ignoring the constant term in Eq. 4.11 and considering the fact that $\mathrm{tr}(X) = \mathrm{tr}(X^T)$, we have

$$\begin{aligned} &\min_{U \in R^{n\times k}} tr \left[U^T (\sum_{i=1}^{M} L_i - \gamma \sum_{i=1}^{M} U_i U_i^T) U \right], \\ &\text{s.t. } U^T U = I. \end{aligned} \tag{4.12}$$

If we set

$$L_{mod} = \sum_{i=1}^{M} L_i - \gamma \sum_{i=1}^{M} U_i U_i^T, \tag{4.13}$$

then, the solution of Eq. 4.12 is the first k eigenvectors of the modified Laplacian L_{mod}.

4.3.3 The multi-layer bipartite graph

Given a multi-layer graph G, if each layer G_i is a bipartite graph, then we say G is a multi-layer bipartite graph.

Let $I = \{p_1, \cdots, p_m\}$ be an image with m pixels, and $\mathcal{S} = \{S^1, \cdots, S^N\}$ be a collection of superpixel clusterings of the image, where $S^i = \{s_1^i, \cdots, s_{k_i}^i\}$ is the i-th superpixel clustering with k_i superpixels. Suppose there are M different features extracted from the superpixels, and let G be the multi-layer bipartite graph, which is written as

$$G = \{G_i\}_{i=1}^M = \{G(V^X, V^Y, W_i)\}_{i=1}^M, \tag{4.14}$$

where W_i is the cross-adjacency matrix corresponding to the i-th feature. And each single layer G_i is constructed by setting $V^X = I \cup \mathcal{S}$ and $V^Y = \mathcal{S}$.

One straightforward way to merge the $\{G_i\}$ is treating every G_i as a normal graph, i.e. extending W_i into a $(m+n) \times (m+n)$ matrix, so Eq. 4.13 can directly work on G. However, this could be intractable in image segmentation because the huge number of pixel results in high complexity both in computation and memory.

Actually, the spectral clustering of each $G_i \in G$ can be done via singular value decomposition (SVD) of the normalized W_i, and the clustering of V^X can be obtained from the clustering of V^Y (Li *et al.*, 2012; Dhillon, 2001). Fortunately, with the following lemma, this is also applicable for the multi-lay bipartite graph.

Lemma 4.3.1. *Given a matrix* $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$, *let* $D_X = diag(A)$ *be a* $m \times m$ *diagonal matrix with the* i*-th entry of the main diagonal is the sum of the* i*-th row of* A *(i.e.* $\sum_{j=1}^n a_{ij}$*), and* $D_Y = diag(A^T)$ *be a* $n \times n$ *diagonal matrix with the* j*-th entry of the main diagonal is the sum of* j*-th column of* A *(i.e.* $\sum_{i=1}^m a_{ij}$*), then, it holds* $D_Y = diag(A^T D_X^{-1} A)$.

Proof. The proof is straightforward. $A^T D_X^{-1} A =$

$$\begin{bmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & & \vdots \\ a_{1n} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} (\sum_{j=1}^n a_{1j})^{-1} & & \\ & \ddots & \\ & & (\sum_{j=1}^n a_{mj})^{-1} \end{bmatrix} \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{a_{11}^2}{\sum_{j=1}^n a_{1j}} + \cdots + \frac{a_{m1}^2}{\sum_{j=1}^n a_{mj}} & \cdots & \frac{a_{11}a_{1n}}{\sum_{j=1}^n a_{1j}} + \cdots + \frac{a_{m1}a_{mn}}{\sum_{j=1}^n a_{mj}} \\ \vdots & & \vdots \\ \frac{a_{1n}a_{11}}{\sum_{j=1}^n a_{1j}} + \cdots + \frac{a_{mn}a_{m1}}{\sum_{j=1}^n a_{mj}} & \cdots & \frac{a_{1n}^2}{\sum_{j=1}^n a_{1j}} + \cdots + \frac{a_{mn}^2}{\sum_{j=1}^n a_{mj}} \end{bmatrix},$$

$$\text{so, } \operatorname{diag}(A^T D_X^{-1} A) = \begin{bmatrix} \sum_{i=1}^{m} a_{i1} \frac{\sum_{j=1}^{n} a_{ij}}{\sum_{j=1}^{n} a_{ij}} & & \\ & \ddots & \\ & & \sum_{i=1}^{m} a_{in} \frac{\sum_{j=1}^{n} a_{ij}}{\sum_{j=1}^{n} a_{ij}} \end{bmatrix} = D_Y. \qquad \square$$

Let W_{in} be the i-th normalized cross-adjacency matrix, according to the definition we have

$$W_{in} = D_{iX}^{-\frac{1}{2}} W_i D_{iY}^{-\frac{1}{2}}, \tag{4.15}$$

where $D_{iX} = \operatorname{diag}(W_i \mathbf{1})$ and $D_{iY} = \operatorname{diag}(W_i^T \mathbf{1})$, and $\mathbf{1}$ is a vector of ones in proper size; $\operatorname{diag}(\cdot)$ is a diagonal matrix whose nonzero entries represented by"$(\cdot)$". Then, we define W_{iYn}, whose eigenvectors are the right singular vectors of W_{in}:

$$W_{iYn} = W_{in}^T W_{in} = D_{iY}^{-\frac{1}{2}} W_i^T D_{iX}^{-1} W_i D_{iY}^{-\frac{1}{2}}, \tag{4.16}$$

If we set $W_{iY} = W_i^T D_{iX}^{-1} W_i$, from Lemma 4.3.1, we know $D_{iY}^{-\frac{1}{2}} = \operatorname{diag}(W_i^T D_{iX}^{-1} W_i)$; so,

$$L_{iYn} = I - W_{iYn} \tag{4.17}$$

is the normalized graph Laplacian of $G_{iY}(V^Y, W_{iY})$. Therefore, for a multi-layer bipartite graph G, the layers can be merged by Eq. 4.13 with the Laplacian in Eq. 4.17. Because the pixel-superpixel relations (i.e. W^{ps} in Eq. 3.2) in each graph layer are the same, the clustering of V^X can be obtained from the clustering of V^Y by the *T-cut*. Algorithm 4.1 shows the details.

Algorithm 4.1 MBG-CCM

Input: A set of weighted cross-adjacency matrix $\{W_i\}_{i=1}^{M}$ of multi-layer bipartite graph G, merging weight parameter γ, number of clusters k;

Output: A final clustering $\mathcal{C} = \{C_1, \cdots, C_n\}$

1: **for** $i = 1, \cdots, M$ **do**
2: Convert W_i into W_{iYn} by Eq. 4.16
3: Compute the normalized Laplacian L_i of G_i by Eq. 4.17.
4: Compute the first k eigenvectors of L_i as U_i.
5: **end for**
6: Compute the merged Laplacian L_{mod}^Y by Eq. 4.13.
7: Compute the first k eigenvectors of L_{mod}^Y as U.
8: Apply the *T-cut* (i.e. Algorithm 2.4) to obtain the final clustering $\mathcal{C}$.

4.4 Experiments

4.4.1 Data sets and settings

The experiments are conducted on two public image segmentation datasets: the Berkeley Segmentation Dataset 300 (BSDS300), and its update, the Berkeley Segmentation Dataset 500 (BSDS500) (Martin *et al.*, 2001; Arbelaez *et al.*, 2011).

For comparision purpose, we set the parameters to the same values as those used in CCM. Specifically, the superpixel segmentations are created by Mean Shift and F-H algorithm with the same parameter settings. There are three superpixel segmentations generated by Mean Shift with the parameters $(h_s, h_r, M) \in \{(7, 7, 100), (7, 9, 100), (7, 11, 100)\}$ where, h_s and h_r are the bandwidth paramters, and M represents the minimum size of the superpixel, and two or three superpixel segmentations produced by F-H algorithm based on the image variance in the *Lab* color space with a given threshold; the parameters are set to be $(\sigma, c, M) \in \{(0.5, 100, 50), (0.8, 200, 100)\}$ for the two-segmentation case, or $(\sigma, c, M) \in \{(0.8, 150, 50), (0.8, 200, 100), (0.8, 300, 100)\}$ or the three-segmentation case, where σ and c are the parameters for smoothing and scale, M is the minimum size of the superpixel.

For MBG-CCM, each single bipartite graph layer is constructed by following the Algorithm 3.1 in Chapter 3. The edge weights parameter α is set to be 1×10^{-3} and the scale parameter β is set to be 20 for all feature spaces. In addition, the γ in Eq. 4.13 is set to $\gamma = 1$.

The experiments are conducted in two parts. First, we manually set the number of segmentations K of every image to find the best performance of the algorithms for comparison. Second, we fix the segment number $K = 2$, which is considered as a way for foreground-background segmentation.

4.4.2 Results

We compare the performance of MBG-CCM with CCM. And the evaluation is based on four popular methods, i.e. PRI (Unnikrishnan *et al.*, 2007), VoI (Meilă, 2005), GCE (Martin *et al.*, 2001), and BDE (Freixenet *et al.*, 2002). The overall performance is represented by Avg.R, i.e. the average rank.

Table 4.1 and Table 4.2 show the results of "foreground-background" segmentation (i.e. $K = 2$). In this case, the scores of MBG-CCM rank first in PRI on both data sets and second in VoI and BDE with performance quite close to the best one.

And the Avg.R score of MBG-CCM is equal to CCM/HP. Moreover, Table 4.3 and Table 4.4 show the performance of K manually adjusted. MBG-CCM also performs competitively. Figure 4.2 gives more visual results of MBG-CCM algorithm.

Table 4.1: Performance over the BSD300 with K fixed to 2

Algorithms	PRI	VoI	GCE	BDE	Avg.R
CCM/HP	0.631	**1.935**	**0.082**	35.876	2
CCM/DP	0.599	2.033	0.089	29.180	3.75
CCM/AD	0.628	1.997	0.094	24.699	2.75
MBG-CCM	**0.641**	2.018	0.104	**21.426**	2

Table 4.2: Performance over the BSD500 with K fixed to 2

Algorithms	PRI	VoI	GCE	BDE	Avg.R
CCM/HP	0.623	**1.996**	**0.087**	36.463	2
CCM/DP	0.596	2.084	0.092	28.685	3.75
CCM/AD	0.620	2.054	0.097	25.379	2.75
MBG-CCM	**0.635**	2.067	0.108	**21.610**	2

Table 4.3: Performance over the BSD300 with K manually adjusted

Algorithms	PRI	VoI	GCE	BDE	Avg.R
CCM/HP	**0.8495**	**1.6260**	**0.1785**	12.3034	1.75
CCM/DP	0.8345	2.1169	0.2341	12.0008	3.75
CCM/AD	0.8397	2.0359	0.2308	11.8868	2.75
MBG-CCM	0.8421	2.0220	0.2231	**11.5773**	1.75

4.5 Conclusion

We have presented a novel approach, MBG-CCM, for improving the CCM. In the MBG-CCM, we employ a multi-layer bipartite graph for modeling the segmentation information provided by superpixel features extracted from different feature spaces and merge the different graph layers via a Grassmann manifold.

This algorithms is motivated by the fact that the cross-adjacency matrices of a multi-layer bipartite graph can be converted into a set of positive semidefinite matrices

Table 4.4: Performance over the BSD500 with K manually adjusted

Algorithms	PRI	VoI	GCE	BDE	Avg.R
CCM/HP	0.8407	**2.0399**	0.2359	**10.7800**	1.5
CCM/DP	0.8275	2.5169	0.2541	11.5002	4
CCM/AD	0.8370	2.0490	0.2503	10.8868	3.5
MBG-CCM	**0.8418**	2.0430	**0.2263**	11.165	1.75

Figure 4.2: More visual results of MBG-CCM on "foreground background" segmentation

so that it can be decomposed into eigenvectors and eigenvalues, by which the subspace representations of the matrices can be found. And the algorithm provides a strong theoretical support for using different features in the original CCM.

The experiment results show that the performance of MBG-CCM is competitive to CCM. And if there are multiple features, the MBG-CCM is undoubtedly a better option than trying the matrix operators one by one as in the original CCM given in the previous chapter.

There is, however, a shortcoming of the MBG-CCM. It needs to compute the eigenvectors for each graph layer, which makes the computation cost higher than CCM. Actually in both MBG-CMM and CMM, the final segmentation is carried out by spectral clustering techniques, so for them, computing the eigenvectors is an inevitable procedure. Moreover, for spectral clustering, an exact number of clusters should be given as prior knowledge. But this may be inapplicable in many image segmentation applications. To solve these problems, we need to develop a new ensemble segmentation method, which can formulate the pixel-superpixel relations without bipartite graph and determine the number of clusters automatically.

Chapter 5

Low-Rank Representation for Covariance Descriptor

5.1 Introduction

Apart from the feature fusing problem, there is another issue in the CCM that needs to be concerned about, i.e. how to find a "good" covariance descriptor that produces the stable performance. Actually, this is not only an issue for CCM but all the segmentation algorithms that take handcrafted covariance descriptors as features. Similar to MBG-CCM in Chapter 4, we also plan to convert this issue into an embedding problem. Specifically, we aim to represent the covariance descriptors in some subspace so that the discriminative information should be kept while the redundancies are removed.

The low-rank representation (LRR) is one of the techniques that may solve the problem mentioned above. LRR was proposed for finding a robust subspace representation for the data represented in the linear feature spaces. Since the linear space is the most common choice for data presentation, the application of LRR involves numerous research fields, such as machine learning, and computer vision etc. (Liu *et al.*, 2010). For different applications, the LRR algorithm is developed by different motivations (Liu *et al.*, 2013). In some works, the subspaces are modeled as a mixture of Gaussian distributions, and the data structure can be obtained by the parameter estimation of the mixture Gaussian model (Gruber and Weiss, 2004; Ho *et al.*, 2003; Fischler and Bolles, 1981). And, some researchers proposed an algebraic way to model the data with LRR and showed that the LRR is a generalized principal component analysis problem (Ma *et al.*, 2008; Wright *et al.*, 2009). Moreover, an augmented Lagrange multipliers (ALM) method is proposed by Lin *et al.* (2010) to solve the LRR

model. Fu *et al.* (2015) extended the LRR model to the Riemannian manifold, which is nonlinear.

In this chapter we propose a low-rank representation method for the covariance descriptors extracted from superpixels. Our contributions are as follows,

- we propose a LRR model to find the subspace structure of the covariance features;
- we improve the CCM algorithm by measuring the similarities of the superpixels with LRR.

In the rest of the chapter, Section 5.2 gives the introductions of the necessary background knowledge for elaborating our algorithms. Section 5.3 is the low-rank representation based CCM algorithm (LRR-CCM). Section 5.4 shows the results of the experiments; in Section 5.5, we give the conclusion.

5.2 Preliminary

5.2.1 Low-Rank representation

The low-rank representation (LRR) can be considered as a generalized principle component analysis (PCA) problem. One basic assumption of the LRR theory is that the given high-dimensional data lie near a lower-dimensional linear subspace. Mathematically, given a dataset $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ and $\mathbf{x}_i$ be a d-dimension column vector, i.e. $\mathbf{x}_i \in \mathbb{R}^d$; suppose X is sampled from a subspace, then, X can be decomposed into

$$X = X_0 + E \tag{5.1}$$

where X_0 is the origin of X and holds $\text{rank}(X_0) < \text{rank}(X)$, and, E is a matrix representing the difference between X_0 and X, also called corruption. The goal of LRR is to estimate the low-dimensional subspace efficiently and accurately. However, modeling this problem depends on the intrinsic structure of the dataset X.

If the corruption is caused by additive i.i.d. Gaussian noise with small magnitude, then, Eq. 5.1 can be modeled as an optimization problem, i.e.

$$\begin{aligned} &\min ||E||_F, \\ &\text{s.t. } \text{rank}(D) \leq k, \text{and, } X = D + E, \end{aligned} \tag{5.2}$$

where D is the low-rank representation of X, k is the target dimension of the subspace, and $||\cdot||_F$ is the Frobenius norm. Actually, Eq. 5.2 is equivalent to a PCA problem (Lin *et al.*, 2010).

But Eq. 5.2 will fail to find the proper $\hat{D}$ when there exists large corruption in X, even the corruption affects only a few of the entities. In this case, Eq. 5.1 can be solved by the following minimization problem (Lin *et al.*, 2010; Wright *et al.*, 2009), i.e.

$$\begin{aligned} &\min_{D,E} ||D||_* + \lambda||E||_1, \\ &\text{s.t. } X = D + E, \end{aligned} \tag{5.3}$$

where $||\cdot||_*$ is the nuclear norm of a matrix (i.e. the sum of its sigular values), $||\cdot||_1$ is the sum of absolute values of matrix entries, and λ is a positive weighting parameter.

In (Liu *et al.*, 2013), a more generalized version of Eq. 5.3 is given, which is based on the fact that many real-world datasets contain multiple subspace structures. Let $\mathcal{S} = \{\mathcal{S}_1, \cdots, \mathcal{S}_k\}$ is a set of subspaces, and we assume X is drawn from a union of these subspaces, denoted as $\mathcal{S} = \cup_{i=1}^{k} \mathcal{S}_i$; let $A = [A_1, \cdots, A_k]$ be a "dictionary" that linearly spans the data space, and A_i is the dictionary for the i-th subspace, then, Eq. 5.1 can be modeled as

$$\begin{aligned} &\min_{Z,E} ||Z||_* + \lambda||E||_\ell, \\ &\text{s.t. } X = AZ + E, \end{aligned} \tag{5.4}$$

where $||\cdot||_*$ is the nuclear norm, and,

$$Z = \begin{bmatrix} Z_1 & & & \\ & Z_2 & & \\ & & \ddots & \\ & & & Z_k \end{bmatrix}$$

is called the low-rank representation of X; $\lambda > 0$ is a parameter and $||\cdot||_\ell$ represents some regularization strategy for modeling the noise, such as the squared Frobenius norm. And, we note that if set $A = I$ and $||\cdot||_\ell$ to be $||\cdot||_1$, then, Eq. 5.4 is equivalent to Eq. 5.3.

Eq. 5.4 can be solved by the augmented Lagrange multiplier (ALM) method proposed in (Liu *et al.*, 2013). First, Eq. 5.4 is rewritten into the following equivalent formation, i.e.

$$\begin{aligned} &\min_{Z,E,J} ||J||_* + \lambda||E||_{2,1}, \\ &\text{s.t. } X = AZ + E, Z = J, \end{aligned} \tag{5.5}$$

where $||E||_{2,1} = \sum_{j=1}^{n} \sqrt{\sum_{i=n}^{n}([E]_{ij})^2}$ is called $\ell_{2,1}$ norm. Then, the augmented Lagrange function is written as

$$\begin{aligned}\mathcal{L} &= ||J||_* + \lambda||E||_{2,1} + \text{tr}(Y_1^T(X - AZ - E)) \\ &+ \text{tr}(Y_2^T(Z - J)) + \frac{\mu}{2}(||X - AZ - E||_F^2 + ||Z - J||_F^2)\end{aligned} \tag{5.6}$$

where $\text{tr}(\cdot)$ is the trace operator, Y_1 and Y_2 are the Lagrange multipliers, and $\mu > 0$ is the penalty parameter. So, Eq. 5.5 can be solved by iteratively updating one variable while fixing the others each time until the convergence conditions are met. The inexact ALM method is show in Algorithm 5.1, which is a variation of ALM method for an unsmooth object function (Liu *et al.*, 2013).

Algorithm 5.1 Inexact ALM for Eq 5.4

Input: A data set $X = \{\mathbf{x}_1, \cdots, \mathbf{x}_m\}$, parameter λ, dictionary A.

Output: The lowest-rank representation Z^*

1: Initialization: $Z = J = 0$, $E = 0$, $Y_1 = 0$, $Y_2 = 0$, $\mu = 10^{-6}$, $\mu_{max} = 10^6$, $\rho = 1.1$, and $\epsilon = 10^{-8}$
2: **while** not converged **do**
3: Fix the others and update J by $J = \text{argmin}\frac{1}{\mu}||J||_* + \frac{1}{2}||J - (Z + Y_2/\mu)||_F^2$.
4: Fix the others and update Z by $Z = (I + A^TA)^{-1}(A^T(X - E) + J + (A^TY_1 - Y_2)/\mu)$
5: Fix the others and update E by $E = \text{argmin}\frac{\lambda}{\mu}||E||_{2,1} + \frac{1}{2}||E - (X - AZ + Y_1/\mu)||_F^2$.
6: Update Y_1 and Y_2 by $Y_1 = Y_1 + \mu(X - AZ - E)$, $Y_2 = Y_2 + \mu(Z - J)$
7: Update μ by $\mu = \min(\rho\mu, \mu_{max})$
8: Check convergence conditions, $||X - AZ - E||_\infty < \epsilon$ and $||Z - J||_\infty < \epsilon$.
9: **end while**

5.2.2 Covariance descriptor and collinearity

Covariance Descriptor and Sym_d^+

Covariance descriptor maps feature functions to a symmetric positive define matrix space.

Specifically, let $\mathbf{F} = (\mathbf{f}_1, ..., \mathbf{f}_d)^T$ be a feature array, where $\mathbf{f}_i$ is a vector whose entries are the observations of the i-th feature. A covariance descriptor is the covariance matrix of $\mathbf{F}$, which is defined as

$$cov(\mathbf{F}) = \left[E((\mathbf{f}_i - \mu_i)^T(\mathbf{f}_j - \mu_j))\right]_{d \times d}, \tag{5.7}$$

where μ_i is the mean of the *i-th* feature $\mathbf{f}_i$, $[\cdot]_{d\times d}$ indicates an $d \times d$ matrix. Apparently, different sets of $\mathbf{f}_i$ generate different $cov(\mathbf{F})$, which brings different performance.

Moreover, since the $d \times d$ covariance matrix is symmetric and semi-positive definite, the space of $d \times d$ covariance matrix is a convex cone in the d^2-dimensional Euclidean space, i.e. a manifold embedding in d^2-dimensional Euclidean space, written as Sym_d^+.

Collinearity

Collinearity (or multi-collinearity) is a term from statistics, which refers to a linear association between two (or more) variables. Specifically, given a feature array $\mathbf{F}$, suppose there exists a set of not-all-zero scalar $\lambda_1, ..., \lambda_n$ that makes the following equation holds

$$\lambda_1\mathbf{f}_1 + \lambda_2\mathbf{f}_2 + \cdots + \lambda_n\mathbf{f}_n + u = 0. \tag{5.8}$$

If $u = 0$, $\mathbf{F}$ is perfect multi-collinearity; while if $u \sim N(0, \sigma)$, $\mathbf{F}$ is nearly multi-collinearity.

In image segmentation, this multi-collinearity phenomenon is common when building the covariance descriptors. For example, if we use the RGB value and intensity value as two features for covariance descriptor construction, the covariance matrix generated by Eq.5.7 is not full rank. Because the intensity value can be converted from the RGB value via a linear transformation, the covariance matrix generated by Eq.5.7 is not full rank. This means there are redundant entries and noises in the covariance descriptor.

5.3 LRR based CCM

In CCM, a color covariance matrix is used as a descriptor of the superpixels. Obviously, it is not the only covariance descriptor available for superpixels. By using different covariance descriptors, the performance of CCM may vary. In many applications, the most suitable covariance descriptor are often chosen in an empirical way, i.e. trying different selections of them and taking the one that gives the best performance (Habiboğlu *et al.*, 2012; Kviatkovsky *et al.*, 2013). This may work in practice but lack theoretical support. In this section, the LRR is used to reduce the noises in the covariance descriptor set.

5.3.1 Refine covariance descriptors with LRR

The low-rank representation (LRR) is proposed for finding a stable and compact representation for a given dataset, which has been proved an efficient method for noise reduction in Euclidean space (Candès *et al.*, 2011; Wright *et al.*, 2009; Ganesh *et al.*, 2009; Chen and Yang, 2014; Liu and Yan, 2011). And recently, it has been extended into the non-Euclidean space, such as Riemannian manifold (Fu *et al.*, 2015; Wang *et al.*, 2015a,b).

The covariance descriptors are the points lying on a Sym_d^+, and in this case, the *Frobenius* norm is used as the metric for it. So, the Sym_d^+ is embedded into the d^2-dimensional Euclidean space. Although this *Frobenius* metric is not geodesic which may loss the intrinsic structure of the dataset in embedding, it allows to apply all the methods from Euclidean space to the manifold directly.

LRR for covariance matrices

Given a set of covariance descriptors $\mathcal{X} = \{X_1, \cdots, X_n\}$, where X_i is a covariance matrix of size $d \times d$. If we stack the X_i in a third dimension, then, $\mathcal{X}$ become a 3-order tensor, i.e. a cube. By the *Frobenius* metric, we can embed $\mathcal{X}$ into the d^2-dimensional Euclidean space, so the LRR model, Eq.5.4, is written as follows,

$$\begin{aligned} &\min_{E,Z} \|E\|_F^2 + \lambda \|Z\|_*, \\ &s.t. \quad \mathcal{X} = \mathcal{X}_{\times_3} Z + E, \end{aligned} \tag{5.9}$$

where $\|\cdot\|_F$ is the *Frobenius* norm; $\|\cdot\|_*$ is the *nuclear* norm; λ is the balance parameter; $\times_3$ means mode-3 multiplication of a tensor and matrix (Kolda and Bader, 2009). Eq. 5.9 can be solved via augment Lagrangian multiplier (ALM) and the solution is as follows (Wang *et al.*, 2015b).

For the error term E, we have $\|E\|_F^2 = \|\mathcal{X} - \mathcal{X}_{\times 3} Z\|_F^2$, and we can rewrite $\|E\|_F^2$ as,

$$\|E\|_F^2 = \sum_i^N \|E_i\|_F^2, \tag{5.10}$$

where $E_i = X_i - \sum_j^N z_{ij} X_j$, i.e. the *i-th* slice of E. Note that for matrix A, it holds

$\|A\|_F^2 = \mathrm{tr}(A^T A)$, and X_i is symmetric, so, Eq. 5.10 can be expanded as

$$\begin{aligned}
\|E_i\|_F^2 &= \mathrm{tr}[(X_i - \sum_j^N z_{ij} X_j)^T (X_i - \sum_j^N z_{ij} X_j)] \\
&= \mathrm{tr}(X_i^T X_i) - \mathrm{tr}(X_i^T \sum_j^N z_{ij} X_j) - \mathrm{tr}(\sum_j^N z_{ij} X_j^T X_i) \\
&+ \mathrm{tr}(\sum_{j_1}^N z_{ij_1} X_{j_1}^T \sum_{j_2}^N z_{ij_2} X_{j_2}) \\
&= \mathrm{tr}(X_i X_i) - 2\mathrm{tr}(\sum_j^N z_{ij} X_i X_j) + tr(\sum_{j_1,j_2}^N z_{ij_1} z_{ij_2} X_{j_1} X_{j_2}).
\end{aligned}$$

Let Δ be a symmetric matrix of size $N \times N$, whose entries are $\Delta_{ij} = \Delta_{ji} = \mathrm{tr}(X_i X_j)$ and $P = \Delta^{\frac{1}{2}}$. Because X_i is a symmetric matrix, Δ_{ij} can be written as $\Delta_{ij} = \mathrm{vec}(X_i)^T \mathrm{vec}(X_j)$, where $\mathrm{vec}(\cdot)$ is an operator that vectorized a matrix. As a Gram matrix, Δ is positive semidefinite. So, we have

$$\begin{aligned}
\|E_i\|_F^2 &= \Delta_{ii} - 2\sum_{j=1}^N z_{ij}\Delta_{ij} + \sum_{j_1}^N \sum_{j_2}^N z_{ij_1} z_{ij_2} \Delta_{j_1 j_2} \\
&= \Delta ii - 2\sum_{j=1}^N z_{ij}\Delta_{ij} + \mathbf{z}_i \Delta \mathbf{z}_i^T.
\end{aligned}$$

For $\Delta = PP^T$,

$$\begin{aligned}
\|E\|_F^2 &= \sum_{i=1}^N \Delta_{ii} - 2\mathrm{tr}[Z\Delta] + \mathrm{tr}[Z\Delta Z^T] \\
&= C + \|ZP - P\|_F^2,
\end{aligned}$$

where C is a constant. The optimization Eq.5.9 is equivalent to:

$$\min_Z \|ZP - P\|_F^2 + \lambda \|Z\|_*. \tag{5.11}$$

We transform the Eq.5.11 into an equivalent formulation

$$\begin{aligned}
&\min_Z \frac{1}{\lambda}\|ZP - P\|_F^2 + \|J\|_*, \\
&\text{s.t.} \qquad J = Z.
\end{aligned} \tag{5.12}$$

Then by ALM (Argument Lagrange Multiplier), we have

$$\min_{Z,J} \frac{1}{\lambda}\|ZP - P\|_F^2 + \|J\|_* + < Y, Z - J > + \frac{\mu}{2}\|Z - J\|_F^2, \tag{5.13}$$

where Y is the Lagrange coefficient; λ and μ are scale parameters. Eq.5.13 can be solved by the following two subproblems (Lin *et al.*, 2010):

$$J_{k+1} = \min_{J}(\|J\|_* + < Y, Z_k - J > + \frac{\mu}{2}\|Z_k - J\|_F^2)$$

and,

$$Z_{k+1} = \min_{Z}(\frac{1}{\lambda}\|ZP - P\|_F^2 + < Y, Z - J_k > + \frac{\mu}{2}\|Z - J\|_F^2).$$

Fortunately according to (Cai *et al.*, 2010), the solutions for the above subproblems have the following close forms:

$$J = \Theta(Z + \frac{Y}{\mu}),$$

$$Z = (\lambda\mu J - \lambda Y + 2\Delta)(2\Delta + \lambda\mu I)^{-1},$$

where $\Theta(\cdot)$ is the singular value thresholding operator (Cai *et al.*, 2010). Thus, by iteratively updating J and Z until the converge conditions are satisfied, a solution for Eq.5.9 can be found.

5.3.2 LRR-CCM algorithm

In the graph construction step of the CCM algorithm, the superpixels-wise similarity matrix W^{ss} in Eq. 3.2 is obtained by measure the similarity between the superpixels within same superpixel segmentation. Let $S^i = \{s_1^i, \cdots, s_{K_i}^i\}$ be a set of covariance descriptors of the i-th superpixel segmenation, it is easy to see S^i is a 3-order tensor. Thus, we can get the LRR of S^i by solving Eq. 5.9, i.e. using Algorithm 5.1. Let Z^i be the LRR coefficient matrix of S^i and $\widetilde{U}_i$ be the row-normalized singular vectors of Z^i. The same as in (Ma *et al.*, 2007), we define a similarity matrix for the superpixels as

$$Sim_{K_i \times K_i} = (\widetilde{U}_i \widetilde{U}_i^T)^2. \tag{5.14}$$

and, the entry at (m, n) of matrix Sim is the similarity between superpixel descriptor s_m^i and s_n^i, written as $Sim(m, n)$. By setting the entries of W^{ss} to the respective $Sim(m, n)$, the superpixel-wise similarity matrix can be obtained. The LRR-CCM algorithm is shown in Algorithm 5.2.

5.4 Experiments

5.4.1 Data sets and settings

The experiments are conducted on two public image segmentation datasets: the Berkeley Segmentation Dataset 300 (BSDS300), and its update, the Berkeley Segmentation

Algorithm 5.2 The LRR-CCM Algorithm

Input: An image I, a collection of superpixel segmentations $\mathcal{S}$, the number of clusters k;

Output: A final clustering $\mathcal{C} = \{C_1, \cdots, C_n\}$

1: Compute the LRR (i.e. Z) for every superpixel segmentation by Algorithm 5.2.
2: Build the bipartite graph G via Algorithm 3.1, in which the superpixel-wise similarity matrix W^{ss} obtained by Eq.5.14.
3: Apply Algorithm 2.4 (i.e. *T-cut*) on G and obtain the final clustering $\mathcal{C}$.

Dataset 500 (BSDS500) (Martin *et al.*, 2001; Arbelaez *et al.*, 2011).

For comparision purpose, we set the parameters to the same values as those used in CCM. Specifically, the superpixel segmentations are created by Mean Shift and the F-H method with the same parameter settings. There are three superpixel segmentations generated by Mean Shift with the parameters $(h_s, h_r, M) \in \{(7, 7, 100), (7, 9, 100), (7, 11, 100)\}$ where h_s and h_r are the bandwidth paramters, and M represents the minimum size of the superpixel, and two or three superpixel segmentations produced by F-H method based on the image variance in the *Lab* color space with a given threshold; the parameters are set to be $(\sigma, c, M) \in \{(0.5, 100, 50), (0.8, 200, 100)\}$ for the two-segmentation case, or $(\sigma, c, M) \in \{(0.8, 150, 50), (0.8, 200, 100), (0.8, 300, 100)\}$ or the three-segmentation case, where σ and c are the parameters for smoothing and scale, M is the minimum size of the superpixel.

For LRR-CCM, the parameter λ in Eq. 5.9 is chosen by a grid search among $\{1, 0.1, 0.01, 0.001\}$ for every image, i.e. we select the value that gives the highest performance.

Moreover, for the LRR based CCM algorithm, three different covariance descriptors are used, which include

- CovI : $[R, G, B]$,
- CovII : $[R, G, B, I, \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}, \frac{\partial I}{\partial^2 x}, \frac{\partial I}{\partial^2 y}]$,
- CovIII :$[R, G, B, \frac{\partial R}{\partial x}, \frac{\partial R}{\partial y}, \frac{\partial G}{\partial x}, \frac{\partial G}{\partial y}, \frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}, \frac{\partial R}{\partial^2 x}, \frac{\partial R}{\partial^2 y}, \frac{\partial G}{\partial^2 x}, \frac{\partial G}{\partial^2 y}, \frac{\partial B}{\partial^2 x}, \frac{\partial B}{\partial^2 y}]$.

From CovI to CovIII, the dimensionality of the covariance descriptor is increasing. For example, CovI contains the patterns in the R, G, B channels, while in CovIII, the patterns of their derivatives are also included. This means the covariance descriptors

Table 5.1: Performance of LRR with different covariance descriptors

Algorithms	PRI	VoI	GCE	BDE	Avg.R
CovI+LRR	0.8454	1.7564	**0.1885**	13.0427	2
CovII+LRR	**0.8499**	**1.7418**	0.1915	12.7635	1.5
CovIII+LRR	0.8451	1.7698	0.1932	**12.4837**	2.5

become more discriminative. But since the partial derivatives are directly computed from other contained features, the tendencies of multi-collinearity are also growing.

The main purpose of our experiments is to evaluate the ability of LRR method in extracting the subspace structure of covariance descriptors. Because the subspace structure of each image may be different, therefore, we partition every image into K regions with $K \in [2; 40]$. And, the reported evaluation results are based on the K that provides the best performance of the algorithms.

5.4.2 Results

The LRR-CCM can successfully run over all images in the BSDS300 but not all images in the BSDS500 since ALM (i.e. Algorithm 5.1) failed to converge when running with a few images in BSDS500. One possible reason is there are too many noises in the covariance descriptor sets of those images, which makes it hard to find the stable subspace structure. And, an incorrect subspace representation may result in a problematic superpixel-wise similarity matrix which causes a failure in the spectral clustering procedure in *T-cut*.

Table 5.1 shows the results of LRR-CCM with different types of covariance descriptors over BSDS300. With CovII descriptor, the PRI and VoI of LRR-CCM reach the best, and, GCE and BDE reach the best with CovI and CovIII respectively. But the overall performance of the three descriptors are very close.

Table 5.2 shows the comparison between the state-of-the-art algorithms with LRR-CCM. The CCM has the highest overall performance but LRR-CCM performs best on PRI.

5.5 Conclusion

We have presented a novel approach, LRR-CCM, for improving the CCM algorithm. In LRR-CCM, we apply the augmented Lagrange multiplier (ALM) method to find the

Table 5.2: Performance of different algorithms over BSDS300

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.8319	1.6849	**0.1779**	11.2900	2.5
ℓ_0-sparse	0.8355	1.9935	0.2297	**11.1955**	3.5
CCM	0.8495	**1.6260**	0.1785	12.3034	2.25
MBG-CCM	0.8421	2.0223	0.2231	11.5771	3.75
LRR-CCM	**0.8499**	1.7418	0.1915	12.7635	3

low-rank representation of the covariance descriptor set and build the superpixel-wise similarity matrix based on the low-rank representation.

We test the LRR-CCM with three different covariance descriptors. Each of them contains noises due to collinearity. The experiment results show that the performance of LRR-CCM with these covariance descriptors are relative stable, which means the LRR method we proposed is able to extract the robust subspace representation for the covariance descriptors. And for covariance descriptors, the low-rank representation may be the "good" descriptor.

But the shortcomings of the MBG-CCM also happen on LRR-CCM. It needs spectral clustering for generating the final segmentation and a specified number of clusters. Moreover, in order to find the most suitable parameters, LRR needs to search over the parameter space, which is a significant overhead for a segmentation algorithm. So, a new algorithm which can avoid these problems is needed. This is the purpose of our next chapter.

Chapter 6

Superpixel Association

6.1 Introduction

As a group of pixels with perceptual similarities, superpixel is widely used in computer vision applications. In many works, the superpixels are used as primitives for image processing tasks. For example, Gould et al. developed a few of object recognition algorithms based on superpixels (Gould *et al.*, 2008, 2014, 2009); Kluckner *et al.* (2009) proposed an image segmentation algorithm that working with superpixels via the random forests. While in some other research, different superpixel segmentations are regarded as segmentation clues, and the ensemble clustering algorithms are applied on the them and produce the final segmentation. Kim *et al.* (2010) proposed an algorithm in which the superpixel segmentations are incorporated in a dense affinity matrix over pixels, and the final segmentation is obtained by spectral clustering on the pixels. A bipartite graph is used to solve the information fusing problem by which both the pixels and superpixels are set as graph vertices (Li *et al.*, 2012; Wang *et al.*, 2013), and the final clustering information of the pixels is delivered from a spectral clustering based algorithm, named *T-cut* (Li *et al.*, 2012).

Although most of the superpixel-based algorithms are well-tuned to provide good performance, there are still a few issues to be addressed. The first one is the selection of superpixel segmentations. Since different superpixel algorithms (or, different parameter settings) generate different superpixel segmentations, the performance of those algorithms varies, especially for those algorithms that work on a single superpixel segmentation directly. Secondly, many superpixel-based algorithms employ graph models and the final segmentation is often obtained via spectral clustering. So, they need a specified cluster number for the final segmentation. However, even for a human,

it is still not easy to given an exact cluster number for image segmentation, because for an image, the "correct" segmentation is not unique, especially for natural images. Fig. 6.1 demonstrates this phenomenon, the first left column is the original image, and the rest are the different segmentations made by human. It is easy to notices that some people partition the sky into a few regions while others don't. Actually, all these segmentations are considered to be "correct". This phenomenon encourages researchers to model the segmentation with a hierarchical structure. For example, Arbelaez *et al.* (2011) proposed the ultrametric contour map (UGM) algorithm in which a hierarchical segmentation tree is built to capture the possible segmentations with different scales.

Figure 6.1: An example of the multiple "correct" segmentations.

In this chapter, we also propose a hierarchical tree model for image segmentation, which is named superpixel-based hierarchical segmentation tree (SHST). With this algorithm, we can generate segmentations with different scales. Different to the UGM in which the tree is constructed from the contours, our algorithm builds the tree with the superpixel associations. To our knowledge, this method has not been sufficiently explored. Our contributions mainly contain the follows,

- we propose the concept of superpixel association and show some nice properties of it;
- we propose the SHST algorithm by which a hierarchical segmentation tree can be built with superpixel associations;
- we propose a strategy for determining the number of segments with the SHST.

The rest of the chapter is organized as follows. Section 6.2 is about the superpixel association and its properties. In Section 6.3, we propose the SHST algorithm and give the details for building the SHST. The experiment results is reported in Section 6.4. And Section 6.5 summarizes the chapter.

6.2 Superpixel association

The concept of superpixel association is proposed with the inspiration from the Hybrid Bipartite Graph Formation (HBGF) algorithm. In HBGF based image segmentation algorithms, like SAS (Li *et al.*, 2012) and CCM, we find that those pixels having the same pixel-superpixel relations are always partitioned into the same segment in the final segmentation. This indicates those pixels can be considered as a unit in the HBGF based algorithm. Actually, we will prove that the segmentation result from an HBGF based algorithm will not change if we take the superpixel associations instead of pixels as primitives for segmentation. Firstly, we give the definitions of superpixel association.

Let $I = \{p_u\}_{u=1}^{m}$ represent an image of m pixels. A superpixel segmentation is a clustering on I denoted by $S = \{s_1, \cdots, s_K\}$, where s_i is a subset of I, called a superpixel, and K is the number of superpixels in S; for $\forall s_i, s_j \in S$, where $i \neq j$, we have $s_i \cap s_j = \emptyset$. We denote $\mathcal{S} = \{S^1, \cdots, S^N\}$ as a collection of superpixel segmentations, where N is the number of superpixel segmentations. And, let $SL^i = \{1, 2, \cdots, K_i\}$ represent the set of superpixel labels of S^i, where K_i is the number of superpixels in S^i.

For a given $S^i \in \mathcal{S}$, we define an indicator $Id : I \to SL^i$, which assigns a superpixel label $l \in SL^i$ to pixel p_u. Then, we have the following definition,

Definition 1 (Superpixel association)**.** *A set of pixels $Sa = \{p_u\}_{u=1}^{n_{Sa}}$ is called a superpixel association if it satisfies the following conditions:*

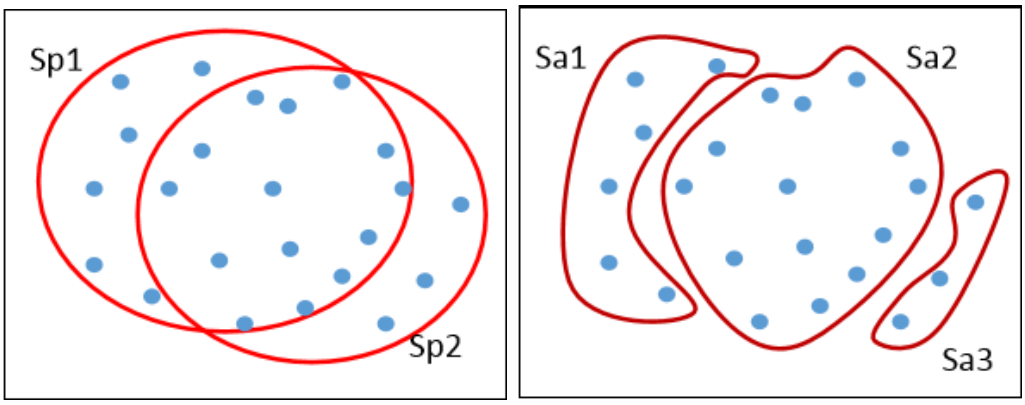
(i) $\forall p_u, p_v \in Sa$, $\forall S^i \in \mathcal{S}$, it holds that $Id(p_u) = Id(p_v)$;

(ii) $\forall p_u \in Sa$ and $p_v \notin Sa$, $\exists S^i \in \mathcal{S}$, such that $Id(p_u) \neq Id(p_v)$;

where n_{Sa} is the number of pixels in the superpixel association.

For a given $\mathcal{S}$, there exists a unique collection of Sa, which is denoted as $\mathcal{A} = \{Sa_1, ..., Sa_M\}$, where M is the number of superpixel associations. Two pixels are in the same Sa if and only if they are in the same superpixel in all of the given superpixel segmentations; Fig. 6.2 gives an example. Besides, for a given threshold $\tau \geq 0$, we say Sa is a tiny superpixel association, if $n_{sa} < \tau$ holds.

Theoretically, we can take superpixel associations as primitives for segmentation instead of pixels because of the following Theorem 6.2.1.



(a) (b)

Figure 6.2: An example for superpixel association: (a) two superpixels; (b) the corresponding superpixel associations. Intuitively, the superpixel associations are the intersected and non-intersected parts of the given superpixels.

Theorem 6.2.1. *For a given I and a collection of its clusterings $\mathcal{S}$, let $G^p(V^X, V^Y, E)$ and $G^{sa}(V^X, V^Y, E)$ denote two bipartite graphs constructed according to the HBGF algorithm, and, S'_p and S'_{sa} denote the final clusterings obtained from the respective graphs. For the spectral clustering on G^p and G^{sa} with a given cluster number k, it holds that $S'_p = S'_{sa}$, if G^p and G^{sa} satisfy the following conditions:*

(i) each vertex in V^X represents a $p_i \in I$ for G^p and a $Sa_i \in \mathcal{A}$ for G^{sa};

(ii) the weights on $E = \{e_r\}_{r=1}^R$ are set to be $\mathbf{C}$ for G^p and $\{n_{sa}^r \mathbf{C}\}_{r=1}^R$ for G^{sa};

where $\mathbf{C}$ is a positive constant; K is the number of edges; and, n_{sa}^r is the number of pixels in the Sa at one of the ends of e_r.

Proof. Without loss of generality, for image $P = \{p_1, \cdots, p_m\}$, we make the following settings to simplify the proof,

1. the superpixels $\mathcal{S} = \{Sp_1, \cdots, Sp_n\}$;
2. the superpixel associations $\mathcal{A} = \{Sa_1, Sa_2\}$;
3. $Sa_1 = \{p_1, \cdots, p_k\} = \cap_{i=1}^r Sp_i$;
4. $Sa_2 = \{p_{k+1}, \cdots, p_m\} = \cap_{i=r+1}^n Sp_i$;
5. G^p is built with Algorithm 2.3 by $V^I = P$;

6. G^{Sa} is built with Algorithm 2.3 by $V^I = \mathcal{A}$, and set w_{ij} be the number of pixels in Sa_i if $Sa_i \subseteq Sp_j$, and 0 otherwise.

Let B^p and B^{Sa} be the respective cross-adjacency matrices, then, we have

$$B^p = \begin{array}{c} \\ p_1 \\ \vdots \\ p_k \\ p_{k+1} \\ \vdots \\ p_m \end{array} \begin{array}{c} \begin{array}{cccccc} Sp_1 & \cdots & Sp_r & Sp_{r+1} & \cdots & Sp_n \end{array} \\ \begin{bmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 1 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 1 & \cdots & 1 \end{bmatrix}_{m \times n} \end{array},$$

and,

$$B^{Sa} = \begin{array}{c} \\ Sa_1 \\ Sa_2 \end{array} \begin{array}{c} \begin{array}{cccccc} Sp_1 & \cdots & Sp_r & Sp_{r+1} & \cdots & Sp_n \end{array} \\ \begin{bmatrix} k & \cdots & k & 0 & \cdots & 0 \\ 0 & \cdots & 0 & m-k & \cdots & m-k \end{bmatrix}_{2 \times n} \end{array}.$$

We denote $D_X = diag(B\mathbf{1})$ and $D_Y = diag(B^T\mathbf{1})$ are two degree matrices corresponding to V^I and V^C, where $\mathbf{1}$ is a vector of ones in proper size and diag$(\cdot)$ is a diagonal matrix whose nonzero entries represented by"$(\cdot)$".

Obviously, we can verify $D_Y^p = D_Y^{Sa}$ and $(B^p)^T(D_X^p)^{-1}B^p = (B^{Sa})^T(D_X^{Sa})^{-1}B^{Sa}$. So, for

$$L_Y^{p(sym)} = (D_Y^p)^{-\frac{1}{2}}(B^p)^T(D_X^p)^{-1}B^p(D_Y^p)^{-\frac{1}{2}}$$

and

$$L_Y^{Sa(sym)} = (D_Y^{Sa})^{-\frac{1}{2}}(B^{Sa})^T(D_X^{Sa})^{-1}B^{Sa}(D_Y^{Sa})^{-\frac{1}{2}},$$

we have

$$L_Y^p = L_Y^{Sa} = \begin{array}{c} \\ 1 \\ \vdots \\ r \\ r{+}1 \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & \cdots & r & r{+}1 & \cdots & n \end{array} \\ \begin{bmatrix} \frac{k}{r} & \cdots & \frac{k}{r} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ \frac{k}{r} & \cdots & \frac{k}{r} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \frac{m-k}{n-r} & \cdots & \frac{m-k}{n-r} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \frac{m-k}{n-r} & \cdots & \frac{m-k}{n-r} \end{bmatrix}_{n \times n} \end{array}.$$

Let u_i and v_i be the i-th eigenvector of L_X and L_Y and λ_i be the respective eigenvalue of L_Y. According to Theorem 1 in (Li *et al.*, 2012), $u_i = \frac{1}{1-\gamma_i}D_X^{-1}Bv_i$, where $\gamma_i(2-\gamma_i) = \lambda_i$. Because the eigenvectors of L_Y^p and L_Y^{Sa} are the same, we can verify

the label indicator in u_i^{Sa} for Sa_1 is the same as those in u_i^X for $\forall p_i \in Sa_1$, and this also holds for Sa_2 and $\forall p_i \in Sa_2$. Thus G^p and G^{Sa} are equivalent in representing the data structure of the pixel set P. □

Theorem 6.2.1 indicates that, in the HBGF, if we take superpixel associations instead of pixels as vertices in G and replace the weight with the number of pixels in the superpixel association, we can get the same clustering. But for superpixels, there is not such a guarantee. This means as primitives, superpixel associations are better than superpixels.

6.3 Segmentation with superpixel associations

Since the superpixel associations is a kind of primitives for ensemble clustering, for hierarchical image segmentation, one intuitive idea is to generate the segmentation by merging the superpixel association gradually, i.e. by a bottom-up process. Moreover, this can be easy done with a tree growing algorithm. Thus, we propose a novel image segmentation algorithm which works on the superpixel association level, named superpixel-based hierarchical segmentation tree (SHST). In our algorithm, the superpixel associations are regarded as the leaves, and, the tree is constructed based on a similarity matrix of the superpixel associations.

The framework of SHST can simply be divided into two parts: measuring the similarities among the superpixel associations and merging them according to the similarities. However, this algorithm has three characteristics which are different to the state of the art. The first one is the method for similarity measure. The second one is a two-stage merging strategy which allows balancing between the local and globe similarity. Finally, SHST is able to determine the number of segmentations automatically by setting a cluster lifetime. The details are elaborated as follows.

6.3.1 The similarity measure

We propose a voting strategy for measuring the similarity between two superpixel associations. Given a set of superpixel segmentations $\mathcal{S}$; $Sa_i = \{p_1^i, \cdots, p_{K_i}^i\}$ and $Sa_j = \{p_1^j, \cdots, p_{K_j}^j\}$ are two superpixel associations obtained from $\mathcal{S}$ which contain K_i and K_j pixels respectively. Moreover, let $\Pi = \{\pi^1, \cdots, \pi^M\}$ be a set of segmentations of the image I and $\pi^m = \{R_1^m, \cdots, R_{r_m}^m\}$ be a segmentation containing r_m regions. We

define a δ function as

$$\delta^m(p_i, p_j) = \begin{cases} 1, & \text{if } p_i \in R_r^m, p_j \in R_r^m, \text{ and, } R_r^m \in \pi^m \\ 0, & \text{otherwise,} \end{cases} \tag{6.1}$$

where p_i and p_j are pixels in I. Then, the co-occurrence p_i and p_j on a given Π is

$$\Delta(p_i, p_j) = \sum_{m=1}^{M} \sum_{r=1}^{r_m} \delta^m(p_i, p_j). \tag{6.2}$$

And, the similarity between Sa_i and Sa_j is defined as,

$$sim(Sa_i, Sa_j) = \frac{\sum_{p^i \in Sa_i} \sum_{p^j \in Sa_j} \Delta(p^i, p^j)}{|Sa_i| \cdot |Sa_j|}, \tag{6.3}$$

where $|Sa_i|$ represents the number of pixels in Sa_i. Actually, if we set H be a histogram of N bins where N is the total number of regions in Π, and each bin is represented by the number of pixels belongs to the respective region, then, Eq. 6.3 can be rewritten as

$$sim(Sa_i, Sa_j) = \frac{H_{Sa_i} H_{Sa_j}^T}{|Sa_i| \cdot |Sa_j|}, \tag{6.4}$$

where H_{Sa_i} is a row vector representing the histogram of occurrence of pixels in Sa_i on Π.

There are a few nice properties for our similarity function. First, the similarity is computed purely from the co-occurrence of the pixels, which is simple and easy to adopt. Given $\mathcal{S}$, we can get the similarity without any other extra feature extraction procedures. Moreover, according to Eq. 6.4, the similarity is actually a normalized inner product, which is quite easy to compute. Second, we can balance the similarity in different scales by using different base segmentations Π, which may benefit image segmentation.

6.3.2 Two-Stage merging

From Eq. 6.3 it is clear that different segmentation sets Π will bring different similarities. Thus, we developed a two-stage merging strategy which merges the superpixel associations in two steps with two different Π. In the first stage, the superpixel associations are refined with the given superpixel segmentations $\mathcal{S}$, and in the second stage, the refined superpixel associations are merged to final segmentation based on a few clusterings of $\mathcal{S}$ which contain the long range relations of the superpixel associations.

Refining superpixel association with $\mathcal{S}$

The whole merging procedure starts with the refining step based on $\mathcal{S}$. There are two reasons for doing this. First, there always exists some tiny superpixel associations, i.e. superpixel association with very small size, because noise or complexity in real-world images makes the superpixel boundaries unstable. Second, many superpixel algorithms tend to over segment the image, which makes H in Eq. 6.4 (the occurrence histogram on $\mathcal{S}$) sparse. This means $\mathcal{S}$ contains strong local similarity information while weak long range relations. Therefore, a refining procedure based on $\mathcal{S}$ is necessary and good for producing the stable performance.

We refine the superpixel associations by merging the tiny entries into their nearest neighbors according to the similarity measure in Eq. 6.3. For a given threshold τ, the tiny superpixel association is merged to the one that is most similar to it. Since it is possible for two tiny superpixel associations to be merged into one but the new one is still tiny, the merging process is done in an iterative manner. In this chapter, we set τ as a p-quantile of the superpixel associations' size sequence, where $p \in [0, 1]$. This makes the selection of τ more flexible.

Long range similarity

Since many superpixel algorithms tend to over segment the image, the base segmentation set $\mathcal{S}$ tends to lose the long range similarities among the superpixel associations, which means the H based on $\mathcal{S}$ is sparse and Eq. 6.4 cannot work properly. The solution for this problem is intuitive, i.e. making a set of clusterings on $\mathcal{S}$ and computing the similarities based on them. Fortunately, we can obtain the clusterings on $\mathcal{S}$ easily with an algorithm similar to HBGF, which models the pixel-superpixel relations with a bipartite graph and get the clusterings via spectral clustering. Algorithm 6.1 shows the details. Besides, it is worthy to mention that, in Algorithm 6.1, we can fuse different features of the superpixels by the function $similar(\cdot, \cdot)$, and then, the segmentation information from different features will transfer into the final segmentation via Eq. 6.3.

Construction of SHST

The tree is built on the refined superpixel associations by a minimum spanning tree algorithm (MST). Let Z be the similarity matrix of the refined superpixel associations, the tree construction process is summarized in Algorithm 6.2.

Algorithm 6.1 The spectral clustering on $\mathcal{S}$

Input: Image $I = \{p_1, \cdots, p_m\}$, Superpixel clusterings $\mathcal{S} = \{S^1, \cdots, S^N\}$, a set of number of clusters $K = \{K_1, \cdots, K_M\}$;

Output: A set of clusterings on $\mathcal{S}$: $\mathcal{C} = \{C^1, \cdots, C^M\}$.

/*Part I: Graph construction.*/
Set $V^X = I \cup S$, $V^Y = S$, $W = \emptyset$; //Another option is $V^X = I$.
Compute $n = \sum_{i=1}^{N} K_i$; // K_i is the number of superpixels in S^i.
for $v_i^X \in V^X, i = 1, \cdots, m+n$ **do**
  for $v_j^Y \in V^Y, j = 1, \cdots, n$ **do**
    $w_{ij} = similar(v_i^X, v_j^Y)$ //$similar(\cdot,\cdot)$ is the similarity measure.
  end for
end for
/* Part II: Partition Superpixels*/
Compute the adjacency matrix on V^Y: $W_Y = W^T D_x^{-1} W$.
Apply spectral clustering on W^Y with $K_i \in K$, and obtain $\mathcal{C}$.

6.3.3 The cluster lifetime

We define the k-cluster lifetime for the SHST. Let $T = \{t_1, \cdots, t_M | t_1 \leq t_2 \leq \cdots \leq t_M\}$ denote the set of sorted edge weights of the SHST, and a cluster lifetime is defined as $\delta_i = (t_{i+1} - t_i)$. Since the SHST is constructed by a MST algorithm, we say δ_i is a k-cluster lifetime if the SHST is separated into k subtrees when removing the edges whose weights t satisfy $t > t_i$.

According to the highest lifetime criterion proposed in Fred and Jain (2002), a higher value of a k-cluster lifetime indicates the k is closer to the true cluster number of the dataset. Thus, given a sorted cluster lifetime set $\Delta = \{\delta_1, \cdots, \delta_{M-1} | \delta_1 \geq \delta_2 \geq \cdots \geq \delta_{M-1}\}$, we can obtain the image segmentation by specifying the lifetime level, which is more flexible and objective than those algorithms that only work with specified cluster numbers. Fig. 6.3 demonstrates the lifetime criterion.

Therefore, we have two ways for getting the segmentation result from SHST. One is cutting the tree with a specified cluster number, another is partitioning the tree with a given lifetime level.

Algorithm 6.2 Construction of SHST

Input: Superpixel associations $\mathcal{A} = \{Sa_1, \cdots, Sa_M\}$, Size threshold τ, base clusterings $\mathcal{S} = \{S^1, \cdots, S^N\}$, a set of number of clusters $K = \{K_1, \cdots, K_M\}$;

Output: A hierarchical segmentation tree H

/* The first stage of merging*/

$\mathcal{A}^* = \emptyset$, $Z = [Z_{ij}]$

for $Sa_i \in \mathcal{A}$ **do**

  $Sa = Sa_i$, remove Sa_i from $\mathcal{A}$;

  repeat

    Compute the $sim(Sa, Sa_j)$ by Eq. 6.4 based on $\mathcal{S}$ for all $i \neq j$;

    Find the most similar Sa_j of Sa;

    $Sa = Sa \cup Sa_j$;

    Remove Sa_j from A^*;

  until $|Sa| > \tau$

  $A^* = A^* \cup Sa$;

end for

/* Computer a set of clusterings of $\mathcal{S}$ */

Apply Algorithm 6.1 on $\mathcal{S}$ with K;

Obtain $\mathcal{C}$;

/* The second stage of merging*/

Compute the similarity matrix Z of $\mathcal{A}^*$ by Eq. 6.4 based on $\mathcal{C}$;

Apply MST on $(-Z)$ and get H //note:$(-Z)$ is the dissimilarity matrix.

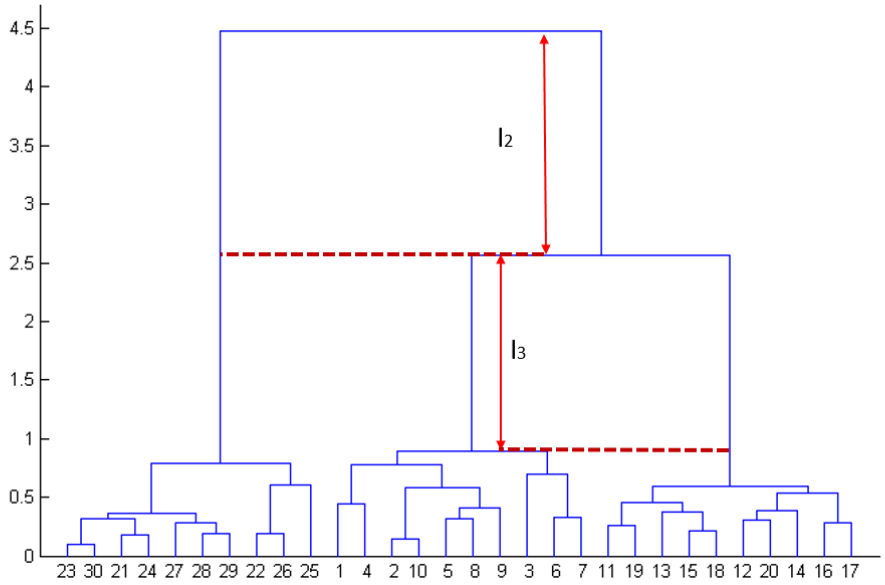


Figure 6.3: A dendrogram produced from SHST. If we remove the edges at l_2 level, the SHST will be partitioned into 2 subtrees. And, if we remove the edges at l_3, the SHST will be partition into 3 subtrees.

6.4 Experiments

6.4.1 Data sets and settings

The experiments are conducted on two public image segmentation datasets: the Berkeley Segmentation Dataset 300 (BSDS300), and its update, the Berkeley Segmentation Dataset 500 (BSDS500) (Martin *et al.*, 2001; Arbelaez *et al.*, 2011).

For the similarity measure between superpixels, color and texture are two significant clues. We take *Lab* color space to compare the colors, in which the Euclidean distance can represent the difference of colors in the human visual system, while for the texture feature, we use texton (Arbelaez *et al.*, 2011) and color covariance matrix (Gu *et al.*, 2014a). In addition, we also consider the geometry relations of the superpixels, i.e. the neighborhoods of the superpixels and take it as a cue in graph construction (Li *et al.*, 2012).

For the parameters in Algorithm 6.1, they are set to the same values as those in (Li *et al.*, 2012; Gu *et al.*, 2014a). For τ in Algorithm 6.2, we set it as the p-qualtile of the input superpixel associations' size sequence, and we adopt a grid search in $[0.1, 0.9]$ with the step of 0.1 for finding the best p. We set the cluster number K from 2 to 20 for base clustering generation, which gives 19 different clusterings for all the superpixels.

Furthermore, we use two classical superpixel algorithms to generate the superpixel segmentations: the F-H method (Felzenszwalb and Huttenlocher, 2004) and Mean Shift (Comaniciu and Meer, 2002); and the parameters for superpixel generation are set to be the same to those in Chapter 3.

6.4.2 Results

We test the SHST with PRI (Unnikrishnan *et al.*, 2007), VoI (Meilă, 2005), GCE (Martin *et al.*, 2001), and BDE (Freixenet *et al.*, 2002). And in some tables, we also listed the performance of the state of the art for comparison, which includes, SAS(Li *et al.*, 2012) and ℓ_0-sparse algorithm (Wang *et al.*, 2013), and CCM.

Firstly, we test the algorithm with different features, segmentation information, and different combinations of them, which includes, the *Lab* color (Col), texton (Tex), color covariance matrix (CovMat) and geometry relations of the superpixels(Geo). The performance of SHST is stable with those features over the two datasets; and the indices values indicate that the combination of Col, Tex, and Geo is better than others. The mean values of each index are demonstrated in Table 6.1 and Table 6.2.

Secondly, the comparisons against other state-of-the-art algorithms are conducted. SHST gets higher performance on PRI, VoI and GCE except BDE, which implies our approach is good at partitioning out the regions containing the objects but with rough boundaries. Table 6.3 and Table 6.4 show the average values of each index over the datasets.

Finally, we test the algorithm by varying the tiny superpixel association threshold p. Fig. 6.4 displays the result on BSDS500, which is conducted by combing the Col, Tex and Geo features. And, we find the the performance gets better with the increasing of p.

In addition, we note that, in all the tables, $(\cdot)^*$ and $(\cdot)^\dagger$ represent the results that were obtained by the lifetime criterion and specified segment numbers respectively; the parameters are well tuned for the highest performance on PRI. And, Fig. 6.5 is an example of hierarchical segmentation tree; Fig. 6.6 shows a few of the segmentations generated by different algorithms.

6.5 Conclusion

We propose a novel concept, named superpixel association, which is the overlap of superpixels from different superpixel segmentations. Then, a similarity measure function

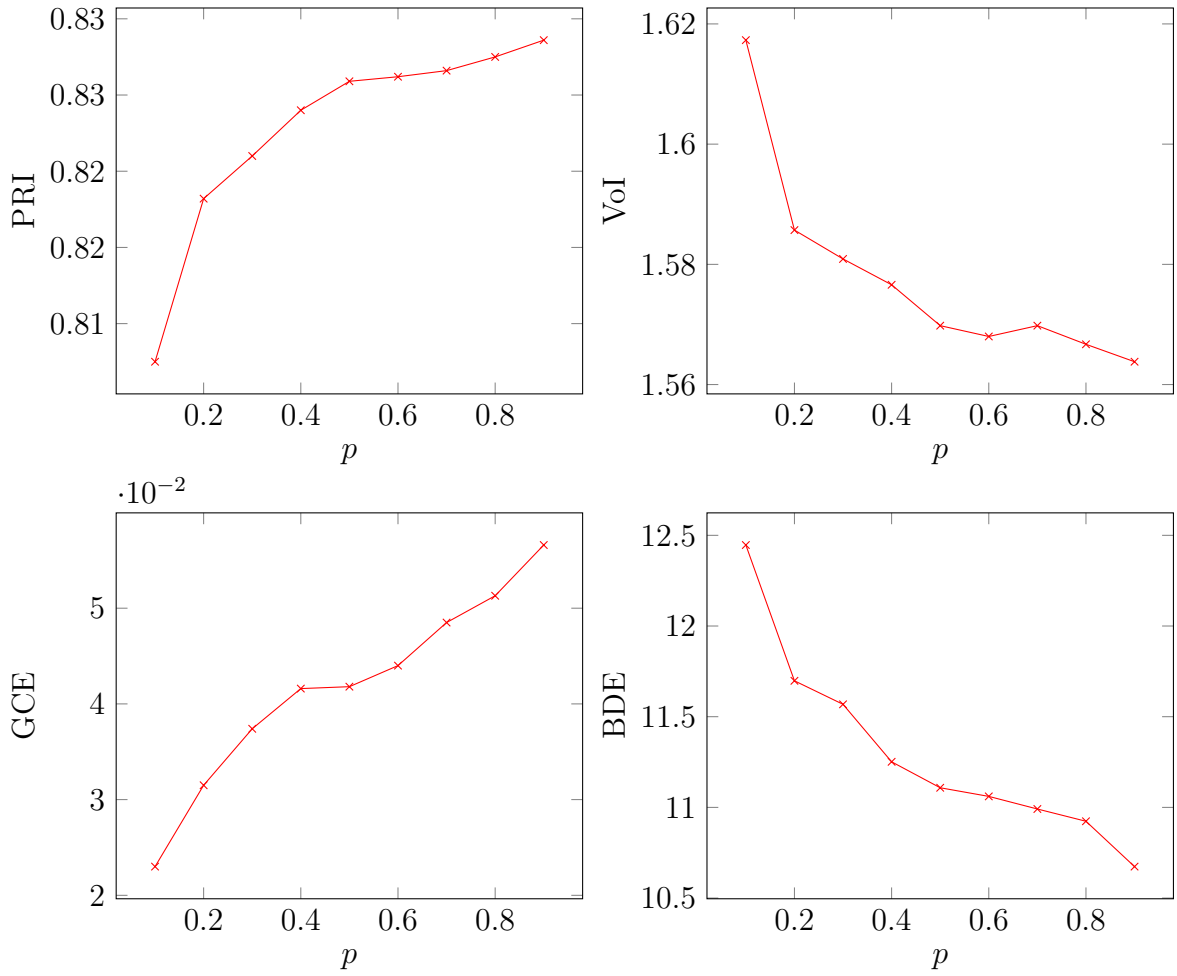


Figure 6.4: Testing on the size threshold parameter p of tiny superpixel association. The four charts correspond to the average scores of PRI, VoI, GCE and BDE on BSDS500; Lab color, texton and geometry relations are used in the test.

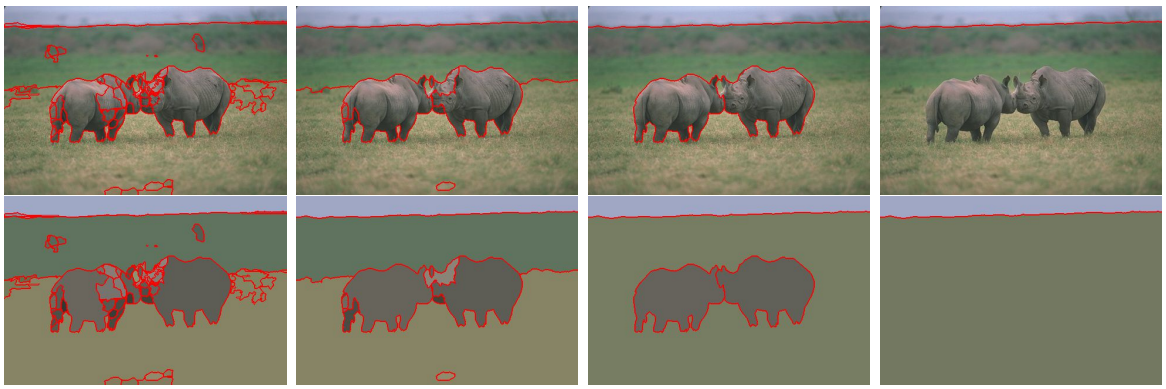


Figure 6.5: An example of the hierarchical structure; from left to right, the number of segments is decreasing.

Table 6.1: Performance of SHST with different features on BSDS300

Features	PRI	VoI	GCE	BDE
(Col+Geo)*	0.8422	1.4668	0.1502	18.2738
(Col+CovMat)*	0.8368	2.3072	0.1945	12.5409
(Col+Tex)*	0.8374	2.2240	0.1939	**11.7387**
(Col+Geo+Tex)*	**0.8416**	**1.4308**	**0.1449**	21.9028
(Col+Geo)†	0.8465	1.4491	0.1465	18.2901
(Col+CovMat)†	0.8335	1.9257	0.2120	12.5758
(Col+Tex)†	0.8339	1.8597	0.2081	**11.3832**
(Col+Geo+Tex)†	**0.8452**	**1.4209**	**0.1420**	23.1781

Table 6.2: Performance of SHST with different features on BSDS500

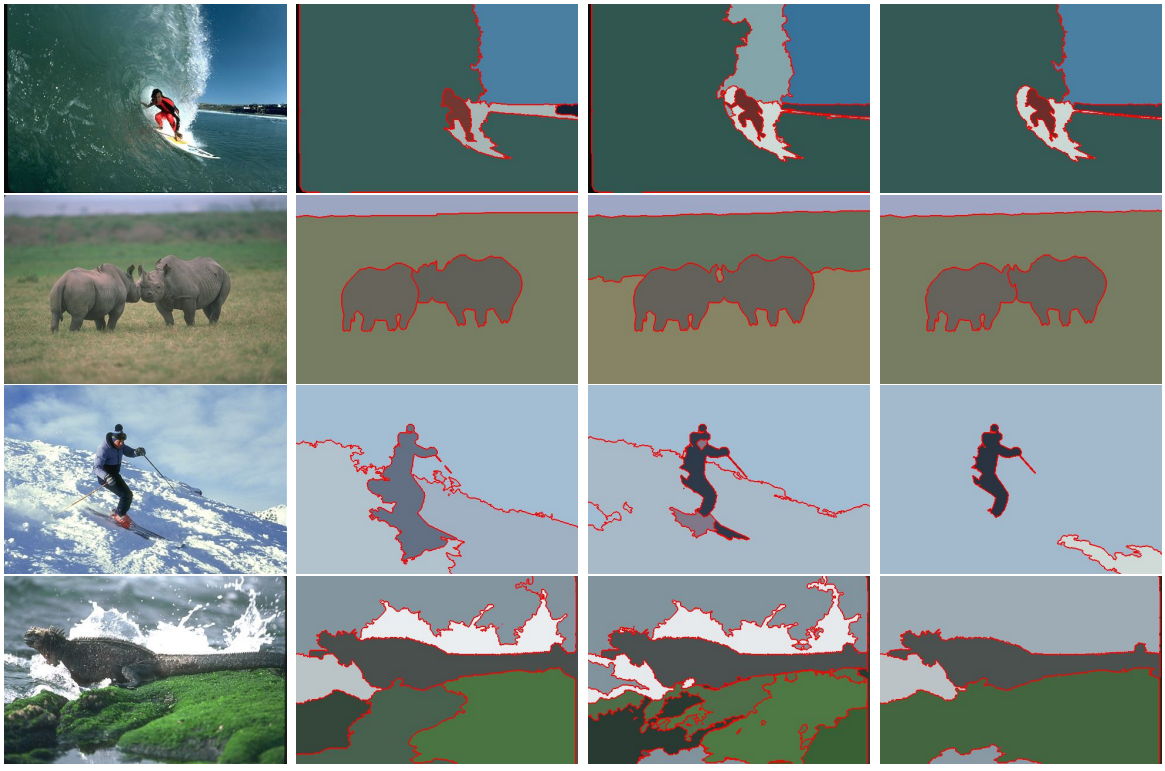
Features	PRI	VoI	GCE	BDE
(Col+Geo)*	0.8399	1.5458	0.1612	15.7963
(Col+CovMat)*	0.8369	2.3513	0.1948	11.8055
(Col+Tex)*	0.8367	2.2909	0.1964	**11.1386**
(Col+Geo+Tex)*	**0.8417**	**1.4846**	**0.1497**	18.1900
(Col+Geo)†	0.8443	1.5264	0.1588	15.6898
(Col+CovMat)†	0.8337	1.9473	0.2158	11.5977
(Col+Tex)†	0.8331	1.9066	0.2152	**10.8030**
(Col+Geo+Tex)†	**0.8451**	**1.4749**	**0.1483**	19.0004

Table 6.3: Performance of Different Algorithms on BSD300

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.8319	1.6849	0.1779	11.2900	3.5
CCM	**0.8495**	1.6260	0.1785	12.3034	2.75
ℓ_0-sparse	0.8335	1.9935	0.2297	**11.1955**	4
SHST*	0.8422	1.4668	0.1502	18.2738	2.75
SHST†	0.8465	**1.4491**	**0.1465**	18.2901	2.25

Table 6.4: Performance of Different Algorithms on BSD500

Algorithms	PRI	VoI	GCE	BDE	Avg.R
SAS	0.8372	1.6914	0.1813	12.6599	3
CCM	0.8407	2.0399	0.2359	**10.7829**	3
ℓ_0-sparse	-	-	-	-	-
SHST*	0.8417	1.4846	0.1497	18.1900	2.25
SHST†	**0.8451**	**1.4749**	**0.1483**	19.0004	1.75



(a) (b) (c) (d)

Figure 6.6: Segmentations from different approach; (a) the original (b) SAS (c) CCM (d) SHST*. SHST* tends to partition the image with fewer segments.

is defined based on the majority voting algorithm. And with this similarity measure, the tiny superpixel associations can be merged into their neighbors so that a more robust over-segmentation of the image (i.e. a set of refined superpixel associations) can be obtained.

And, we also proposed a segmentation tree algorithm, i.e. SHST, which can build a hierarchical segmentation tree on the superpixel associations (or, refined superpixel associations). SHST has two advantages. First, it is more flexible in partitioning the image since it can generate the segmentation with a specified number of segments or a scale level. Second, SHST is built with the superpixel associations which makes the computational cost affordable.

Extensive experiments have been conducted and the results have shown that our method gets stable performance with different features; and compared with other state-of-the-art segmentation algorithms, the outputs of SHST are competitive. Moreover, the grid search of the size threshold τ for the tiny superpixel association shows this parameter has a strong connection with the performance of SHST. So, as a parameter for balancing the quantity and quality of superpixel associations, it is efficient and effective.

Chapter 7

Semantic Segmentation with Unsupervised Segmentation

7.1 Introduction

Semantic segmentation is one of the frontiers in computer vision, which attempts to partition the image into semantically meaningful parts and classify each part into one of the pre-determined classes. Compared with unsupervised segmentation, semantic segmentation not only partitions the image into several "coherent" parts but also tries to understand what these parts represent. To this extent, semantic segmentation and unsupervised segmentation are different in solving the segmentation problem. For semantic segmentation, it is more like a classification task, while for unsupervised image segmentation, it is a task of clustering.

A wide range of semantic segmentation algorithms has been published in the past few years. In many early works, great efforts have been put on building frameworks for the semantic segmentation. Shotton *et al.* (2008) proposed the semantic random forests, which labels every pixel by a set of decision trees simply with a few low-level features. Fulkerson *et al.* (2009) proposed a two-stage model to do the semantic segmentation on superpixel-level, in which the superpixels are labeled by a support vector machine and then the labels are refined by a conditional random field (CRF). Koltun (2011) built a fully connected CRF over the pixels, and, they proposed an efficient inference algorithm to label every pixel. Another focus in this region is the features extraction. Gould *et al.* (2008) proposed a superpixel-based algorithm, in which the relative location prior is incorporated into the CRF. And Liu et al. employed a convolutional neural network is employed to extract the 'deep features', and the performance of the state of art is

improved when replacing the tradition features with those 'deep' ones (Liu *et al.*, 2015).

Moreover, the unsupervised segmentation also draws the attentions of researchers in semantic segmentation. Because the image cues containing in the unsupervised segmentations, such as contours, object shape etc., are informative, and they may be helpful in deciding the pixel labels. Kohli *et al.* (2009) proposed a higher order conditional random fields, which expands the basic CRF framework to incorporate higher-order potentials defined on superpixels. This higher order CRF incorporates the superpixels from different superpixel segmentations and improves object segmentation with better boundaries. Kluckner *et al.* (2009) also take use of the region consistency extracted from superpixels to improve the labelling accuracy.

In this chapter, we propose an algorithm that integrates the superpixel associations into semantic segmentation. In our algorithm, we adopt a random forests algorithm which provides structured labels, and the final labelling is generated via a puzzle game framework. The rest of the chapter is organized as follows. Section 7.2 is a brief introduction of semantic segmentation. In Section 7.3, we propose a generalized puzzle game framework, and Section 7.4 specifies the algorithm we proposed for semantic segmentation. In Section 7.5, the results of the experiments are reported. And in Section 7.6, the conclusion is given.

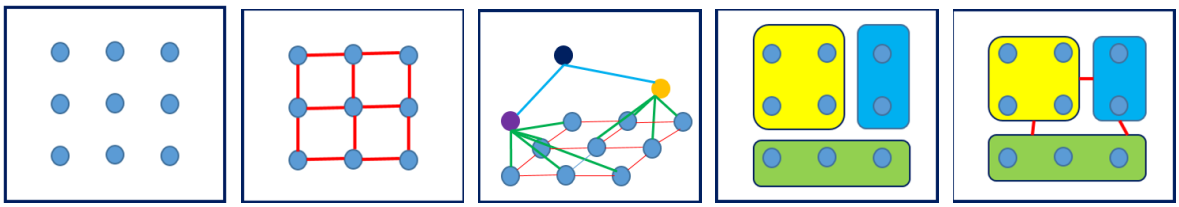
7.2 Semantic image segmentation

An image always contains one or more objects, including things like animals, people, sky, water, and mountains. The appearances of the objects generate intensity edges between one object and its neighbors in the image, and semantic image segmentation aims to split the image into regions corresponding to the objects and label them with the relevant object category simultaneously.

Usually, this task is approached with supervised machine learning techniques, which use a set of training images with manually segmented and labeled ground-truth to learn the parameters for discriminating different regions.

7.2.1 Overview

Essentially, the semantic segmentation task is a pixel-classification problem. A variety of methods have been proposed for solving this problem in recent years. However, based on the relationships of encoding between different pixels, these methods can be categorized into two classes. In the first class, the pixel-labeling problem is solved by



(a) (b) (c) (d) (e)

Figure 7.1: Methods of modeling pixel relationships: (a) pixels are independent, (b) pairwise relationships between pixels, (c) higher order relationships between pixels, (d) using superpixels, (e) pairewise relationships between superpixels. Random field models are built on different pixel relationships.

classifying each pixel independently, i.e., with a pixel-level model, such as Shotton *et al.* (2006, 2008). While in the second class, the algorithms work with pixel groups (i.e. superpixels) and assign a label to each group; we call them region-level models (Gould *et al.*, 2008). Compared with those pixel-level algorithms, the region-based methods are more computational efficient but may lead to an incorrect final labelling.

Many semantic segmentation algorithms for both two categories have a common framework, which is considered as a two-stage process (Fulkerson *et al.*, 2009; Gould *et al.*, 2008). The first stage is for extracting the unary potentials, and the second stage is determined the labels based on the relationships between the neighboring pixels. The second stage is modeled by a pairwise Markov Random Field (MRF) or Conditional Random Field (CRF) (Lafferty *et al.*, 2001; Kumar *et al.*, 2003). These models encourage the adjacent pixels to take the same semantic label, which leads the smooth boundaries in the segmentation results. In (Kohli *et al.*, 2009; Ladicky *et al.*, 2010), the pairwise interaction between pixels are replaced by higher-order relationships between pixels (or pixel groups), which improves the segmentation results with better object boundaries. Figure 7.1 shows different ways of using the pixel (or pixel group) relationships. The random field models are constructed on those relationships.

Moreover, from Shotton *et al.* (2006), we know that the unary potential has a significant influence on the success of a segmentation algorithm. Many efforts have been put on the generation of good unary potentials. In some early works, many classical classifiers are used as unary classifiers, such as support vector machine, logistic regression, and random forests etc. (Shotton *et al.*, 2006; Verbeek and Triggs, 2007; Gould *et al.*, 2008; Fulkerson *et al.*, 2009). Recently, structured prediction algorithms

are also introduced to training the unary classifiers, for example, the structured support vector machine in (Liu *et al.*, 2015) and the structured random forests in (Kontschieder *et al.*, 2011, 2014).

7.2.2 Features

Feature extraction is one critical issue in image segmentation. There are a lot of informative image cues that can be used for semantic segmentation.

Obviously, the most widely used features are the low-level features, such as intensity, color, texture, etc. These features are easy to get and always computed on a per-pixel basis and incorporate local color or texture statistics. Another popular descriptors are the mid-level features, which are extracted from regions (i.e. superpixels) to provide shape, continuity or symmetry information. For example, in Kluckner *et al.* (2009), the covariance descriptors extracted from superpixels are used to train the random forests. In addition, many hand crafted features are also involved in generating mid-level descriptors. For example, the density of SIFT and HoG feature is used to represent the superpixels (Fulkerson *et al.*, 2009; Ladicky *et al.*, 2010), and Bo *et al.* (2011) encode the pixels by a multi-layer sparse coding algorithm. A stronger mid-level feature is the deep feature introduced by Liu *et al.* (2015). This feature is extracted by a well-trained convolutional neural network and improves the performance of the CRF based semantic segmentation model.

In addition to the above mentioned features, the context information is also prevailing in semantic segmentation (Johnson *et al.*, 2013; Torralba *et al.*, 2003; Rabinovich *et al.*, 2007). The motivation behind using this feature is the perceptual psychology, which claims that the global image statistics and information about the contextual relations can help to seek the proper configurations of the objects in images.

7.2.3 Database

To a large extent, the quality of the training data set is also related to the performance of the semantic segmentation algorithms. Most techniques require a variety of training images with full pixel-wise labels. Unfortunately, such a kind of databases is expensive to obtain. But Shotton *et al.* (2008) shown that the performance of the algorithm can be improved by a random transform of the training images, which includes rotating with some random angles, rescaling with random ratios, and adding Gaussian noises. Actually, with such transfer operations, the number of training images is increased.

So, the random transform is a simple but effective method which may bring a boost in performance of the semantic segmentation algorithm.

7.3 Generalized puzzle game for semantic segmentation

Kontschieder *et al.* (2011) proposed a framework for semantic segmentation with squared label patches, and they named it as label puzzle game. We generalize this framework and make it integrated with unsupervised segmentation. To avoid unnecessary confusions, we inherit part of the notations from Kontschieder *et al.* (2011) in this section.

7.3.1 The generalized puzzle game

An image is a function $f : D \to R^d$ mapping the pixels in a 2-dimensional lattice $D \subseteq Z^2$ to d-dimensional feature vectors. Let $Y = \{1, \cdots, k\}$ be the class label set, and a labelling for an image is a function $\ell : D \to Y$ mapping the pixels to labels. A (label) puzzle piece is a label configuration, which is defined as a function $p : P \to Y \cup \{\perp\}$ mapping 2-dimensional points to labels or void (i.e. $\{\perp\}$, the absence of label), where $P \subseteq Z^2$ is a neighborhood of the pixel. Since one pixel can be associated to multiple puzzle pieces, we set $\mathcal{P}$ to represent the set of puzzle piece associating to one pixel.

We set $\mathscr{F}$, $\mathscr{L}$ and $\mathscr{P}$ to denote the set of images, labellings and puzzle pieces respectively. Then, a puzzle configuration is defined as a function $z : D \to \mathscr{P}$ assigning each pixel in D with a puzzle piece in $\mathscr{P}$. And, the set of puzzle configuration is denoted by $\mathscr{Z}$.

Moreover, let (i, j) and (u, v) represent the coordinates on D, and the puzzle piece at (i, j) is written as $p_{i,j}$ and by setting (i, j) be the center of $p_{i,j}$, we denote $p_{i,j}(u-i, v-j)$ the label in position (u, v), and this denotation also holds for the puzzle configuration z. Let $S \subseteq D$ be a region on image D, and the set of S is written as $\mathscr{S}$. We denote $\mathcal{S}_{i,j} = \{S \in \mathscr{S} | S \sim \mathcal{P}_{i,j} \neq \emptyset\}$ the regions associated with (i, j), where the symbol '$\sim$' indicates some relation. And, given a labelling ℓ, $\ell(S)$ is the label configuration of S.

The generalized puzzle game of semantic segmentation has three components: the puzzle generator, the agreement, and the game solver. They are given by the following four definitions.

Definition 2 (puzzle generator)**.** *For semantic segmentation, a puzzle generator is a function π that mapping each pixel $(i, j) \in D$ to a non-empty set of puzzle pieces*

$\mathcal{P}_{i,j} \subseteq \mathscr{P}$.

Let π be a puzzle game, then from Definition 2, all possible puzzle configurations obtained from π can be written as

$$\mathscr{Z}|_\pi = \{z \in \mathscr{Z} | z_{i,j} \in \mathcal{P}_{i,j}\}.$$

Definition 3 (generalized agreement)**.** *Given an image labelling ℓ and a set of region $\mathcal{S}$ respecting to the puzzle piece p, the agreement of p is defined as*

$$\phi(p, \ell, \mathcal{S}) = \sum_{S \in \mathcal{S}} similarity(p, \ell(S)), \tag{7.1}$$

where similarity$(\cdot, \cdot)$ is a function measure the similarity between p and $\ell(S)$.

Moreover, for a puzzle piece configuration, we define a total agreement as the sum of its puzzle piece agreements, i.e.

Definition 4 (total agreement)**.** *Given a puzzle piece configuration z, a labelling ℓ and the region set $\mathscr{S} = \{\mathcal{S}_{i,j}\}$, the total agreement of z is*

$$\Phi(z, \ell, \mathscr{S}) = \sum_{z_{i,j} \in z} \phi(z_{i,j}, \ell, \mathcal{S}_{i,j}). \tag{7.2}$$

Here, we note that Definition 3 and Definition 4 are the generalized versions of the respective definitions in (Kontschieder *et al.*, 2011), which extend the similarity measured with a region set $\mathscr{S}$.

Then, the solution for the puzzle game can be defined as an optimization problem of Eq. 7.2.

Definition 5 (game solver)**.** *Given a labelling set $\mathscr{L} = \{\ell^{(1)}, \cdots, \ell^{(T)}\}$ respected to $\mathscr{Z}|_\pi$ and a region set $\mathscr{S}$, the solution (z^*, ℓ^*) of the puzzle game satisfies*

$$(z^*, \ell^*) = arg \max_{(z,\ell)} \{\Phi(z, \ell, \mathscr{S}) | (z, l) \in \mathscr{Z}|_\pi \times \mathscr{L}\}. \tag{7.3}$$

In our puzzle game definitions, the unsupervised segmentation is able to effect the labelling updating via the agreement, which is different to those proposed by Kontschieder *et al.* (2011). Figure 7.2 shows the framework.

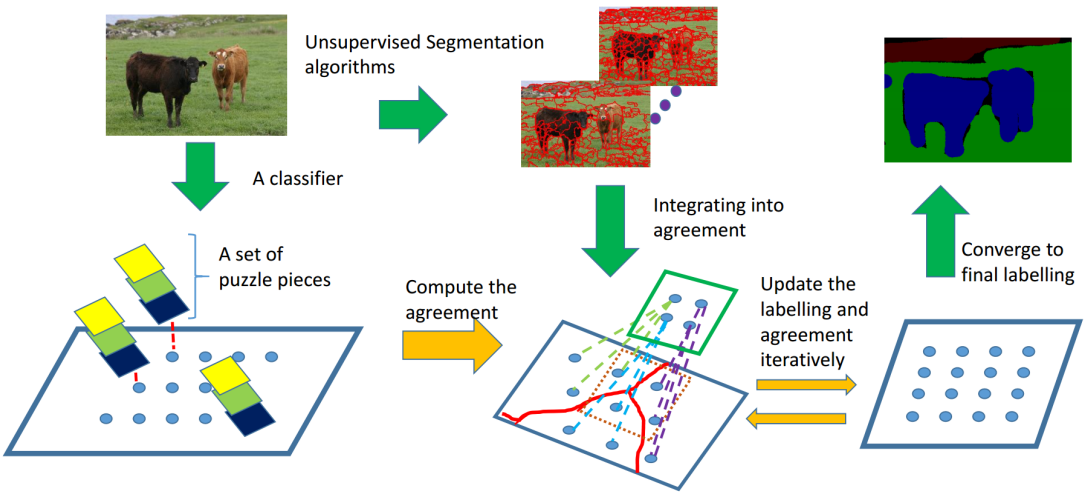


Figure 7.2: The framework of our algorithm. From left to right: first, a classifier generates a few puzzle pieces for each pixel; meanwhile, a few unsupervised segmentation algorithms produce some unsupervised segmentations; second, we compute the agreement by integrating the unsupervised segmentation and alternatively update the labelling and agreement; finally, the labelling converges to the final result.

7.3.2 Agreement with unsupervised segmentation

The agreement is critical for solving the puzzle game. With the generalized agreement (i.e. Definition 3), we can integrate the information provided by the unsupervised segmentation into the game solver.

The puzzle piece is set to be in a square patch shape with the size of $d \times d$, which is same as Kontschieder *et al.* (2011). But for the regions associated with pixel (i, j), we define them as $\mathcal{S}_{i,j} = \{S^D \in \mathscr{S}^D | S^D \cap \mathcal{P}_{i,j} \neq \emptyset\}$, where $\mathscr{S}^D$ is a given unsupervised segmentation $\mathscr{S}^D = \{S_1^D, \cdots, S_K^D\}$. Intuitively, those pixel labels within one region should be the same, so, given a labelling ℓ, we set the label of a segmentation S_k^D by a majority vote on the labels within it, written as $\text{Vote}[\ell(S_k^D)]$. Therefore, the similarity is defined as the coincidence of the labels of $p_{i,j} \cap S_{i,j}$ in $p_{i,j}$ and $\ell(S_{i,j})$, which is

$$\phi_{i,j}(p_{i,j}, \ell, \mathcal{S}_{i,j}) = \sum_{(u,v)\in D} \sum_{S_{i,j}\in\mathcal{S}_{i,j}} \delta[p_{i,j}(u-i, v-j) = \text{Vote}[\ell(S_{i,j})] | (u,v) \in S_{i,j}], \quad (7.4)$$

where $\delta[\cdot]$ is a delta function which yielding 1 if the input is true, 0 otherwise.

7.3.3 Optimization of game solver

We follow the algorithm in (Kontschieder *et al.*, 2011) to find the solution for the puzzle game. Specifically, the optimization of game solver (i.e. Eq. 7.3) is obtained by iteratively switching between optimizing the labelling $\ell \in \mathscr{L}$ and the puzzle configuration $z \in \mathscr{Z}|\pi$.

Let $\ell^{(t)}$ be the labelling of the image at time $t \geq 0$. The entries in puzzle configuration $z^{(t+1)}$ at time $t+1$ is individually updated via

$$z_{i,j}^{(t+1)} \in \arg\max_{p_{i,j}} \left\{ \phi_{i,j}(p_{i,j}, \ell^{(t)}, \mathcal{S}_{i,j}) | p_{i,j} \in \mathcal{P}_{i,j} \right\}. \tag{7.5}$$

And then, the $\ell^{(t+1)}$ is computed with the given $z^{(t+1)}$ by taking a majority vote over all puzzle pieces in z. Let C be the set of puzzle pieces z that covering pixel px_i and $z(i)$ be the label of px_i in z, we have

$$\ell^{(t+1)}(u,v) \in \arg\max_{y} \left\{ \sum_{(i,j)\in D} \delta[z_{i,j}^{(t+1)}(u-i, v-j) = y | y \in Y] \right\}. \tag{7.6}$$

Given an initial labelling $L^{(0)}$, then by updating the z and ℓ alternatively, it will reach the local maximum of the game solver, which is proofed as **Theorem** 1 in (Kontschieder *et al.*, 2011).

7.4 Integrating unsupervised segmentations

In our generalized puzzle game for semantic segmentation, we employ a structured prediction random forests algorithm as the game generator. Theoretically, a structured prediction algorithm is able to produce more candidates for the puzzle piece and enrich the searching space. The unsupervised segmentation is generated from superpixel associations via the SHST algorithm in Chapter 6.

7.4.1 Structured prediction with random forests

The random forests algorithm is an ensemble learning method for classification proposed in (Breiman, 2001), which has a few appealing properties, such as robustness to noise and resistance of over-fitting. Traditionally, the random forests algorithm assigns the input data samples with single, atomic class labels, but for many computer vision application, this kind of prediction models is limited because the inherent topological structure in the label space is ignored. Kontschieder *et al.* (2014) propose a random

forests algorithm for structured prediction, i.e. a random forests algorithm predicts structured objects rather than scalar discrete or real values. Here, we refer to the traditional random forests as *standard* random forests, while for the one providing structured predictions, we refer to it as *structured* random forests.

Basically, the two types of random forests have common structures, i.e. they are both ensemble of decision trees. Let $\mathcal{X}$ be a dataset, $\mathcal{Y} = \{y_i\}$ represent the class labels and π be a prediction of the class label. A decision tree t is a tree-structured classifier which makes a prediction by routing a sample $x \in \mathcal{X}$ through the nodes to a leaf, where the final prediction is proposed. A leaf $L_F(\pi) \in t$ is a node without any children nodes and is able to cast a class prediction π for any x that reaches it. For all other nodes, written as $N_D(\psi, t_l, t_r) \in t$, each of them is associated with a decision binary split function $\psi(x) : \mathcal{X} \to \{0, 1\}$, which determines the next route of sample x. If $\psi(x) = 0$, x will be forwarded to the left sub-tree $t_l \subset t$, or sent to the right sub-tree $t_r \subset t$, if $\psi(x) = 1$. And, a random forest is written as $F = \cup_{i=1}^{k} t_i$, i.e. an ensemble of a couple of decision trees.

The main difference between *standard* and *structured* random forests is in the training of split function. In the following, we first introduce the prediction function and then show the difference between the *standard* and *structured* random forests.

Formally, the prediction function $h(x|t) : \mathcal{X} \to \mathcal{Y}$ for the nodes in a decision tree T is written as

$$
\begin{aligned}
h(x|N_D(\psi, t_l, t_r)) &= \begin{cases} h(x|t_l), & \text{if } \psi(x) = 0, \\ h(x|t_r), & \text{if } \psi(x) = 1, \end{cases} \\
h(x|L_F(\pi)) &= \pi.
\end{aligned} \tag{7.7}
$$

A sample x is branched recursively, and the procedure stops until x reaches a leaf. Obviously, the split function ψ is critical for the prediction, and different ψ leads different outputs. In computer vision, there are four types of ψ are commonly used; let $f(x)$ be the feature vector of sample x and $f(x)_\theta$ be the value at the θ-th dimension.

$$
\begin{aligned}
\psi^{(1)}(x|\theta_1, \tau) &= [f(x)_{\theta_1} > \tau], \\
\psi^{(2)}(x|\theta_1, \theta_2, \tau) &= [f(x)_{\theta_1} - f(x)_{\theta_2} > \tau], \\
\psi^{(3)}(x|\theta_1, \theta_2, \tau) &= [f(x)_{\theta_1} + f(x)_{\theta_2} > \tau], \\
\psi^{(4)}(x|\theta_1, \theta_2, \tau) &= [|f(x)_{\theta_1} - f(x)_{\theta_2}| > \tau],
\end{aligned}
$$

where τ is a threshold, and $[\cdot]$ is an operator that gives 1 if the '$\cdot$' is true and 0 otherwise. The split function ψ and the parameters θ and τ are learned based on the

information gain theory, which is computed from the class label distribution. Let $\mathcal{P}$ be the samples reached node N_D in the training, $E(\mathcal{P})$ represent the entropy of the class label distribution of N_D, and Ψ be the set of ψ. When training the random forests, firstly, a ψ is selected from Ψ randomly, and then, the algorithm randomly generates a few sets of θ and τ, by which the $\mathcal{P}$ is split into $\{\mathcal{P}_l, \mathcal{P}_r\}$. The information gain is defined as (Shotton *et al.*, 2008)

$$\Delta E = -\frac{|\mathcal{P}_l|}{|\mathcal{P}|}E(\mathcal{P}_l) - \frac{|\mathcal{P}_r|}{|\mathcal{P}|}E(\mathcal{P}_r), \tag{7.8}$$

and, $\{\psi, \tau\}$ is selected to maximize the ΔE.

In *standard* random forests, every sample in the training data is assigned with exactly one label, so the entropy $E(\mathcal{P})$ is computed on the distribution of one variable. But for the *structured* random forests, the samples for training are associated with a batch of labels, which means the $E(\mathcal{P})$ is computed on a distribution of multiple variables, i.e. a joint distribution. This makes the computation cost of the *structured* random forests higher than the *standard* one. But, it has been pointed out that by randomly choosing a (marginal) distribution from the joint distribution, the output of the *structured* random forests remains as effective as the one using joint distribution (Kontschieder *et al.*, 2011).

Moreover, the leaf node of *structured* random forests is also different to the *standard* one. Because the samples reached the leaves are assigned to multiple labels, the label presentation of a leaf should also be a set of labels. In this chapter, we extract the label representations for the leaves in the same way as in (Kontschieder *et al.*, 2011). Specifically, let $\mathcal{P}_t = \{\mathbf{p}_1 \cdots, \mathbf{p}_k\}$ be the puzzle pieces that reach leaf L_{Ft} during the training process. The joint probability of the labels in $\mathbf{p} \in \mathcal{P}_t$ is defined as

$$Pr(\mathbf{p}|\mathcal{P}_t) = \prod_{i,j} Pr^{(i,j)}(\mathbf{p}(i,j)|\mathcal{P}_t), \tag{7.9}$$

where $Pr^{(i,j)}(\mathbf{p}(i,j)|\mathcal{P}_t)$ is the marginal probability over all $\mathbf{p} \in \mathcal{P}$ of the label at position (i,j). Then, the label representation π of leaf L_{Ft} will be the one puzzle piece that maximizing the joint probability, i.e.

$$\pi = \arg\max_{\mathbf{p}\in\mathcal{P}_t} Pr(\mathbf{p}|\mathcal{P}). \tag{7.10}$$

7.4.2 Integrating the unsupervised segmentation

We take the superpixel associations as the unsupervised segmentation for integration into the puzzle game.

The superpixel association is the intersections of superpixels from different over-segmentations and has been proved to be a good replacement of pixels for region-level models. The main motivation for using superpixel associations is because the probability of the pixels having the same labelling inside the superpixel association is higher than those regions obtained from single superpixel segmentation.

To obtain superpixel associations, we first use two classical superpixel algorithms to generate the superpixel segmentations, i.e. the F-H method (Felzenszwalb and Huttenlocher, 2004) and Mean Shift (Comaniciu and Meer, 2002), and the parameters in the superpixel generation procedure are set to be the same as those in Chapter 6. And then, following the Definition 1 in Chapter 6, we can get the superpixel associations.

Moreover, for those superpixel associations whose size is smaller than a given threshold, i.e. the tiny superpixel associations, we can adopt an optimization option to merge them into their nearest neighbors.

7.5 Experiment

For semantic segmentation, the feature representations have a significant influence on the success of a labelling algorithm. However in this chapter, since the primary intention of our experiments is to demonstrate the effects of integrating the unsupervised segmentation, we simply use the *Lab* color and a multi-layer sparse coding feature to be the pixel representation. The effect of the integration of unsupervised segmentation can still be observed even with this simple feature.

7.5.1 Data set and settings

The experiments of our algorithm are conducted on MSRC21. Following the protocol of previous works using MSRC21 (Shotton *et al.*, 2008; Gould *et al.*, 2008; Kohli *et al.*, 2009), we split the data into 276 training and 256 test samples and ignore those pixels with the void label during both training and evaluation. The results are evaluated with recall and precision, i.e. Avg(Class) and Global.

In our experiment, each pixel is represented by a vector which incorporating the color and texture features. For the color feature, we concatenate the Lab color values of pixels within the $d_{col} \times d_{col}$ neighbor (centered at the current pixel).

As for the texture feature, we employ the sparse coding algorithm with a spatial pyramid. Specifically, we first use the sparse coding techniques to extract a sparse representation for each pixel, and then, adopt a few spatial max pooling procedures

on the sparse representations, i.e. a spatial pyramid. Finally, the texture feature is generated by concatenating the coefficients of different layers of the spatial pyramid.

After all the parameters are empirically set to be those providing the highest performance. For the color feature, the neighborhood size is set to be 23×23; while for the texture feature, the patch size for sparse coding is set to be 15×15, and the spatial max pooling is adopted within a 2×2 grid of each pixel, the number of the layers in spatial pyramid is set to be 3. Besides, the training samples are collected on a regular lattice with a stride of 4, which leads to approximately 500k training samples.

7.5.2 Results

In order to demonstrate the effects of integrating unsupervised segmentation, the experiments are conducted with different algorithms on MSRC21. We set RF_{std} as the standard random forests, and RF_{str} as the structured random forests. For different *agreement* definitions, '*reg*' represents the definition proposed by Kontschieder *et al.* (2011), and, '*usp*' denotes the one proposed in this chapter. Particularly, we use 'usp_{all}' represents the superpixel associations without optimization, and 'usp_{100}' and 'usp_{40}' represent the superpixel associations that are optimized into 100 units and 40 units respectively.

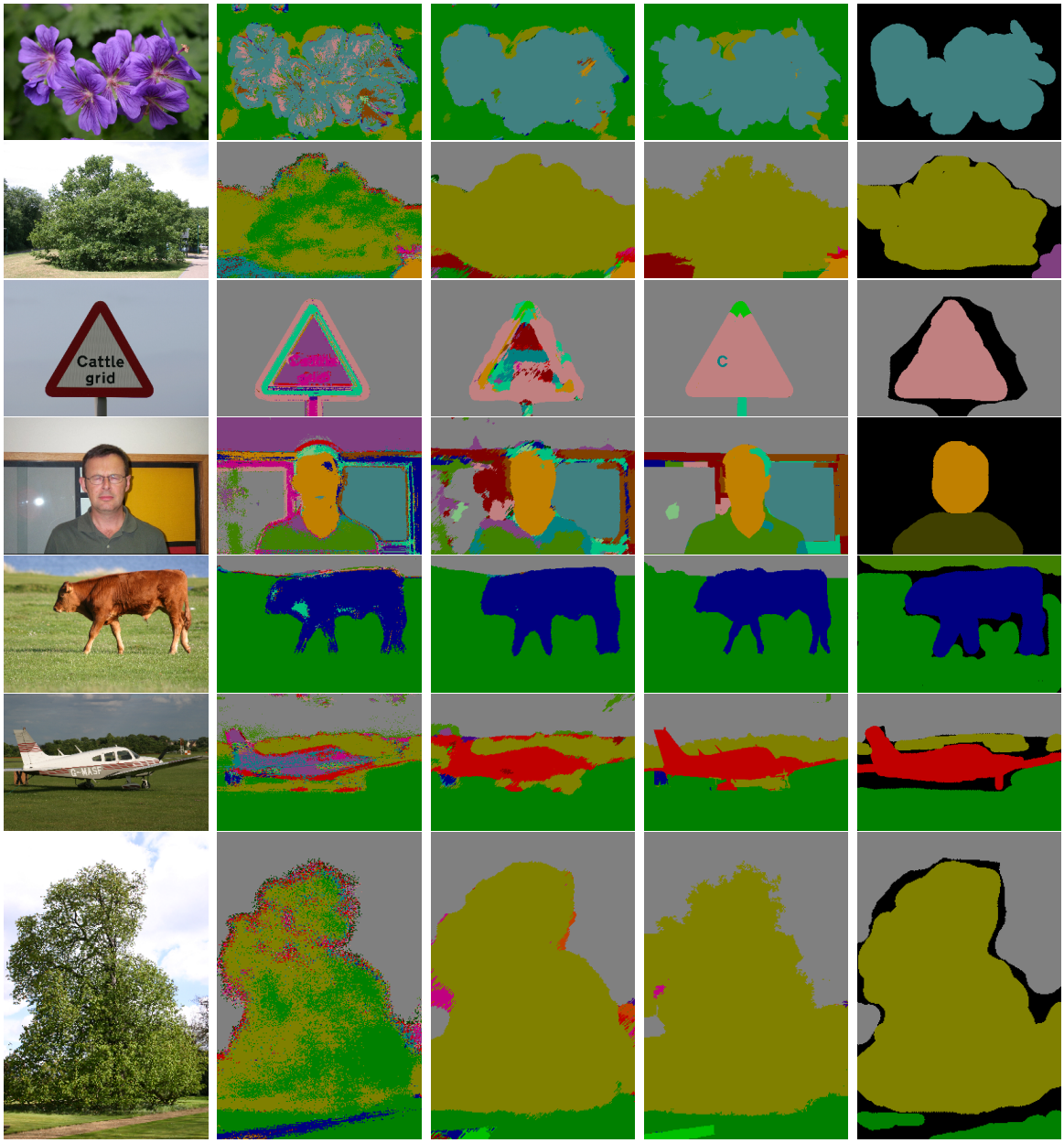
Table 7.1 shows the results of experiments. Some good results from our algorithms are shown in Figure 7.3. Also, Figure 7.4 demonstrates a few failures of our algorithm. Actually, the algorithm is successful in figuring out most of the objects but fails in labeling them with correct labels. This is most likely due to the weak feature scheme we used in the experiments.

7.6 Conclusion

In this chapter, we propose a generalized puzzle game framework for semantic segmentation, by which the unsupervised segmentations can be easily integrated into the labeling procedure. The experiment results show that the integration of unsupervised segmentation brings obvious improvement for the labelling algorithm. Compared with the *standard* random forests, the *structured* random forests improves the average class and the global accuracy by 4.3% and 3.8% respectively; while integrating the unsupervised segmentations, the maximum improvements (i.e. $RF_{str}(usp_{40})$) are 7.3% for average class accuracy and 6.6% for global accuracy. Also, within the *structured* random forests, the accuracy increments from using different unsupervised segmentations

Table 7.1: Accuracy of Different Random Forests on MSRC21

Algorithm	RF_{std}	$RF_{str}(reg)$	$RF_{str}(usp_{all})$	$RF_{str}(usp_{100})$	$RF_{str}(usp_{40})$
building	26.09%	29.84%	31.64%	32.21%	31.98%
grass	92.47%	93.85%	94.30%	94.22%	94.41%
tree	73.14%	77.64%	75.81%	78.19%	78.23%
cow	43.19%	48.26%	45.14%	46.94%	45.29%
sheep	54.94%	63.29%	65.27%	65.40%	68.86%
sky	91.25%	93.54%	95.98%	95.63%	95.45%
airplane	44.67%	55.41%	58.44%	58.79%	61.53%
water	51.07%	53.94%	59.39%	55.31%	55.70%
face	58.24%	64.00%	66.53%	67.62%	68.69%
car	32.84%	39.68%	42.97%	47.81%	48.88%
bicycle	60.66%	68.31%	68.46%	71.45%	74.12%
flower	37.06%	41.44%	40.48%	40.29%	40.97%
sign	23.35%	27.28%	39.30%	37.24%	37.66%
bird	7.81%	6.32%	2.38%	5.51%	3.57%
book	41.28%	49.55%	51.57%	53.05%	55.36%
chair	18.52%	21.85%	22.78%	22.98%	23.82%
road	58.74%	64.22%	74.51%	70.30%	73.28%
cat	43.38%	52.25%	55.52%	55.43%	59.76%
dog	20.11%	19.63%	24.43%	20.01%	20.53%
body	17.03%	18.79%	20.73%	20.28%	19.84%
boat	12.79%	10.05%	5.74%	5.39%	4.03%
Avg(Class)	43.27%	47.57%	49.59%	49.72%	50.57%
Global	57.77%	61.57%	63.91%	63.72%	64.37%



(a) (b) (c) (d) (e)

Figure 7.3: Semantic segmentation from different random forests: (a) the original, (b) *standard* random forest, (c) *structured* random forests, (d) *structured* random forest with unsupervised segmentation, (e) the ground truth with class labels shown in color (the void label is in black).

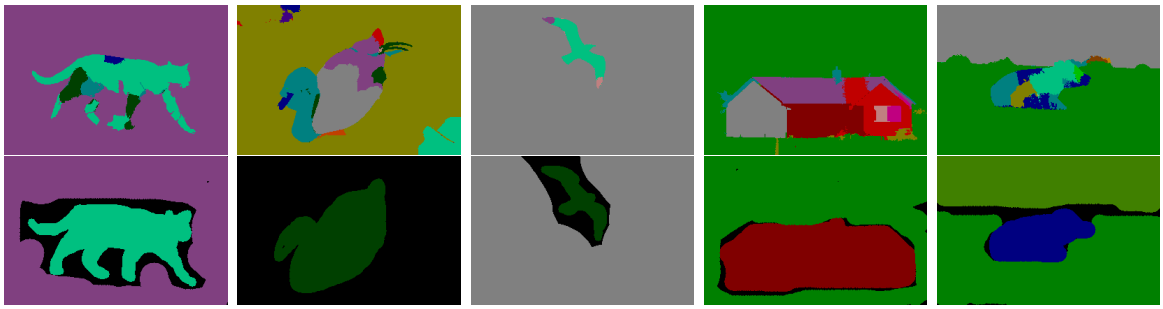


Figure 7.4: A few failures of out algorithm. The outputs of our algorithm are listed in the upper row, and their respective ground truth segmentations are in the lower row with class labels shown in color (the void label is in black). Most objects are properly figured out but with wrong labels, which is most likely due to the weak feature scheme that used in the experiments.

are around 2%.

The feature scheme we used in the experiments is a concatenation of *Lab* color values and a texture extracted by sparse coding techniques. When compared with those popular "deep features", it is less effective in capturing the segmentation information, which results in relatively low performance of out algorithm.

Chapter 8

Conclusion and Future work

8.1 Conclusions

Applications of image segmentation in the foreseeable future will be on demand. For example, automatic car driving, medical imaging, image editing for artistic purposes will require exact pixel-accurate segmentation of an object. Superpixel segmentation, as a preprocessing procedure, is widely used in image segmentation applications, by which object parts or image features can be extracted. In this thesis, we have examined the issues about using superpixels in image segmentation. Our research began by asking the following questions:

- Can we develop an efficient descriptor for superpixels which is able to improve the state-of-the-art segmentation algorithms?
- Can we find some methods for combining the superpixel descriptors that extracted from different feature spaces?
- Is there any method that can improve the performance of the handcrafted superpixel descriptors?
- Is there any new method that can generate image segmentation with superpixels more effective than the state of the art?
- Can we take use of the image cues in superpixel segmentations to improve the supervised segmentation?

These questions have been explored in one chapter, or relevant chapters jointly, as follows:

- **Proposing a novel covariance descriptor for superpixel and developing a corresponding segmentation algorithm "CCM" in Chapter 3.**

  The SAS algorithm (Li *et al.*, 2012) is a superpixel-based segmentation algorithm, which employs a bipartite graph to model the pixel-superpixel relations. The performance of this model is critically influenced by the similarity defined on superpixels. So, we proposed a color covariance matrix, as a representation of superpixel, adding to the discrimination ability of the color feature. And, we also propose the CCM algorithm, by which the similarity of superpixel covariance descriptor in measured not only in color space but also a manifold of the covariance matrix. With a properly defined metric for the covariance descriptor, CCM can perform better than SAS.

- **Proposing a multi-layer bipartite graph model "MBG-CCM" for fusing superpixel descriptors from different feature spaces in Chapter 4.**

  The feature fusing method for bipartite graph has not been sufficiently explored. In the CCM algorithm, the superpixel similarities measured in color space and covariance manifold are combined by a few matrix operators, which may be inapplicable with multiple features. So, we develop the MBG-CCM algorithm, which employs a multi-layer bipartite graph to formulate the similarities from different features. And, the layers of the graph are fusing by their subspace representation on a Grassmann manifold. Moreover, due to the high computational cost, the spectral clustering algorithm of normal multi-layer graph is intractable on our multi-layer bipartite graph. To solve this problem, we propose a transfer procedure for fusing the subspace representations, which is based on a property of singular value decomposition, i.e. the left and right singular vectors can be computed from each other. Theoretically, the MBG-CCM is able to generate robust final segmentation with multiple features.

- **Proposing a low-rank representation model "LRR-CCM" for utilizing subspace structure of the covariance descriptors of superpixel in Chapter 5.**

  The covariance descriptors are common handcrafted features for superpixel. But the research about how to use the covariance descriptors effectively is not sufficient. So, we develop a low-rank representation algorithm that can find the subspace structures for the covariance descriptor set. We combine this algorithm with CCM and propose the LRR-CCM algorithm, which measures the similarity

between the superpixels via the low-rank representation. The empirical experiments show that LRR-CCM is able to generate stable segmentation with noisy covariance descriptors.

- **Proposing a new over segmentation method "superpixel association" and a novel hierarchical segmentation algorithm "SHST" in Chapter 6.**

  Inspired by the fact that, in HBGF based segmentation algorithms, the pixels having the same pixel-superpixel relations are always partitioned into the same segment in the final segmentation, we propose the superpixel association method. Moreover, we proved that there exist an explicit relation between superpixel association and the pixels in the HBGF algorithm. This indicates that superpixel association is a good replacement of pixel in superpixel-based segmentation. Thus, we proposed a hierarchical segmentation algorithm, i.e. SHST. This novel segmentation algorithm takes superpixel associations as leaves and grows the segment by iteratively merging those closest regions. Since the number of superpixel associations is far less than that of pixels, the computational cost of the SHST is affordable. Another advantage of SHST is the number of segments in the final segmentation can be determined automatically, while for most of the existing algorithms, a specified number of segments is a prerequisite.

- **Developing a framework for integrating superpixel segmentation into semantic segmentation in Chapter 7.**

  Apart from those unsupervised segmentation methods, we extend our research into semantic segmentation. We propose a semantic segmentation algorithm, by which the superpixel segmentations can be easily integrated into the labeling procedure. This algorithm is named as *generalized puzzle game*, because its framework is inspired from the puzzle game. And, it contains three parts. The first part is called *game generator*, which is in charge of generating *label puzzle pieces* from the given image. The second is named *agreement*, which is a predefined similarity between the puzzle pieces and the current image labelling. The last one is called *game solver*, which is a predefined objective function, i.e. the condition that the final labelling should satisfy. A modified random forests algorithm is employed as a *game generator*, which can provide more proposals (i.e. *label puzzle pieces*) than tradition random forests. Moreover, we introduce a new definition of *agreement*, by which the superpixel segmentation can influence the similarity measuring between the *label puzzle pieces* and current labelling. And,

the final image labelling is obtained by alternatively updating the selection of puzzle pieces and the respective image labelling until the *game solver* converges.

8.2 Future work

Superpixel-based image segmentation is a research topic with an extensive and multi-faceted scope. In the following, we intend to discuss some current limitations and a few possible future directions that may extend from the work in this thesis.

- Computational cost related study

  For online segmentation applications or some applications running on devices with low computational capacity, the computational cost is critical. Unfortunately, in our study, this has not been sufficiently considered because our major work is concentrating on improving the accuracy of the segmentation. Actually, there is a conflict between computational cost and segmentation quality. For most existing segmentation models, the high segmentation accuracy will definitely result in the high computational complexity. However, one possible solution for this problem is parallel computing. And, for superpixel-based segmentation, this means new ensemble segmentation techniques should be proposed with the ability to be parallelized and operated on multiple processors.

- Application of superpixel association

  Superpixel association is, in fact, an ensemble of superpixels, which can be considered as a replacement of pixel in many image processing tasks. It is able to keep more image details especially the boundary information than superpixel, yet, its amount is far smaller than that of the pixel. To bring about a more accurate segmentation result but with a relatively low computational cost, it may be possible to modify the superpixel-based segmentation algorithms with superpixel association. For example, it would be interesting to see how the performance of the models proposed by Gould *et al.* (2014) might be possibly improved by replacing the superpixels with superpixel associations.

- Deep learning with superpixel

  Supervised image segmentation would benefit from integrating the unsupervised segmentation. In our study, we only combined the unsupervised segmentation into a two-stage labeling framework (i.e. generating proposals and optimizing

their combination). But, a more efficient and complicated structure, called deep learning, has been proposed recently (LeCun *et al.*, 2015), which achieves remarkable performance in supervised image segmentation. Moreover, studies in (Zheng *et al.*, 2015; Arnab *et al.*, 2016) show that the performance of VGG-16 network (Simonyan and Zisserman, 2014) is enhanced by considering the information from unsupervised segmentations. In order to boost both the training and inference stages, it may be possible to formulate a framework that integrates the superpixel segmentations with deep neural networks.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., and Süsstrunk, S. (2012). SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, *34*(11), 2274–2282.

Alush, A. and Goldberger, J. (2012). Ensemble segmentation using efficient integer linear programming. *IEEE transactions on pattern analysis and machine intelligence*, *34*(10), 1966–1977.

Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. (2011). Contour detection and hierarchical image segmentation. *TPAMI*, *33*(5), 898–916.

Arnab, A., Jayasumana, S., Zheng, S., and Torr, P. H. (2016). Higher order conditional random fields in deep neural networks. In *European Conference on Computer Vision*, 524–540. Springer.

Bo, L., Ren, X., and Fox, D. (2011). Hierarchical Matching Pursuit for Image Classification: Architecture and Fast Algorithms. In *Advances in Neural Information Processing Systems*.

Boden, M. A. (2006). *Mind as Machine: A History of Cognitive Science*. Clarendon Press.

Borenstein, E., Sharon, E., and Ullman, S. (2004). Combining top-down and bottom-up segmentation. In *Proceedings IEEE workshop on Perceptual Organization in Computer Vision*.

Borji, A., Cheng, M.-M., Jiang, H., and Li, J. (2014). Salient object detection: A survey. *arXiv preprint arXiv:1411.5878*.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Cai, J.-F., Candès, E. J., and Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, *20*(4), 1956–1982.

Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011). Robust principal component analysis? *Journal of the ACM (JACM)*, *58*(3), 11.

Chen, J. and Yang, J. (2014). Robust subspace segmentation via low-rank representation. *Cybernetics, IEEE Transactions on*, *44*(8), 1432–1445.

Cheng, B., Liu, G., Wang, J., Huang, Z., and Yan, S. (2010). Multi-task low-rank affinity pursuit for image segmentation. In *ICCV'10*, 2439–2446.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE transactions on pattern analysis and machine intelligence*, *17*(8), 790–799.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *TPAMI*, *24*(5), 603–619.

Couprie, C., Grady, L., Najman, L., and Talbot, H. (2009). Power watersheds: A new image segmentation framework extending graph cuts, random walker and optimal spanning forest. In *ICCV'09*, 731–738.

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Volume 1, 886–893. IEEE.

de Sa, V. R. (2005). Spectral clustering with two views. In *ICML workshop on learning with multiple views*.

Dhillon, I. S. (2001). Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD'01*, 269–274. ACM.

Dhillon, I. S., Guan, Y., and Kulis, B. (2004). *A unified view of kernel k-means, spectral clustering and graph cuts*. Citeseer.

Ding, L. and Yilmaz, A. (2008). Image segmentation as learning on hypergraphs. In *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*, 247–252. IEEE.

Dong, X., Frossard, P., Vandergheynst, P., and Nefedov, N. (2014). Clustering on Multi-Layer Graphs via Subspace Analysis on Grassmann Manifolds. *IEEE Trans. on Signal Processing, Vol.62, no.4*, 905–918.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object Detection with Discriminatively Trained Part Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.

Felzenszwalb, P. F. and Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *IJCV*, *59*(2), 167–181.

Fern, X. Z. and Brodley, C. E. (2004). Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the twenty-first international conference on Machine learning*, 36. ACM.

Fidler, S., Mottaghi, R., Yuille, A., and Urtasun, R. (2013). Bottom-Up Segmentation for Top-Down Detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Förstner, W. and Moonen, B. (2003). A Metric for Covariance Matrices. In E. Grafarend, F. Krumm, and V. Schwarze (Eds.), *Geodesy - The Challenge of the 3rd Millennium*, 299–309. Springer.

Franek, L., Abdala, D. D., Vega-Pons, S., and Jiang, X. (2010). Image segmentation fusion using general ensemble clustering methods. In *Asian Conference on Computer Vision*, 373–384. Springer.

Fred, A. L. N. and Jain, A. K. (2002). Evidence Accumulation Clustering Based on the K-Means Algorithm. *Lecture Notes in Computer Science*.

Freixenet, J., Muñoz, X., Raba, D., Martí, J., and Cufí, X. (2002). Yet another survey on image segmentation: Region and boundary information integration. In *ECCV'02*, 408–422. Springer.

Fu, Y., Gao, J., Hong, X., and Tien, D. (2015). Low Rank Representation on Riemannian Manifold of Symmetric Positive Definite Matrices. In *Proceedings of SDM*. SIAM.

Fukunaga, K. and Hostetler, L. (1975). The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, *21*(1), 32–40.

Fulkerson, B., Vedaldi, A., Soatto, S., *et al.* (2009). Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, Volume 9, 670–677. Citeseer.

Ganesh, A., Lin, Z., Wright, J., Wu, L., Chen, M., and Ma, Y. (2009). Fast algorithms for recovering a corrupted low-rank matrix. In *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2009 3rd IEEE International Workshop on*, 213–216. IEEE.

Gould, S., Gao, T., and Koller, D. (2009). Region-based segmentation and object detection. In *Advances in neural information processing systems*, 655–663.

Gould, S., Rodgers, J., Cohen, D., Elidan, G., and Koller, D. (2008). Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, *80*(3), 300–316.

Gould, S., Zhao, J., He, X., and Zhang, Y. (2014). Superpixel graph label transfer with learned distance metric. In *European Conference on Computer Vision*, 632–647. Springer.

Gruber, A. and Weiss, Y. (2004). Multibody factorization with uncertainty and missing data using the EM algorithm. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, Volume 1, I–707. IEEE.

Gu, X., Deng, J. D., and Purvis, M. K. (2014a). Improving superpixel-based image segmentation by incorporating color covariance matrix manifolds. In *Image Processing (ICIP), 2014 IEEE International Conference on*, 4403–4406. IEEE.

Gu, X., Deng, J. D., and Purvis, M. K. (2014b). Superpixel-based segmentation using multi-layer bipartite graphs and grassmann manifolds. In *Proceedings of the 29th International Conference on Image and Vision Computing New Zealand*, 119–123. ACM.

Gu, X., Deng, J. D., and Purvis, M. K. (2016). A hierarchical segmentation tree for superpixel-based image segmentation. In *Image and Vision Computing New Zealand (IVCNZ), 2016 International Conference on*, 1–6. IEEE.

Gu, X. and Purvis, M. (2016). Image Segmentation with Superpixel Based Covariance Descriptor. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 154–165. Springer.

Habiboğlu, Y. H., Günay, O., and Çetin, A. E. (2012). Covariance matrix-based fire and flame detection method in video. *Machine Vision and Applications*, *23*(6), 1103–1113.

Hamm, J. and Lee, D. D. (2008). Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the 25th international conference on Machine learning*, 376–383. ACM.

Ho, J., Yang, M.-H., Lim, J., Lee, K.-C., and Kriegman, D. (2003). Clustering appearances of objects under varying illumination conditions. In *Computer vision and pattern recognition, 2003. Proceedings. 2003 IEEE computer society conference on*, Volume 1, I–11. IEEE.

Hoffman, D. D. and Singh, M. (1997). Salience of visual parts. *Cognition*, *63*(1), 29–78.

Hosang, J., Benenson, R., and Schiele, B. (2014). How good are detection proposals, really? *arXiv preprint arXiv:1406.6962*.

Huang, D., Lai, J.-H., Wang, C.-D., and Yuen, P. C. (2016). Ensembling over-segmentations: From weak evidence to strong segmentation. *Neurocomputing*.

Huang, Q., Han, M., Wu, B., and Ioffe, S. (2011). A hierarchical conditional random field model for labeling and segmenting images of street scenes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 1953–1960. IEEE.

Jain, A. K. (1989). *Fundamentals of digital image processing*. Prentice Hall.

Joachims, T. (2003). Transductive Learning via Spectral Graph Partitioning. In *ICML'03*, 290–297.

Johnson, M., Shotton, J., and Cipolla, R. (2013). Semantic Texton Forests for Image Categorization and Segmentation. In *Decision Forests for Computer Vision and Medical Image Analysis*, 211–227. Springer.

Kim, H., Thiagarajan, J. J., and Bremer, P.-T. (2014). Image segmentation using consensus from hierarchical segmentation ensembles. In *Image Processing (ICIP), 2014 IEEE International Conference on*, 3272–3276. IEEE.

Kim, T. H., Lee, K. M., and Lee, S. U. (2010). Learning full pairwise affinities for spectral segmentation. In *CVPR'10*, 2101–2108.

Kluckner, S., Mauthner, T., Roth, P. M., and Bischof, H. (2009). Semantic Image Classification using Consistent Regions and Individual Context. In *BMVC*, Volume 6, 7.

Kohli, P., Torr, P. H., *et al.* (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, *82*(3), 302–324.

Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, *51*(3), 455–500.

Koltun, V. (2011). Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, *2*(3), 4.

Kontschieder, P., Bulo, S. R., Bischof, H., and Pelillo, M. (2011). Structured class-labels in random forests for semantic image labelling. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 2190–2197. IEEE.

Kontschieder, P., Bulò, S. R., Donoser, M., Pelillo, M., and Bischof, H. (2011). Semantic Image Labelling as a Label Puzzle Game. In *BMVC*, 1–12.

Kontschieder, P., Bulo, S. R., Pelillo, M., and Bischof, H. (2014). Structured labels in random forests for semantic labelling and object detection. *IEEE transactions on pattern analysis and machine intelligence*, *36*(10), 2104–2116.

Kontschieder, P., Fiterau, M., Criminisi, A., and Rota Bulo, S. (2015). Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision*, 1467–1475.

Kumar, S. *et al.* (2003). Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 1150–1157. IEEE.

Kviatkovsky, I., Adam, A., and Rivlin, E. (2013). Color invariants for person reidentification. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *35*(7), 1622–1634.

Ladicky, L., Russell, C., Kohli, P., and Torr, P. H. (2010). Graph cut based inference with co-occurrence statistics. In *European Conference on Computer Vision*, 239–253. Springer.

Lafferty, J., McCallum, A., Pereira, F., *et al.* (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the eighteenth international conference on machine learning, ICML*, Volume 1, 282–289.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.

LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 253–256. IEEE.

Levinshtein, A., Stere, A., Kutulakos, K. N., Fleet, D. J., Dickinson, S. J., and Siddiqi, K. (2009). tTurbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence*, *31*(12), 2290–2297.

Li, Z., Wu, X.-M., and Chang, S.-F. (2012). Segmentation using superpixels: A bipartite graph partitioning approach. In *CVPR'12*, 789–796.

Lin, Z., Chen, M., and Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*.

Liu, F., Lin, G., and Shen, C. (2015). CRF learning with CNN features for image segmentation. *Pattern Recognition*, *48*(10), 2983–2992.

Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., and Ma, Y. (2013). Robust recovery of subspace structures by low-rank representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *35*(1), 171–184.

Liu, G., Lin, Z., and Yu, Y. (2010). Robust subspace segmentation by low-rank representation. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 663–670.

Liu, G. and Yan, S. (2011). Latent low-rank representation for subspace segmentation and feature extraction. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 1615–1622. IEEE.

Liu, M.-Y., Tuzel, O., Ramalingam, S., and Chellappa, R. (2011). Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 2097–2104. IEEE.

Liu, W., He, J., and Chang, S.-F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 679–686.

Liu, Y., Zhang, D., Lu, G., and Ma, W.-Y. (2007). A survey of content-based image retrieval with high-level semantics. *Pattern recognition*, *40*(1), 262–282.

Ma, Y., Derksen, H., Hong, W., and Wright, J. (2007). Segmentation of multivariate mixed data via lossy data coding and compression. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (9), 1546–1562.

Ma, Y., Yang, A. Y., Derksen, H., and Fossum, R. (2008). Estimation of subspace arrangements with applications in modeling and segmenting mixed data. *SIAM review*, *50*(3), 413–458.

Malisiewicz, T. and Efros, A. A. (2007). Improving spatial support for objects via multiple segmentations.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. (2001). A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, Volume 2, 416–423. IEEE.

Mei, X., Sun, X., Dong, W., Wang, H., and Zhang, X. (2013). Segment-tree based cost aggregation for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 313–320.

Meilă, M. (2005). Comparing clusterings: an axiomatic view. In *ICML'05*, 577–584. ACM.

Mignotte, M. (2008). Segmentation by fusion of histogram-based k-means clusters in different color spaces. *IEEE Transactions on image processing*, *17*(5), 780–787.

Mignotte, M. (2014). A label field fusion model with a variation of information estimator for image segmentation. *Information Fusion*, *20*, 7–20.

Moore, A. P., Prince, S. J., and Warrell, J. (2010). Lattice Cut-Constructing superpixels using layer constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2117–2124. IEEE.

Moore, A. P., Prince, S. J., Warrell, J., Mohammed, U., and Jones, G. (2008). Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8. IEEE.

Ng, P. C. and Henikoff, S. (2003). SIFT: Predicting amino acid changes that affect protein function. *Nucleic acids research*, *31*(13), 3812–3814.

Nguyen, H., Yang, W., Shen, F., and Sun, C. (2015). Kernel low-rank representation for face recognition. *Neurocomputing*, *155*, 32–42.

Palmer, S. E. (1999). *Vision science: Photons to phenomenology*. MIT press.

Panagiotakis, C., Papadakis, H., Grinias, E., Komodakis, N., Fragopoulou, P., and Tziritas, G. (2013). Interactive Image Segmentation Based on Synthetic Graph Coordinates. *Pattern Recognition*, *46*(11), 2940–2952.

Rabinovich, A., Vedaldi, A., Galleguillos, C., Wiewiora, E., and Belongie, S. (2007). Objects in context. In *Computer vision, 2007. ICCV 2007. IEEE 11th international conference on*, 1–8. IEEE.

Rao, S. R., Mobahi, H., Yang, A. Y., Sastry, S. S., and Ma, Y. (2009). Natural image segmentation with adaptive texture and boundary encoding. In *Asian Conference on Computer Vision*, 135–146. Springer.

Ren, X. and Malik, J. (2003). Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, 10–17. IEEE.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, Volume 23, 309–314. ACM.

Shi, J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(8), 888–905.

Shotton, J., Johnson, M., and Cipolla, R. (2008). Semantic texton forests for image categorization and segmentation. In *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*, 1–8. IEEE.

Shotton, J., Winn, J., Rother, C., and Criminisi, A. (2006). Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision*, 1–15. Springer.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Tang, M., Gorelick, L., Veksler, O., and Boykov, Y. (2013). Grabcut in one cut. In *Proceedings of the IEEE International Conference on Computer Vision*, 1769–1776.

Tang, W., Lu, Z., and Dhillon, I. S. (2009). Clustering with multiple graphs. In *2009 Ninth IEEE International Conference on Data Mining*, 1016–1021. IEEE.

Torralba, A., Murphy, K. P., Freeman, W. T., Rubin, M. A., *et al.* (2003). Context-based vision system for place and object recognition. In *ICCV*, Volume 3, 273–280.

Tsai, Y.-H., Hamsici, O. C., and Yang, M.-H. (2015). Adaptive region pooling for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 731–739.

Unnikrishnan, R., Pantofaru, C., and Hebert, M. (2007). Toward objective evaluation of image segmentation algorithms. *TPAMI*, *29*(6), 929–944.

Van den Bergh, M., Boix, X., Roig, G., de Capitani, B., and Van Gool, L. (2012). SEEDS: Superpixels extracted via energy-driven sampling. In *European conference on computer vision*, 13–26. Springer.

Vedaldi, A. and Soatto, S. (2008). Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision*, 705–718. Springer.

Vega-Pons, S., Jiang, X., and Ruiz-Shulcloper, J. (2011). Segmentation ensemble via kernels. In *Pattern Recognition (ACPR), 2011 First Asian Conference on*, 686–690. IEEE.

Verbeek, J. and Triggs, B. (2007). Region classification with markov field aspect models. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, 1–8. IEEE.

Vincent, L. and Soille, P. (1991). Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE transactions on pattern analysis and machine intelligence*, *13*(6), 583–598.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, *17*(4), 395–416.

Wang, B., Hu, Y., Gao, J., Sun, Y., and Yin, B. (2015a). Kernelized Low Rank Representation on Grassmann Manifolds. *arXiv preprint arXiv:1504.01806*.

Wang, B., Hu, Y., Gao, J., Sun, Y., and Yin, B. (2015b). Low Rank Representation on Grassmann Manifolds: An Extrinsic Perspective. *arXiv preprint arXiv:1504.01807*.

Wang, H., Zhang, Y., Nie, R., Yang, Y., Peng, B., and Li, T. (2014). Bayesian image segmentation fusion. *Knowledge-Based Systems*, *71*, 162–168.

Wang, X., Du, J., Wu, S., Li, X., and Li, F. (2013). Cluster ensemble-based image segmentation. *International Journal of Advanced Robotic Systems*, *10*.

Wang, X., Li, H., Bichot, C.-E., Masnou, S., and Chen, L. (2013). A graph-cut approach to image segmentation using an affinity graph based on ℓ_0-sparse representation of features. In *ICIP'13*.

Wang, X., Li, H., Masnou, S., and Chen, L. (2013). Sparse Coding and Mid-Level Superpixel-Feature for 0-Graph Based Unsupervised Image Segmentation. In *International Conference on Computer Analysis of Images and Patterns*, 160–168. Springer.

Wang, X., Tang, Y., Masnou, S., and Chen, L. (2015). A global/local affinity graph for image segmentation. *IEEE Transactions on Image Processing*, *24*(4), 1399–1411.

Wang, Y., Guan, L., and Venetsanopoulos, A. N. (2012). Kernel cross-modal factor analysis for information fusion with application to bimodal emotion recognition. *IEEE Transactions on Multimedia*, *14*(3), 597–607.

Wei, X., Yang, Q., Gong, Y., Yang, M.-H., and Ahuja, N. (2016). Superpixel hierarchy. *arXiv preprint arXiv:1605.06325v1*.

Wertheimer, M. (1938). Laws of organization in perceptual forms.

Wright, J., Ganesh, A., Rao, S., Peng, Y., and Ma, Y. (2009). Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, 2080–2088.

Wu, S., Wang, X., Ye, Q., and Dong, J. (2012). Region clustering with high level semantics for image segmentation. In *Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on*, Volume 2, 946–950. IEEE.

Yao, J., Fidler, S., and Urtasun, R. (2012). Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 702–709. IEEE.

Zhang, L., Zhang, Q., Zhang, L., Tao, D., Huang, X., and Du, B. (2015). Ensemble manifold regularized sparse low-rank approximation for multiview feature embedding. *Pattern Recognition*, *48*(10), 3102–3112.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 1529–1537.

Zhou, D. and Burges, C. J. (2007). Spectral clustering and transductive learning with multiple views. In *ICML'07*, 1159–1166. ACM.

Zhou, P., Du, L., Wang, H., Shi, L., and Shen, Y.-D. (2015). Learning a Robust Consensus Matrix for Clustering Ensemble via Kullback-Leibler Divergence Minimization. In *IJCAI*, 4112–4118.

Zhou, Z.-H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.

Zhu, H., Lu, S., Cai, J., and Lee, Q. (2015). Diagnosing state-of-the-art object proposal methods. *arXiv preprint arXiv:1507.04512*.

Zhu, H., Meng, F., Cai, J., and Lu, S. (2016). Beyond pixels: A comprehensive survey from bottom-up to semantic image segmentation and cosegmentation. *Journal of Visual Communication and Image Representation*, *34*, 12–27.