Semantic Image Understanding: From Pixel to Word

Hao Fu, BEng.

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

September 2012

Abstract

The aim of semantic image understanding is to reveal the semantic meaning behind the image pixel. We categorize semantic image understanding into two broad categories: pixel-level and image-level semantic image understanding. While pixel-level image understanding aims to obtain the semantic meaning of each pixel, image-level understanding aims to obtain the semantic meaning of the whole image, and both levels involve feature extraction and combination. In this thesis, we study semantic image understanding and have made following novel contributions:

We investigated the utility of Multiple Kernel Learning (MKL) for feature combination. We introduced the concept of kernel histogram. We observed that the kernel histograms of different features are usually very different, and argued that traditional MKL's linear kernel combination strategy is not particularly meaningful. Then we proposed the concept of Relative Kernel Distribution Invariance (RKDI) for kernel combination, and have developed a very simple histogram matching technique to achieve RKDI by transforming different kernel histograms to a canonical histogram. We have also developed two kinds of measure for automatically choosing the canonical histogram. Extensive experiments on various computer vision and machine learning datasets have shown that calibrating the kernels to a canonical histogram before they are linearly combined can always achieve a performance gain over state of the art MKL methods.

For the problem of understanding image at the pixel level, we advocate the segment-then-recognize strategy. We have developed a new framework which tries to integrate semantic segmentation with low-level segmentation by introducing a semantic feature feedback mechanism. Experiments on two well-known datasets have confirmed that our new segmentation method can indeed produce regions that are more object-consistent. Besides this, we have also developed a novel idea trying to integrate semantic segmentation with interactive segmentation. We treat the semantic segmentation module as an unreliable teacher which automatically generates tokens to guide the interactive segmentation module. Some qualitative results have shown the promise of this approach.

We follow the segment-then-recognize strategy, and found this philosophy works very well on medical image data. We have developed a novel algorithm termed as GlandVision for detecting glandular structures contained in a microscopic image of human tissue. We first transform the image from Cartesian space to polar space and introduce a novel random field model with an efficient inference strategy that uses two simple chain graphs to approximate a circular graph to infer possible boundary of a gland. We then develop a visual feature based support vector regressor (SVR) to verify if the inferred contour corresponds to a true gland. And finally, we combine the outputs of the random field and the regressor to form the GlandVision algorithm for the detection of glandular structures. In the experiments, we treat the task of detecting glandular structures as object (gland) proposal, detection and segmentation problems respectively and show that our new technique outperforms state of the art computer vision algorithms in all these tasks.

For the problem of semantic image understanding at the image level, we aim to utilize the large repository of image data present on the web together with their associated tags. We have developed a novel random forest model for image annotation and image retrieval. On one hand, the tree structure inherited in the random forest serves as an efficient data structure for storing and fast retrieving image data; on the other hand, we utilize the tag information to guide the generation of the random forest. Thus the human knowledge (in the form of tags) is implicitly embedded in our random forest, helping us to tackle the semantic gap problem lying in the core part of image-level semantic image understanding. Different from conventional random forest model, which fuse the information contained at each leaf node individually, our method treats the random forest as a whole, and introduces the new concepts of semantic nearest neighbors (SNN) and semantic similarity measure (SSM). Based on which we performed the image retrieval task and casted the task of image annotation as a learning to rank problem. Our new technique is intrinsically scalable and we will present experimental results to demonstrate that it is competitive to the state of the art methods.

.

List of Publications

1. **Hao Fu** and Guoping Qiu. Integrating Low-level and Semantic Features for Object Consistent Segmentation. *Accepted for Neurocomputing,* 2012 [Chapter 4]

2. **Hao Fu** and Guoping Qiu. Histogram Matching for Kernel based Feature Combination. *International Conference on Machine Learning (ICML) workshop on Object, functional and structured data: towards next generation kernel-based methods.* 2012 [Chapter 3]

3. **Hao Fu**, Guoping Qiu, and Hangen He. Feature Combination beyond Basic Arithmetics. *In British Machine Vision Conference (BMVC).* 2011 [Chapter 3]

4. **Hao Fu** and Guoping Qiu. Integrating Low-level and Semantic Features for Object Consistent Segmentation. *In International Conference on Image and Graphics (ICIG),* 2011 [Chapter 4]

5. **Hao Fu**, Guoping Qiu, Mohammad Ilyas and Jie Shu. *GlandVision: A Novel Polar Space Random Field Model for Glandular Biological Structure Detection. In British Machine Vision Conference (BMVC),* 2012 [Chapter 5]

6. **Hao Fu**, Qian Zhang, and Guoping Qiu. Random Forest for Image Annotation. *In European Conference on Computer Vision (ECCV),* 2012 [Chapter 6]

7. **Hao Fu**, and Guoping Qiu. Fast Semantic Image Retrieval based on Random Forest. *In ACM Multimedia (ACM MM),* 2012 [Chapter 6]

8. Ligang Zheng, Guoping Qiu, Jiwu Huang, and **Hao Fu**. Salient Covariance for Near-duplicate Image and Video Detection. *In International Conference of Image Processing (ICIP),* 2011

Acknowledgements

In writing this acknowledgement, I realized that my 21 years' journey as a student has come to an end. Before moving to the next journey, this acknowledgement is really a good opportunity for me to look back and give my sincere thanks to all the people who have helped me through this journey.

Firstly, my deepest thanks go to my supervisor: Dr.Guoping Qiu, who has guided me through all the steps in the whole PhD process. He is always so encouraging, enthusiastic and knowledgeable. I feel so lucky to have him as my supervisor and he will be my supervisor forever. I would also like to thank my second supervisor: Dr.Bai Li, who has also helped me a lot during my PhD life.

Special thanks go to my two examiners: Dr.Tony Pridmore and Professor.Chang-Tsun Li, whose careful proofreading and insightful suggestions have led to an improvement of the quality of this thesis.

Thanks to all the members in IMA and VIPLAB. All of you together have made our group such a vibrate and stimulating research environment. Special thanks go to my colleagues: Orod Razeghi, Bozhi Liu, Wenwen Bao, Qian Zhang, Jie Shu, Yujie Mei, Min Zhang, Mercedes Torres, Salvador Garcia Bernal, Sameh Zakhary, Poay Hoon Lim, Dr.Hieu V. Nguyen, Dr.Tao Zhang, Dr.Jianyong Sun, Dr.Peer-Olaf Siebers and Nadine Holmes.

I would also like to thank my teachers and colleagues in China. They are Professor.Hangen He, Professor.Bin Dai, Professor.Xin Xu, Dr.Xiangjing An, Dr.Daxue Liu, Dr.Jinze Song, Jian Li, Yiming Nie, Yuqiang Fang, Jun Tan and Erke Shang. Special thanks go to Dr.Tao Wu, whose encouragement has helped me so much to overcome my lowest time in PhD life, when my first paper got rejected.

Thanks to all the friends who have added so much joy to my daily life. They are Huanlai Xing, Haowen Ruan, Fangfang Zhu, Haiping Shen, Bei Zhang, Feng Hu, Heng Yang, Xiaoqin Sun, Florent Balestrieri, George Giorgidze, Wei Chen, Joao Ferreira, Sihai Yang, Xue Han, Kai Wang, Baobin Liao, Xiaocong Zhong and Tao Ma.

This thesis will not be possible without all the unconditional help and love from my girlfriend, Wenjuan Yuan.

I would also like to thank my country and University of Nottingham to offer me the PhD scholarship, without which this PhD journey won't be possible. The picturesque environment in the University of Nottingham has left me so many enjoyable memories, which I will miss forever.

This thesis is dedicated to my family, especially my uncle who has just passed away. Wish you rest in peace in the heaven.

Contents

Abstract i

List of Publications iii

Acknowledgements iv

List of Figures ix

List of Tables xiv

1 Introduction 1
- 1.1 Motivations . . . 1
- 1.2 Technical Challenges . . . 2
- 1.3 Contributions . . . 4

2 Literature and Methods 7
- 2.1 Feature Extraction and Feature Combination . . . 8
  - 2.1.1 Feature Extraction . . . 8
  - 2.1.2 Feature Combination . . . 10
- 2.2 Classification Methods . . . 11
  - 2.2.1 From SVM to MKL . . . 12
  - 2.2.2 Peril and Promise for Nearest Neighbor method . . . 15
  - 2.2.3 How can Random forest help? . . . 17
  - 2.2.4 Integrating Classifiers in a Random Field Model . . . 19
- 2.3 Semantic Image Understanding at the Pixel Level . . . 21
  - 2.3.1 Object Detection . . . 21
  - 2.3.2 Semantic Segmentation . . . 23
- 2.4 Semantic Image Understanding at the Image Level . . . 25
- 2.5 Summary . . . 26

3 Feature Combination beyond Basic Arithmetics **28**
3.1 Introduction . . . 29
3.2 Kernel Histogram and Histogram Matching . . . 30
3.3 Standardizing Kernel Values through Histogram Matching . . . 31
3.4 Transforming Kernels to Positive Semi-Definite . . . 34
3.5 Automatically Choosing the Canonical Kernel Histogram . . . 36
3.6 Experimental Results . . . 38
3.6.1 Adaptive Kernel Combination . . . 38
3.6.1.1 Experiment Setup . . . 38
3.6.1.2 Oxford Flowers Dataset . . . 39
3.6.1.3 MSRC21 Dataset . . . 41
3.6.1.4 UCI Machine Learning Repository . . . 43
3.6.1.5 Results Summary . . . 44
3.6.2 Non-adaptive Kernel Combination . . . 45
3.6.2.1 Corel5K Dataset . . . 45
3.6.2.2 Caltech101 in 39 Kernels [84] . . . 47
3.6.2.3 Results Summary . . . 48
3.7 Concluding Remarks . . . 49

4 Integrating Low-level and Semantic Features for Object Consistent Segmentation **51**
4.1 Introduction . . . 52
4.2 Combining Low-level and Semantic Segmentation for Object Proposal . . . 54
4.2.1 The Advantage of Choosing Regions as Processing Primitives . . . 54
4.2.2 The Proposed Framework . . . 54
4.2.3 Spectral Methods for Low-level Segmentation . . . 56
4.2.4 Pixel or Superpixel based Semantic Segmentation . . . 57
4.2.5 Mid-level Segmentation with Semantic Feature Feedback 58
4.2.6 Object-like Primitives Labeling . . . 59
4.3 Experiments . . . 61
4.3.1 Segmentation Quality . . . 61
4.3.2 Comparison of Pixel Labeling Accuracy on MSRC21 . . 62
4.3.3 Experiments on Stanford Background Dataset . . . 65
4.3.4 Computational Complexity Analysis . . . 67
4.4 Hard Integration . . . 67
4.5 Concluding Remarks . . . 69

5 GlandVision: A Novel Polar Space Random Field Model for Glandular Biological Structure Detection **71**
5.1 Introduction . . . 72

5.2 Related Work . . . 73
5.3 A Novel Random Field Model for Gland Proposal . . . 74
5.3.1 A Novel Random Field in the Polar Space . . . 75
5.3.2 Inference . . . 76
5.3.3 Unary Potential . . . 77
5.3.4 Pairwise Potential . . . 80
5.4 Verification by a Visual Feature based Regressor . . . 81
5.5 GlandVision: Integrating Random Field with Regressor . . . 83
5.6 Experiments . . . 83
5.6.1 Gland Proposal Accuracy . . . 84
5.6.2 Gland Detection Accuracy . . . 85
5.6.3 Gland Segmentation Accuracy . . . 87
5.7 Concluding Remarks . . . 89

6 Random Forest for Image Annotation and Retrieval 91
6.1 Introduction . . . 92
6.2 The Construction of the Random Forest . . . 94
6.3 Image Retrieval based on Random Forest . . . 97
6.3.1 Example-based Retrieval . . . 97
6.3.2 Keyword-based Retrieval . . . 97
6.4 Image Annotation based on Random Forest . . . 98
6.4.1 Tag Prediction based on Semantic Neighbors . . . 98
6.4.2 Image Annotation as Learning to Rank Semantic Neighbors . . . 99
6.5 Experiments . . . 106
6.5.1 Example-based Image Retrieval . . . 106
6.5.2 Keyword-based Image Retrieval . . . 109
6.5.3 Image Annotation Performances . . . 110
6.5.4 Learning to Rank vs. Ad Hoc Annotation Functions . . . 114
6.6 Concluding Remarks . . . 115

7 Concluding Remarks 116
7.1 Main Contributions . . . 116
7.2 Limitations and Suggestions for Improvement . . . 118
7.3 Summary . . . 119

A More visual examples 121

Bibliography 125

List of Figures

2.1 A ‘car’ on the left and a ‘pedestrian’ on the right. In courtesy from http://web.mit.edu/torralba/www/carsAndFacesInContext.html 23

3.1 Typical feature combination methods always represent features into their kernel forms. These kernels are then combined. Traditionally, the kernels are combined directly through one of the methods in (3.2) or (3.3). In this chapter, we proposed to add a histogram matching module before these kernels are combined by one of the methods in (3.2) or (3.3). 33

3.2 The blue points correspond to the histogram of intra kernel, while the red points correspond to the inter kernel. The regression results are shown as the blue line and the red line, and the shaded area corresponds to the area in one standard deviation. To accelerate the regression speed, the histogram bin index, which ranges from 1 to 1500, are linear mapped to (-1,1). Ideally, the intra kernel value should be larger than inter kernel value; therefore the combined histogram should be double peaked. 40

3.3 Kernel histograms of three features used in MSRC21 ((a) to (c)) and seven features ((d) to (j)) used in Oxfordflowers. The histogram in the red box is chosen as the standard histogram. . . . 43

3.4 Kernel histograms of the features used in [46]. The histogram in the red box is chosen as the standard histogram, which corresponds to the feature of hue descriptors extracted at Harris-Laplacian interest points 46

3.5 Histograms in Fig.3.4 after histogram matching. 47

3.6 Some representative kernel histograms among the 39 kernels used in [84], the one in the red box is the best performing canonical kernel histogram. 48

3.7 The classification results on Caltech101. From the figure, we can see the average of histogram matched kernels can always perform better than averaging original kernels. Note that the author of [84] report results on five random splits of the dataset. However, they have only released their gram matrixes of one split. We did experiments only on this split. This results in the slight difference between our implementation on average and the average accuracy reported in [84]. 49

4.1 Top row: (left to right) original image, ground truth segmentation, pixel based semantic labeling [9] without CRF [57] post processing, two groups of superpixel based methods [101] by varying the parameters of mean shift; Middle row: a state of the art spectral segmentation method named Full Pairwise Affinity (FPA) [102] which considers only color features to produce K (K=3,4,...7) clustering; Bottom row: our full model by combining color features and semantic features to produce K clustering. It is seen that in all cases our method consider the head and the body of the sheep to be one object. 53

4.2 The flow chart of the proposed framework 55

4.3 Segmentation covering score of different methods. X axis represents dividing an image into # regions, while Y axis corresponds to the accuracy. NCut means regions generated by normalized cut, FPA means Full Pairwise Affinity model in [102] 62

4.4 Some examples of the results of three segmentation methods. Each image is segmented into a predefined 7 regions. From left column to right column: original image; results generated by NCuts, Full Pairwise Affinity (FPA) [102], and our full model. The results of a typical semantic segmentation algorithm [101] is also shown in the last column. 63

4.5 Global pixel accuracy on MSRC21 64

4.6 From left to right: original image; object-like regions generated by our full model; superpixels commonly used in conventional semantic segmentation methods 65

4.7 Semantic class accuracy on Stanford background dataset. 'Ours+Augment' represents results obtained by augmenting the training set. . . . 66

4.8 The procedure for generating the tokens. Based on the outputs from the semantic segmentation algorithm and their predicted semantic classes, the highly probable large regions went through a series of morphological operations including gaussian filtering, threshold, thinning and dilation. 68

4.9 From left to right: original image; outputs from the semantic segmentation algorithm; the predicted semantic class; automatically generated tokens; segmentation results generated by [114] based on the tokens. 69

5.1 A microscopic image of human colon tissue containing glands. The glands are manually annotated in blue solid color as shown in the right image. 73
5.2 For the polar image, its horizontal axes corresponds to the distance to the origin, while the vertical axes corresponds to the angle ranges from 1 to 360. A, B and C correspond to cases where the polar space's origin is inside a gland while D corresponds to the case where the polar space's origin is not inside a gland. We can clearly see a continuous line structure in A, B and C, while this kind of structure cannot be seen in D. 74
5.3 (a) The graphical model of our CRF model; (b) Each row of the polar image is assigned a random variable; (c) The factor graph of our CRF model. 77
5.4 An Efficient Inference Strategy. (a) An image is transformed into two polar images, one's θ ranges from 0 to 2π, and the other's ranges from π to 3π. The Viterbi inference algorithm is performed separately on these two polar images, and the results are shown in (b) and (c). We can see that the resulted contour is not a closed shape. This is because we use the chain structure to approximate the circular graph. These two results are then combined using Algorithm 2 to generate the final result shown in (d). 78
5.5 Illustration of Unary Potential. Given an image as shown in (a), it is firstly transformed into the polar space (b). The histogram equalized polar image is shown in (c) left, and the unary potential as defined in (5.4) is shown in (c) right in pseudo color. This kind of unary potential definition is flawed which will result in the bad detection result as shown in (d) and (e). Instead, we first calculate the cumulative edge map (g) based on the edge map (f). Then this cumulative edge map is combined with the original polar image according to (5.5) to generate our new unary potential shown in (h). Based on this unary potential, we obtain the correct results as shown in (i) and (j). 80
5.6 A Complete GlandVision Procedure 83
5.7 Nott-Gland dataset. 84

5.8 (a) The DR/STN curve of [135] and our Random Field (RF) model at threshold 0.5 (RF_0.5) and 0.8 (RF_0.8). (b) The original microscopic image. (c) The ground truth image of glands. (d) The results obtained by [135]. (e) The results obtained by our full model. 86
5.9 (a) Precision-Recall curve of different methods. (b)-(d) The histogram of original phog_SVR values, the Random Field (RF) outputs and their summed results. (e)-(g) The histogram of the phog_SVR values, RF outputs and summed value after Histogram Matching (HM) (please refer to Chapter 3). We could see that the histogram of phog_SVR and RF becomes the same and the HM operator. 87
5.10 MAP value at different thresholds. 88
5.11 (a) The original microscopic image. (f) The ground truth image of glands. (b)-(e) The detection results obtained by [61] with threshold varies from 0.5 to 0.8 (g)-(j) The results obtained by by our model. As the threshold increases, the bounding boxes have to be more accurate. As the accuracy requirements increases (higher threshold), the detection result of [61] dropped significantly. 88
5.12 Segmentation results. 89

6.1 Illustrative example on the effect for the size of the dataset. This example is adapted from [44]. 93
6.2 An example showing the concepts of semantic neighbor set and semantic neighbors. A query image passes through all random trees. The training images stored at the leaf nodes on which the query image falls into form the semantic neighbor set. Based on this, the semantic similarity measure (SSM) between the query and a particular training image is calculated as the number of times that the particular training image appears in the semantic neighbor set. A larger SSM indicates higher similarity. 96
6.3 Distribution of the SSM (i.e. $count$) value across different datasets. Randomly sampled images from each dataset are fed into the random forest. The top K semantic nearest neighbors retrieved with their SSM values are gathered to plot this graph. 100
6.4 From left to right: the matrix $\mathbf{\Psi} = [\mathbf{\Psi_1}; \mathbf{\Psi_2}; ...; \mathbf{\Psi_M}]$, where $\mathbf{\Psi_j}$ is defined in (6.7); $\mathbf{\Psi_{RO}}$ obtained by rearranging $\mathbf{\Psi}$ in its Right-Ordered form; the cumulative sum of each row of $\mathbf{\Psi_{RO}}$ in the reverse order. 106
6.5 A comparison of K-Nearest Semantic Measure (KNSM) between our method and JEC [1]. 108

6.6 Some examples of the semantic nearest neighbor images retrieved by our random forest method and by JEC method. The tags associated with each image are also shown beneath each image. The tags which are in accordance with the test image are colored in blue and underlined, while the false tags are colored in red. The numbers underneath the SNN images are the values of SSM. 108
6.7 The MAP achieved by our Random Forest (RF) method with comparison with previously reported results. 110
6.8 The relation between the system performance F_value ($F\_value = 2 * Precision * Recall/(Precision + Recall)$), and the parameters of N_T (number of trees) and K (number of Nearest Neighbors used for prediction). From left to right: the results on Corel5K, results on IAPR-TC12 and results on ESP game. . . 111
6.9 Some more examples of the nearest neighbor images retrieved by our random forest method and by JEC method. The Semantic Similarity Measure or the rank indexes are shown beneath each retrieved image. The images in the top three rows are from the ESP game, while the bottom three rows are from IAPR-TC12. From this figure, we can see that our method outperforms JEC, and both these two datasets contain some duplicate images that both exist in the training set and testing set. 112

A.1 Some examples of the nearest neighbor images retrieved by our random forest method and by JEC method [1], together with their semantic similarity measure (SSM) or the rank index shown under each image. 122
A.2 Examples from EPS-game 123
A.3 Examples from IAPR-TC12 124

List of Tables

3.1 Experimental results on Oxfordflower. OBSCURE[93] is one of the state-of-art MKL solvers. HE is short for Histogram Equalization, and Gaussian represents a predefined $\mathcal{N}(0,1)$ normal distribution. From the table, we could see that the best performed two kernels: K_color and K_hsv, have the 2 smallest norm score and their KA score are ranked as 3 and 1 respectively. 41

3.2 Experimental Results on MSRC21. Accuracy1 = Per-class region-wise accuracy. Accuracy2 = Overall region-wise accuracy. Accuracy3 = Per-class pixel-wise accuracy. Accuracy4 = Overall pixel-wise accuracy. From the table, we could see that K_texton performs best, and it also has the lowest norm score and the highest KA score among the three kernels. 42

3.3 Experimental Results on UCI datasets. The above table shows the results on Sonar dataset while the bottom is for breast. K_rbf performs best across the two datasets, and it also has the lowest norm score and the highest KA score among the three kernels. . 44

3.4 Image Annotation Performances on Corel5K Dataset. HM is short for **H**istogram **M**atching. Rate+ is the number of tags whose recall is above zero. 47

4.1 Segmentation covering score of the segmentation pool 62

4.2 Semantic labeling accuracy on MSRC21 when each test image is partitioned into a fixed 8 regions. While 'Global' means the overall pixelwise labeling accuracy, 'Average' means the average of classwise accuracy. The results are shown in terms of percentages. 64

5.1 Segmentation Accuracy 89

6.1 Image Annotation Performances on Corel5K, IAPR-TC12 and ESP game. RF represents our Random Forest method. RF_$count$ denote $f(c_i) = c_i$, RF_$count^2$ denote $f(c_i) = {c_i}^2$ and RF_$optimize$ denote $f(c_i)$ is learned based on our optimization framework . . 113

6.2 The number of correctly predicted tags on each dataset. RF_$optimize$ consistently outperforms the ad-hoc functions. . . . 114

To my family...

Chapter 1

Introduction

This thesis deals with the problem of semantic image understanding, the goal of which is to reveal the semantic meaning lying behind the pixels of an image. This task is always the holy grail in computer vision, and (partly) solving it will be of great significance with many potential applications, such as autonomous driving, Content Based Image Retrieval (CBIR), Intelligent Robots, Human Computer Interaction, etc.

In the following of this chapter, we will firstly highlight the motivations behind this thesis in section 1.1. In section 1.2 we analyze the challenges which make the semantic image understanding problem a difficult task, and we present our contributions to the resolution of these challenges in section 1.3.

1.1 Motivations

Digital images are ubiquitous in our daily life: almost every mobile phone has a camera embedded in it; CCTV camera is placed across the whole UK; Facebook, Flickr or Youtube are consistently encouraging people to upload their own pictures or videos; the 3D movie is gradually dominating the cinemas over the

traditional 2D movie... It is no doubt that digital image technology has brought many joys and more safety to our daily life. What accompanies these emerging images and videos is an urgent need for intelligent techniques to store, retrieve, or even 'understand' the images.

To endow the computer the ability to understand the image will bring us more joys and convenience. For example, you do not bother to search folder by folder when you want to find a specific image from your enlarging image collections. If the computer can understand the semantic content of an image captured from a camera mounted on a car, then it can drive the car automatically. If the CCTV camera can understand the semantic meaning of its captured video, then it can immediately report to the police in case of danger or criminal.

1.2 Technical Challenges

Owing to the fast growing computational power and more advanced machine learning techniques, recent years have witnessed a fast development in semantic image understanding. A huge gap, however, still exists between human and computer: We can easily recognize about 30,000 different categories [2] with high precision; but the computer is still struggling to recognize 256 categories [3].

Researchers try to tackle the semantic image understanding problem from different angles: **Image Classification or annotation** deals with the problem of whether an image contains a specific object; **Object Detection** tries to answer where the object is, if present; **Semantic Segmentation** aims to assign a semantic label to every single pixel contained in the image, but not necessarily count the number of objects. These seemingly different tasks are in fact closely related to each other, and they share at least one common module: feature extraction/combination.

One of the most difficult problems in image understanding is perhaps the high intra-class variation V.S. low inter-class variation, and a good feature extraction module should be designed to directly tackle this problem. A good feature should be both invariant and discriminative [4]. Here, 'invariant' means the feature should not change among the different entities in the same semantic class, i.e. it can reduce the intra-class variation. It also should be 'discriminative', which means it is helpful for distinguishing one semantic class from another, and therefore can enlarge the inter-class variation. As features are so important, there are numerous methods in the literature designed for extracting meaningful features [5–8].

As different features represent different aspects of an image, it is often helpful and sometimes necessary to combine various features together in order to gain a comprehensive understanding of an image. Therefore, a natural question arises: *how to effectively and efficiently combine these different kinds of features?*

To understand the image at the pixel level, we can adopt two possible strategies: the first one is to consider each pixel or superpixel (a small group of pixels which share similar low-level properties) as the processing primitive, and directly assign a semantic label to it. Most of the state of the art methods [9–15] adopt this strategy. Another method is to adopt a segment-then-recognize strategy, i.e. first segmenting an image into different regions, in the hope that each region will correspond to one object, then recognizing each object one by one.

One of the main drawbacks of the first strategy is that it loses the information of an object, as each 'object' is just a group of pixels or superpixels with the same semantic labels. On the contrary, by adopting the second strategy, as long as the segmentation module can produce object-consistent regions, the output will naturally be an object consistent result. However, the biggest challenge behind the second strategy is *how to design an object-consistent segmentation method?*

Although the idea of understanding the semantic meaning of each pixel in an image seems appealing, it is still too difficult at the current level of technology. If we take a step back, and only consider understanding the image at the image level, this task perhaps becomes easier, while it is still of great practical significance, especially in multimedia.

Different from understanding an image at the pixel level, which usually needs lots of manually labeled ground truth data to train the algorithm, there already exists a huge amount of images on the internet with human labeled tags. These tags reflect human's understanding of an image. We believe this large repository of image data and their associated tags contain valuable knowledge worth exploring. Efficiently utilizing these information and knowledge could help us overcome the semantic gap problem lying in the core of the image-level semantic image understanding problem. *How to design a scalable algorithm that can handle large-scale image data and efficiently utilizing their (noisy) tags is also a challenging task.*

1.3 Contributions

In this thesis, we have made the following contributions:

- In the feature combination scenario, we noticed that kernel based feature combination techniques, such as Multiple Kernel Learning (MKL), have drawn increasing interest recently. In Chapter 3, we transfer the concept of the histogram, which is commonly used in image processing area, to the kernel matrix, and proposed the concept of kernel histogram. We make an important observation that the kernel histograms of different features are usually very different, thus making MKL's linear kernel combination strategy not particularly meaningful. To remedy this, we have developed

a very simple histogram matching based technique to transform the histograms of different kernels to be the same before they are linearly combined. Extensive experiments on various computer vision and machine learning datasets have shown that our method can always boost the performance of MKL. Furthermore, our technique is also beneficial for unsupervised cases, where there is a need to sum different kernels (similarity measures) together.

- For the problem of pixel-level semantic image understanding, we advocate the usage of the segment-then-recognize strategy. Previous methods tend to abandon this approach partly because of the difficulty in generating object consistent regions. In Chapter 4, we treat the output of typical semantic segmentation module as semantic features, and have developed a framework that fuses these semantic features with low-level features. With the help of these semantic features, our segmentation algorithm is more likely to produce object-consistent regions. Experiments on two well known datasets have confirmed the effectiveness of our approach. We further showed that these semantic features can not only be fused with low-level features in a soft way; on their own, they can be utilized to generate hard tokens, which are then used to guide the interactive segmentation module. Some qualitative results have shown the promise of this approach.

- We follow this segment-then-recognize strategy, and find this philosophy works well for medical applications. In Chapter 5, we have developed a novel polar space random field model for detecting glandular structures in medical images. We treat the task of detecting glandular structures as object (gland) proposal, detection and segmentation problems respectively and show that our new technique outperforms state of the art computer vision algorithms in all these tasks.

- For the problem of image-level semantic image understanding, we have developed a novel usage of random forest to explore knowledge from the huge amount of tags widely present in the web currently. We treat the random forest *as a whole*, and introduce two new concepts: semantic nearest neighbors (SNN) and the semantic similarity measure (SSM). Based on which we performed the task of image retrieval and casted the task of image annotation as a learning to rank problem. Our new technique is intrinsically scalable and we will present experimental results in Chapter 6 to demonstrate that it is competitive to the state of the art methods.

Chapter 2

Literature and Methods

In this chapter, we give an overview of the related methods and the literature in the tasks related to semantic image understanding. We categorize these tasks into two categories: pixel-level semantic image understanding methods (including object detection and semantic segmentation) and image-level semantic image understanding methods (including image classification, image retrieval and image annotation). No matter if it is pixel-level or image-level semantic image understanding, they share at least two common modules: a feature extraction/-combination module and a classification module.

This chapter is structured as follows: we will first review related methods in feature extraction/combination and classification in section 2.1 and 2.2 respectively. In section 2.3, we review literature in object detection and semantic segmentation which we categorized as pixel-level semantic image understanding problems, and then we review literature in section 2.4 on image-level semantic understanding. Section 2.5 will give a short summary of this chapter.

2.1 Feature Extraction and Feature Combination

Feature is perhaps the most important concept in computer vision. The concept of feature is used to 'denote a piece of information which is relevant for solving the computational task related to a certain application.'[1] This definition implies that feature is a higher level concept than pixel which is specifically designed and represents our knowledge about a specific problem. With the help of feature, we no longer need to deal with the pixel, and a classifier could be applied directly to those features.

2.1.1 Feature Extraction

Numerous methods in the literature have been designed for extracting meaningful features. Based on their processing primitives, we categorize them into three groups: pixel-level features, regional features and image-level features.

On its own, a pixel should only have a color feature (R, G, B) and a geometric feature (x, y) which corresponds to its position in an image. However, a pixel can also be put into a larger spatial context. This is especially necessary for semantic segmentation, where a semantic label is required for every single pixel. For example, we can extract features from a patch which centers on this pixel, and consider these features as belonging to this pixel. In the prominent work of [9], the authors propose to extract the pixel features from an even larger area which also centers on this pixel. They randomly crop rectangles from this larger area and extract features from it. This feature together with the location of the rectangle are assembled as a two-tuple and considered to be the feature of the pixel. In this way, there could even be an infinite number of features for a pixel.

[1] http://en.wikipedia.org/wiki/Feature_(computer_vision)

For a region, features can be extracted corresponding to its color (color name [16], color SIFT [8], etc), texture (texton histogram [7]), shape (Histogram of Gradient (HoG) [6]), geometry information [17] or its appearance (Scale Invariant Feature Transform (SIFT) [5]). In [18], the authors proposed a kernel view of different regional features, and their proposed kernel descriptor can directly turn pixel attributes into compact regional features.

Although an image is composed of different regions, image feature extraction is not merely assembling regional features together. Concatenation is the least desirable method we would adopt in assembling regional features, as it will produce features of higher dimensions and worsen 'the curse of dimensionality'. Instead, the most popular method in the literature is the bag-of-words representation, which assigns each regional feature an index in a pre-trained codebook. These indexes, also called words, are then assembled together, and its histogram is considered as the image feature. Besides the bag-of-words representation [19, 20], there also exists other methods, such as the covariance matrix [21, 22] representation, fisher vector representation [23], graph representation [24], etc.

All the features described above are bottom-up features. Some recent works advocate using the outputs from classifiers as new and higher-level features. [25] considers the outputs of various object detectors as new features and successfully used them in the task of image classification; [26] extend this idea and use the outputs of many individual action detectors as new features which are then used for action recognition. In one of our works [27], we consider the outputs of typical semantic segmentation algorithm as high level features, and successfully integrated them with low-level features in hopes of generating object-consistent regions.

In summary, there is a crowd literature on image feature extraction, and many new kinds of features are still emerging. Just as we mentioned at the beginning of this section, a good feature should be problem dependant and the whole

purpose of feature extraction is to facilitate solving the specific problem. If an idealized feature could be extracted, then this specific problem is almost solved. For example, if our task is to classify apple from pear, then the shape feature will give us a good result. As most apples are round, while pears are not. However, many problems can't be easily solved by a single type of feature, and we had to combine different types of features together.

2.1.2 Feature Combination

The importance of feature combination has long been recognized by the computer vision community. Different feature types, such as local, global, color, texture, etc, capture different characteristics of an image. It is often helpful and sometimes necessary to combine various features together in order to gain a comprehensive understanding of an image.

For a typical semantic image understanding system, what follows a feature extraction step is a classification step. If we consider a classifier as a one-input-one-output black box, then the feature combination can happen both on the input and the output level. On the input level, we can simply concatenate different features together. In [28], the authors found that they can obtain state of the art semantic segmentation results by simply concatenating local regional features with global bag-of-words features; on the output level, we can fuse the outputs of different classifiers together. There exists research [29] showing that summing the classifiers' probabilistic outputs is always a robust way of combining classifiers. Previous works have always shown a performance gain when combination is used.

Besides combination on the input and the output level, we can utilize a kernel to represent the relations between samples in different feature channels, and then

perform the combination at the kernel level, which can be considered as a middle level fusion stage.

Let $\{\mathbf{x}_i\}_{i=1}^{N}$ be N instances, and $\{f_m\}_{m=1}^{F}$ represent a set of F feature extractors. K_m is a kernel function which will be performed on the *m*-th feature channel. Then the similarity between two instances based on their *m*-th feature f_m is defined as $K_m(f_m(\mathbf{x}_i), f_m(\mathbf{x}_j))$. Kernel based feature combination is about combining different K_m into a single kernel K^*. Possible methods include linear combining them:

$$K^*(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{F} \beta_m K_m(f_m(\mathbf{x}_i), f_m(\mathbf{x}_j)) \tag{2.1}$$

where β_m are the linear combination coefficients, which can either be fixed or to be learned by the subsequent classification algorithm.

2.2 Classification Methods

A classifier, which is a concept borrowed from machine learning community, is another basic module shared by different semantic image understanding tasks. The problem of semantic image understanding is in fact a pattern recognition problem, which is defined as the task of assigning a label to a given input value[2]. Most of the methods adopted in the literature transformed the recognition problem to a classification problem, although these two concepts are not exactly the same.

For example, if we see an object that we have never seen before, we would instantly know that we don't know this object. This is an ideal case of recognition. But a classification based system will compare this object with all previously

[2]http://en.wikipedia.org/wiki/Pattern_recognition

learned classes. Only when all of these classes output negative values, can the system get to the conclusion that it doesn't know about this object.

The drawback of the classification paradigm is obvious: it has to undergo a laborious one-to-one comparison step, and can only recognize the classes that have been previously met. Although there is some research [30] trying not to classify the data, they are seldom used in the literature partly because of their relatively low performance.

In the following, we will review several classification techniques that are commonly used in the literature.

2.2.1 From SVM to MKL

Support Vector Machine (SVM) is perhaps the most widely used machine learning tool in computer vision research. Its popularity is mostly because of its many attractive properties, like its sound theoretical justification, a simple geometric interpretation, a sparse solution, less prone to overfitting, etc.

Linear SVM aims to learn a hyperplane that can separate the training data $\{\mathbf{x}_i\}_{i=1}^N$ based on their corresponding labels $\{y_i\}_{i=1}^N$, while simultaneously trying to maximize the margin between different classes. By adopting the kernel trick, SVM can handle non-linear data. A kernel $K(\mathbf{x}_1, \mathbf{x}_2)$ is formally defined as the inner product of $\phi(\mathbf{x}_1)$ and $\phi(\mathbf{x}_2)$:

$$K(\mathbf{x}_1, \mathbf{x}_2) = \langle \phi(\mathbf{x}_1), \phi(\mathbf{x}_2) \rangle \tag{2.2}$$

where $\phi(\bullet)$ is a mapping function that projects the original data $\mathbf{x}$ into a higher, or even infinite dimensional space $\phi(\mathbf{x})$. For the multi-class problem, we can project $\mathbf{x}$ together with its label information y into a joint high dimensional

space $\phi(\mathbf{x}, y)$. A simple way of defining $\phi(\mathbf{x}, y)$ could be [31]:

$$\phi(\mathbf{x}, y) = [\mathbf{0}, \cdots, \mathbf{0}, \underbrace{\phi(\mathbf{x})}_{y}, \mathbf{0}, \cdots, \mathbf{0}] \tag{2.3}$$

For example, if we have three classes, then $y \in \{1, 2, 3\}$. Equation (2.3) means $\phi(\mathbf{x}, y = 1) = [\phi(\mathbf{x}), \mathbf{0}, \mathbf{0}]$, $\phi(\mathbf{x}, y = 2) = [\mathbf{0}, \phi(\mathbf{x}), \mathbf{0}]$ and $\phi(\mathbf{x}, y = 3) = [\mathbf{0}, \mathbf{0}, \phi(\mathbf{x})]$.

Therefore, a standard multi-class kernel SVM can be defined as:

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{b},\xi} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{N} \xi_i \\ s.t. \quad &(\mathbf{w} \cdot \phi(\mathbf{x}_i, y_i) + b_{y_i}) - (\mathbf{w} \cdot \phi(\mathbf{x}_i, y) + b_y) \geq 1 - \xi_i, \quad \forall i, y \neq y_i \end{aligned} \tag{2.4}$$

where $\mathbf{b}$ is a vector composed of $\{b_y, y \in \mathcal{Y}\}$.

The predicted class y for a new test sample $\mathbf{x}$ is:

$$y = \arg\max_{y \in \mathcal{Y}} \ \mathbf{w} \cdot \phi(\mathbf{x}, y) + b_y = \arg\max_{y \in \mathcal{Y}} \ \sum_{i=1}^{N} \alpha_{iy} K(\mathbf{x}_i, \mathbf{x}) + b_y \tag{2.5}$$

where α_{iy} is the Lagrange multiplier.

The original SVM only utilizes one kernel matrix, which is sometimes deemed to be inadequate. On one hand, there are many different kinds of kernel functions, and each kernel function has parameters which are usually difficult to tune. On the other hand, there are scenarios where there is a need to utilize different kernels, such as the feature combination scenario introduced above. Therefore, instead of utilizing only one kernel, Multiple Kernel Learning (MKL) tries to utilize a series of kernels and learn an optimal linear combination of them.

Suppose we have F kernels $\{K_1, K_2, ..., K_F\}$, which are obtained either by varying the parameters of the kernel function or performed on different feature

channels, then we replace the single kernel K of SVM appeared in (2.5) with a linear combined kernel $K' = \sum_{m=1}^{F} \beta_m K_m$. Thus (2.5) can be re-written as:

$$y = \arg\max_{y \in \mathcal{Y}} \sum_{m=1}^{F} \beta_m \mathbf{w}_m \cdot \phi_m(\mathbf{x}, y) + b_y = \arg\max_{y \in \mathcal{Y}} \sum_{i=1}^{N} \alpha_{iy} \sum_{m=1}^{F} \beta_m K_m(\mathbf{x}_i, \mathbf{x}) + b_y \tag{2.6}$$

Thus the multi-class MKL can be formulated as:

$$\begin{gathered} \min_{\mathbf{w},\mathbf{b},\beta,\xi} \quad \frac{1}{2} \sum_{m=1}^{F} \beta_m \|\mathbf{w_m}\|^2 + \sum_{i=1}^{n} \xi_i \\ s.t. \quad \forall i, y \neq y_i, \\ \left(\sum_{m=1}^{F} \beta_m \mathbf{w}_m \cdot \phi_m(\mathbf{x}_i, y_i) + b_{y_i} \right) - \left(\sum_{m=1}^{F} \beta_m \mathbf{w}_m \cdot \phi_m(\mathbf{x_i}, y) + b_y \right) \geq 1 - \xi_i \end{gathered} \tag{2.7}$$

The seminal work of MKL dates back to [32]. After MKL was proposed, many variants of it have been proposed [31, 33, 34], and have been quickly adopted to deal with various computer vision problems [4, 35]. [36] provided a good survey on various kinds of MKL learning algorithms. They empirically compared different MKL algorithms and found no large differences between them in terms of accuracy.

Despite its huge success, the formulation of MKL is still being questioned by researchers. In essence, MKL is simply a linear combination of different kernels. It implies that the contribution of each kernel is fixed for all the training samples [37]. This seems to be an unnecessary, too strong constraint. In [37], the authors propose to learn augmented coefficients for each sample in each feature channel. They achieve this by augmenting the kernel matrixes. However, as the augmented kernel they used is still a block diagonal matrix, the coefficients they learned are equivalent to learning different kernels separately and adding an appropriate bias term for all the kernel classifiers. In [38], the authors proposed

several interesting heuristics and schemes for combining kernels in a nonlinear and data dependent way, including exponential weighting scheme ('ExpWS'), 'MaxMin' method, etc.

Although SVM (including MKL) has achieved tremendous success in the last decade, it still has some limitations. For the linear SVM, it's mostly limited by its linear nature, which makes it unable to deal with non-linear data. Therefore, kernel SVM is more preferred in common practice, and it is generally believed that kernel SVM can obtain a higher accuracy than linear SVM. However, kernel based methods (including kernel SVM, MKL, etc) can not easily scale to large-scale settings, as we had to store the kernel matrix whose size is as large as N^2, where N is the number of training samples. Therefore, we had to resort to linear SVM in large-scale scenarios.

2.2.2 Peril and Promise for Nearest Neighbor method

SVM is in fact a parametric model [39] which is characterized by its parameters. In contrast, there also exist non-parametric models which are parameter-free. Among them, Nearest Neighbor method is perhaps the most prominent one. In a typical K-Nearest Neighbor (KNN) method, one only needs to store all the training samples together with their labels. When a test sample comes, its distance from all the stored training samples are calculated, and the label that appears most in its K nearest neighbors will be assigned to the test sample, where K could be any integer.

In the past, NN is always considered as a baseline method, and it is generally believed that its performance cannot compete with discriminative classifiers, such as SVM [40]. This belief is however, challenged by the work of [39], where the authors showed that NN based methods can beat the performance of SVM in image classification tasks. This idea is then further explored in [41–43]. Besides

classification, NN based methods have also shown their effectiveness in other computer vision tasks, including scene completion [44], image parsing [17, 45], image annotation [1, 46], etc.

Compared to SVM, NN clearly enjoys some advantages. For example, NN does not need a complicated and usually time-consuming training procedure which is required by SVM. This is especially important when more training samples become available: SVM needs to be re-trained based on the enlarged training set, whilst NN only needs to store those new coming samples. Besides, NN can easily deal with large number of classes, which however poses a big challenge for SVM. In fact, we believe that the style of NN classification resembles more to human recognition, as we do not necessarily need to transform the recognition problem into many classification problems.

From the machine learning point of view, we are always interested in the generalization power of an algorithm. For SVM, it utilizes a set of training samples, i.e. support vectors, in hopes that all these training samples can together generalize well; while for NN, it only utilizes the K nearest neighbors. Although the idea of utilizing a set of samples to achieve a good generalization power seems more appealing than only using a few nearest neighbors, there exists recent work [47] showing that even based on one sample, the algorithm can also generalize well.

On the other hand, the drawback of NN is also obvious. One of the biggest concerns is the time complexity when making predictions. Suppose there are n training samples in $\mathbb{R}^m$, then it requires $O(nm)$ time to predict one test sample. This is in contrast with $O(m)$ for linear SVM or $O(cm)$ for kernel SVM, where c is the number of support vectors which is usually much smaller than n. Although there exist efficient data structures which can accelerate the searching speed, like kd-trees, it is only useful for low-dimensional data [48]. There also exist scenarios where there is no need to return the actual NN, and a 'good guess' is

enough. In those cases, we can adopt Approximate Nearest Neighbor (ANN) methods, like Locality-Sensitive Hashing (LSH), randomized kd-trees [49], etc.

Besides the high prediction complexity, another problem facing NN in image understanding area is the semantic gap problem, which states that the nearest neighbors retrieved by visual similarity doesn't necessarily guarantee semantic similarity. After all, NN is still an unsupervised method.

2.2.3 How can Random forest help?

Random Forest, also known as decision forest or decision trees, is becoming more and more popular in recent years. A random forest if formally consisted of a set of random trees, where each tree is trained on a random subset of the training data, thus each tree can be considered as independent from each other. The outputs from each tree are then combined together as the final output of the random forest.

Consider a typical classification scenario, where we try to build a decision tree for the classification purpose. Suppose we have a set of training samples $\{\mathbf{x}_i\}_{i=1}^N$ and their corresponding class labels $\{y_i\}_{i=1}^N$, we first extract features $\{\mathbf{F}_i\}_{i=1}^N$ from these samples, and then define a split function in order to split the samples into two subsets, which we named as left child and right child. Possible split functions include a linear classifier [50, 51]:

$$\begin{cases} \mathbf{w}^\mathbf{T}\mathbf{F} + b \geq 0 & \text{go to left child} \\ \text{otherwise}, & \text{go to right child} \end{cases} \tag{2.8}$$

or the feature difference between two feature dimensions [52]:

$$\begin{cases} F_i - F_j \geq \text{thresh} & \text{go to left child} \\ \text{otherwise}, & \text{go to right child} \end{cases} \tag{2.9}$$

Based on the split function defined above, we can split samples to the left child node or right child node accordingly. We can generate multiple splits by choosing different feature dimensions or different thresholds. Then we can use the widely used information gain criteria [10, 50] to pick the best split:

$$Score(split) = \triangle E = -\frac{|I_l|}{|I_n|}E(I_l) - \frac{|I_r|}{|I_n|}E(I_r) \tag{2.10}$$

where $E(I)$ is the Shannon entropy of the class distribution in the set of samples I. $|I|$ means the number of samples contained in I. I_n is the set of training sample in node n, while I_l and I_r represent the training images contained in node n's left and right child node respectively.

This kind of split recursively performs on the training samples until the stopping criterion is satisfied. The stopping criteria could be set as the maximum depth of the tree or the least number of samples contained in a node. At each leaf node, we store the posterior probability $p(\mathbf{y}|\mathbf{x})$ as the output of this tree, and the outputs from different decision trees are combined together as the final output of the random forest.

When a test sample comes, it drops from the root node and keeps falling according to the split functions stored at each node until it reaches a leaf node. Then the posterior probability stored at that leaf node will be used to classify this sample.

Past research on random forest usually focuses on its discriminative power, and it has been successfully used in many different computer vision tasks, including image classification [50], object detection [53], human pose estimation [54], etc. In this thesis, we would like to emphasize its tree structure. We believe that the tree structure inherited in the random forest can not only facilitate the decision making process of the random forest, but also serve as an ANN (Approximate Nearest Neighbor) search method. If we consider random forest as an ANN

method, then it can perfectly overcome the two inherited problems described above in NN based methods: firstly, the tree structure inherited in the random forest can ease the computational burden of NN in making predictions; secondly, random forest can easily incorporate supervised information in generating the random trees, which can tackle the semantic gap problem.

2.2.4 Integrating Classifiers in a Random Field Model

Sometimes, we not only need to tackle the single-input single-output classification problem, but also a single-input structure-output problem. This is generally known as a structure learning [55, 56] problem. For example, when we need to assign each pixel of an image a semantic label, we not only need to perform the classification on each single pixel, but also need to combine the classification results together to obtain a globally consistent result.

Conditional Random Field model (CRF) [57] is an elegant method to deal with this problem. A CRF formally consists of a random variable $\mathbf{X}$ over the observed data and a set of random variables $\mathbf{Y} = \{Y_1, Y_2, ..., Y_n\}$ over the labels to be inferred. All components Y_i of $\mathbf{Y}$ are usually assumed to range over a finite label alphabet $\mathcal{Y}$. When conditioned on $\mathbf{X}$, if $\mathbf{Y}$ obeys the Markov property: $p(Y_v \mid \mathbf{X}, Y_w, w \neq v) = p(Y_v \mid \mathbf{X}, Y_w, w \sim v)$, where $w \sim v$ means there exists a link between w and v, then $(\mathbf{X}, \mathbf{Y})$ is a CRF.

A *clique* C in a CRF is defined as a fully connected subset of the vertices Y_i. Then the conditional probability of label $\mathbf{Y}$ given $\mathbf{X}$ is defined as:

$$p(\mathbf{Y} \mid \mathbf{X}) \propto \exp(-E) = \exp(-(\sum_{C} \psi_C(Y_C \mid \mathbf{X}))) \tag{2.11}$$

where E is the energy function of a CRF, and ψ_C is the so-called potential function which reflects the cost of the current label configuration Y_C. When the clique C contains only a single Y_i, then ψ_C is named as the unary potential;

when it contains a pair (Y_i, Y_j), then ψ_C corresponds to the pairwise potential. C could also contains more than two variables, and those ψ_C are named as high order potentials [14].

The inference procedure of a CRF corresponds to finding an optimal $\mathbf{Y}$ which can achieve the lowest E or the highest $p(\mathbf{Y} \mid \mathbf{X})$:

$$\arg\max_{\mathbf{Y}\in\mathcal{Y}} p(\mathbf{Y} \mid \mathbf{X}) = \arg\min_{\mathbf{Y}\in\mathcal{Y}} E = \arg\min_{\mathbf{Y}\in\mathcal{Y}} \sum_{C} \psi_C(Y_C \mid \mathbf{X}) \tag{2.12}$$

whilst the training procedure of a CRF is to find appropriate parameters in ψ_C in order that the optimal $\mathbf{Y}$ for the training data corresponds to the ground truth.

CRF is an elegant and principled framework for combining different classifiers, in which we can define suitable forms of ψ_C to tackle the problem at hand. For example, the unary potential can be defined as the output of a classifier, either SVM or random forest, which is specifically trained to predict the label of X_i; the pairwise potential can be defined to reflect the co-occurrence statistics obtained from the training set. For example, 'water' and 'boat' are more likely to co-occur than 'water' and 'bus'. Thus we can assign a higher cost for assigning neighboring nodes as 'water' and 'bus'. We don't need to specify these rules manually, as these rules can be learned by the CRF automatically in its training phase.

After reviewing the methods in feature extraction/combination and classification domain, we now introduce the literature in the tasks related to semantic image understanding. We will first review related methods on pixel-level semantic image understanding and then on the image-level.

2.3 Semantic Image Understanding at the Pixel Level

The most related two tasks in the field of pixel-level semantic image understanding are object detection and semantic segmentation. We now review related literature in these two tasks.

2.3.1 Object Detection

Generally speaking, objects can be divided into two kinds [58]: rigid objects with specific shapes (such as pedestrians, cars, etc) and objects of amorphous spatial extent (e.g. trees, road, sky). However, most existing methods are designed to detect rigid objects, as it is still very difficult to detect non-rigid ones. We methodologically categorize existing object detection techniques into three categories: *treating object as a whole*, *treating objects as a constellation of parts*, and *context based methods*.

Treating Object as a Whole

The most straightforward approach in *treating object as a whole* is the sliding window method. For a test image, a predefined sub-window slides over the image, trying to cover all possible locations and scales. The features in this sub-window are extracted and fed to a classifier which is trained to decide if the sub-window contains the specific object or not. Although this idea is simple, it's still the dominating approach in the literature. Recent modifications to this approach are methods which try to increase the detection speed using branch and bound [59], or combine the holistic window with some inner parts of the window which represent the parts of the object [60, 61].

Treating Object as a Constellation of Parts

The major problem of treating objects as a whole is that it can hardly deal with the problem of occlusion. However, in our visual world, objects are often occluded by other objects.

Part-based methods are inherently more robust than holistic methods in dealing with occlusion. Existing successful methods include the constellation model [62, 63] and Hough voting based methods [64]. The constellation model treats the object as a constellation of local parts, and infers their 'optimal' combination in a Bayesian framework. Here the 'optimal' means lower intra class distance and the higher inter-class distance. Hough voting is an old idea in detecting lines or some regular shapes. This idea was then generalized and successfully used in object detection scenario by the Implicit Shape Model (ISM) [64]. In ISM, each part votes for the most probable places of the object center. The evidence from different parts are accumulated and the peak of the accumulation map is chosen as the object center.

Although the strategy of treating object as a constellation of parts is more tolerant to occlusion, its drawback is a lack of discriminative power. As the features extracted from parts (usually in the form of a patch or a small region) do not necessarily contain enough discriminative information. In these cases, we have to extract features from a larger context than the part itself.

Context-based Object Detection

It is generally believed that context plays a very important role in human object recognition. Taking the image in Fig.2.1 as an example, we can easily judge that there exists a car in the left image and a person in the right image. However, it turns out that the blob on the right is identical to the one on the left after a 90 degree rotation!

Many works exist on utilizing context to detect objects. To name a few, [65] uses the image categorization result as a prior to guide the object detection. [66]

FIGURE 2.1: A 'car' on the left and a 'pedestrian' on the right. In courtesy from http://web.mit.edu/torralba/www/carsAndFacesInContext.html

uses the entropy criterion to select the 'unknown' objects. For those 'unknown' classes, the author concatenates the features of this unknown object with the features extracted in neighboring known classes, and uses the concatenated features to re-detect the unknowns.

In summary, there are numerous methods exist for object detection, but still it is an unresolved problem. Although the detection performance is progressing year by year by benefiting from the increase of computational power and the carefully designed machine learning techniques, the best state of the art pedestrian detector in real-world scenario is still far below our requirements [67]. Besides, object detection can only be used to detect a specific class of objects in an image, it cannot obtain the semantic meaning of every single pixel, and this is exactly what the semantic segmentation tries to do.

2.3.2 Semantic Segmentation

Semantic segmentation aims to assign a semantic label to every single pixel in an image. A common aspect shared by most of the existing algorithms is that they usually choose pixel or superpixel [9, 12, 68] as their processing primitives. For each pixel or superpixel, some features as described in section 2.1.1 are extracted first. Then a classifier is performed on these extracted features to predict the semantic class that this pixel might belong to. To ensure that we will obtain

a globally consistent result, the classifier outputs of different pixels are then assembled together in a CRF framework.

One notable exception beyond this theme is the work in [69], where the authors directly chose the region as their processing primitives, and they achieved the best segmentation accuracy on the recent PASCAL challenge [70]. Compared with choosing pixel or superpixel as processing primitives, choosing regions enjoys some benefits: the information contained in a region is more comprehensive than that contained in a superpixel, and the shape of the region is beneficial for recognizing some shape dominate objects. However, one obstacle in adopting this strategy is that it is very difficult for the segmentation algorithm to produce semantic consistent regions. [69] circumvents this problem by generating multiple figure-ground hypotheses. However, the segmentation module they adopted is still low-level cue based. Although they performed a supervised ranking after the hypothesis generation, the errors that occurred in the low-level segmentation module will not be remedied.

In fact, the segmentation problem is an ill-posed problem. It is generally believed that we need to utilize top-down information to guide the bottom-up segmentation. In other words, if we want to achieve a semantic consistent segmentation, we need to incorporate information beyond the image itself. While the authors in [71] used a shape prior to guide the segmentation, classcut [72] or co-segmentation [73] utilize information from other images. Another interesting work in [74] retrieves similar images from the internet, and these retrieved images are treated as additional information to guide the segmentation.

2.4 Semantic Image Understanding at the Image Level

Although the idea of understanding the semantic meaning of each pixel seems appealing, it is perhaps too difficult at the current level of technology. For example, the most state of the art results on a challenging dataset can only obtain an accuracy of around 43% [75]. Therefore, if we take a step back, and consider the image-level semantic image understanding, this task perhaps becomes easier, while it is still of practical significance. On one hand, there exist scenarios where we only need image-level semantic meanings, such as the content based image retrieval (CBIR) system; on the other hand, an accurate image-level semantic understanding can serve as the context information, which can in turn boost the performance of pixel-level understanding [65].

There are at least three tasks that are related to image-level semantic image understanding: image classification, image retrieval and image annotation.

Image classification aims to classify an image whether it contains a specific object or not. Among all the image classification methods, Bag-of-Words (BoW) model is the most well-known. It dates back to the seminal paper of [20], although similar ideas can be found even earlier [19]. Ever since BoW was proposed, a lot of variants and improvements have been made upon the original one [76–78]. We refer the reader to the recent PASCAL VOC challenge[3] for the state of the art methods in image classification.

Given a query, either a keyword or an image, the task of image retrieval is to return a ranked list of images from a large image repository that are mostly related to the query. For a good survey, please refer to [79].

[3] http://pascallin.ecs.soton.ac.uk/challenges/VOC/

Different from image classification or image retrieval, the task of image annotation is to automatically assign metadata, in the form of caption or keywords, to a digital image. Large quantity of literature exists in the image annotation area, and methods range from generative models [80–82] to discriminative models [83]. While generative models need a large quantity of training data to learn the joint probability of semantic concepts and image visual features, discriminative models treat each tag as a semantic class, and try to learn a different classifier for each tag. Therefore, it is not easy to model the correlations between tags and this discriminative model will inevitably encounter difficulties when dealing with large number of tags. Recently, nearest neighbor based methods [1, 46] have attracted much attention. Among them, Tagprop [46] is perhaps the most successful method which shows superior performance on several benchmark image annotation datasets. It uses a weighted nearest neighbor model to predict the possible tags. Its superior performance relies on its sophisticated training procedure, and its optimization function is composed of items corresponding to every tag in every image, which will inevitably hinder its applicability to large scale datasets.

In fact, these three tasks are closely related. If the image annotation algorithm can automatically assign an image with keywords, then the image retrieval system can retrieve images directly based on these keywords. The problem of image annotation itself can be decomposed into a set of image classification tasks, where each classification task aims to predict the existence of a particular tag.

2.5 Summary

In this chapter, we have reviewed related methods and the literature in tasks related to semantic image understanding. Semantic Image Understanding is in

fact a very broad area, and it's impossible to review all related methods here. We will also review the most related methods to ours in the later chapters where necessary.

Chapter 3

Feature Combination beyond Basic Arithmetics

Just as we mentioned in the previous chapter, feature plays a fundamental role in semantic image understanding, and how to efficiently combine different features is a non-trivial task. In this chapter, we introduce our new method which can consistently boost the performance of Multiple Kernel Learning, which is deemed to be the most prominent method in kernel-based feature combination. Furthermore, we will also show that our proposed method also works in unsupervised scenarios, where there is a need to sum different similarities together.

This chapter is organized as follows: in section 3.1, we review some basics for kernel based feature combination and introduce the rationale behind our new method. Then we introduce the concept of kernel histogram in section 3.2, and present our histogram matching based kernel combination technique in section 3.3. As the histogram matched kernels may be no longer positive semi-definite, they need to be transformed to positive semi-definite in section 3.4. We also propose to use two kinds of measures adapted from traditional MKL solvers for automatically choosing the canonical kernel histogram in section 3.5. Extensive

experiments on various computer vision and machine learning datasets are done in section 3.6 and some concluding remarks are given in section 3.7.

3.1 Introduction

Let $(x_i, y_i), i = 1, 2, ..., N$ be *N* instances consisting of samples $x_i \in X$ and class labels $y_i \in \{1, 2, ..., C\}$; $f_m \in \mathbb{R}^{d_m}, m = 1, 2, ..., F$, represent a given set of features, where d_m denotes the dimensionality of the *m*-th feature. Feature combination is to use all these F features together to learn a classifier to classify X into Y. Kernel methods make use of kernel functions to define a measure of similarity between pairs of instances. Let K be a kernel function, the similarity between two instances based on their *m*-th feature, f_m, is defined as:

$$K_m(x_i, x_j) = K(f_m(x_i), f_m(x_j)) \tag{3.1}$$

Kernel based feature combination is about combining different K_m into a single kernel K^* and can be done with various arithmetical operations [84] including baseline average (3.2) and MKL (3.3).

The baseline average kernel:

$$K^*(x_i, x_j) = \frac{1}{F} \sum_{m=1}^{F} K_m(x_i, x_j) \tag{3.2}$$

In the case of MKL, the combined kernel K^* is a linear combination of different kernels weighted by a set of adaptive parameters $\{\beta_m\}$ to be learned by the MKL algorithms.

$$K^*(x_i, x_j) = \sum_{m=1}^{F} \beta_m K_m(x_i, x_j) \tag{3.3}$$

In essence, almost all previous works on Multiple Kernel Learning (MKL) tried to learn a linear combination of different kernels, and each individual kernel are considered *as is*. In this chapter, we make an important observation of the distribution of different kernels that are routinely used in the literature. We discovered that the histograms of the kernel values of different features are usually quite different from each other. Some histograms may be narrow and occupy only a short range, while others may span a wide range; some histograms may look like a gaussian distribution, while others may look like an exponential distribution. As these histograms differ so much, it means that their units of measure are not the same. In other words, for the same similarity/difference value, it may represent a 'huge' difference in one feature channel, but only a 'tiny' difference in the other channel. Therefore, it is necessary to standardize each feature channel before they are combined together.

3.2 Kernel Histogram and Histogram Matching

Before delving into our method on standardizing the kernel, we first introduce the concept of **Kernel Histogram** and **Histogram Matching**.

Given a kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, suppose all its elements lie in the range of (K_{min}, K_{max}), then we equally divide this value range into M discrete bins. Here M is an integer, and the width of each bin is $(K_{max} - K_{min})/M$. The **histogram** $H = \{h_v, v \in (1, 2, ..., M)\}$ of this kernel matrix is a vector storing the number of kernel elements that lie in each bin. It is formally defined as:

$$\begin{aligned} h_v = \sum_{i=1}^{n}\sum_{j=1}^{n} \delta\left(k_{i,j} > K_{min} + (v-1) * \frac{(K_{max} - K_{min})}{M}\right) \\ * \delta\left(k_{i,j} < K_{min} + v * \frac{(K_{max} - K_{min})}{M}\right) \end{aligned} \tag{3.4}$$

$\delta(\bullet) = 1$ if the condition in the bracket holds and 0 otherwise. $k_{i,j}$ is the kernel element which appears in the ith row and jth column of the kernel matrix $\mathbf{K}$. This histogram can then be transformed to have probabilistic meaning by dividing a normalizing constant which is $n \times n$ in this case. In essence, the histogram is a general property of a kernel matrix that depicts its value distribution, and each kernel has its own unique histogram.

The concept of histogram is also commonly used in image processing. An image can be considered as a two dimensional matrix, and it also has its own histogram. **Histogram matching** is a well known technique used in image processing. It can adjust the histogram of an image to any shape while still maintaining the order. Here the order means if pixel A is brighter than pixel B before the adjustment, then it is still brighter after the adjustment. This technique can be adapted to perform on the kernel histogram as well. We will develop a piecewise linear histogram matching algorithm in Algorithm.1 to achieve this. It differs from typical histogram matching methods[1] in that the order of kernel elements that fall onto the same bin are also preserved after the histogram matching.

3.3 Standardizing Kernel Values through Histogram Matching

An inspection of the histograms of $K_m(x, x')$ for different features (see Fig.3.4) shows that they are very different for different features. Linear combination of the kernels as in (3.2) and (3.3) can be seen as combining 'things' measured with different units directly without converting them to the same standard. We argue that before they are linearly combined, the kernel values should be calibrated to exhibit the same distribution.

[1]For a brief introduction on histogram matching, please refer to http://paulbourke.net/texture_colour/equalisation/

Intuitively, the similarity distributions amongst the data points for a given dataset should not change with their representation features. As kernels measure the similarities between samples, we call this intuition the relative kernel distribution invariance (RKDI) property. In the following, we will try to make this intuition concrete.

Defining the inverse cumulative density function (ICDF) of kernel *m* as:

$$ICDF_m(u) = \inf_{v \in \Re} \left(\int_{-\infty}^{v} p_m \left(K_m(x, x') = w \right) \, dw \geq u \right) \tag{3.5}$$

where $p_m \left(K_m(x, x') \right)$ is the probability density function of the *m*-th feature channel, then RKDI can be defined as:

$$\int_{-\infty}^{ICDF_m(u)} p_m \left(K_m(x, x') = w \right) \, dw = \int_{-\infty}^{ICDF_n(u)} p_n \left(K_n(x, x') = w \right) \, dw \; \forall m, n, u \tag{3.6}$$

where $p_n(K_n(x, x'))$ and $ICDF_n(u)$ represent the probability density function and the inverse cumulative density function respectively.

Clearly, (3.6) states that the percentiles of the relative similarities of the given data should be the same in any feature space and should be calibrated to be the same. Although there is no formal proof known to us at this stage, we believe it is a reasonable assumption and will show experimentally that maintaining such invariance can help improve performance.

The problem of (3.6) is the well-known histogram matching problem and our new feature combination framework is illustrated in Fig.3.1. Let $HM \left(K_m(x, x') \right)$ represent the **H**istogram **M**atching operator being performed on the *m*-th kernel, then average and MKL are represented as follows.

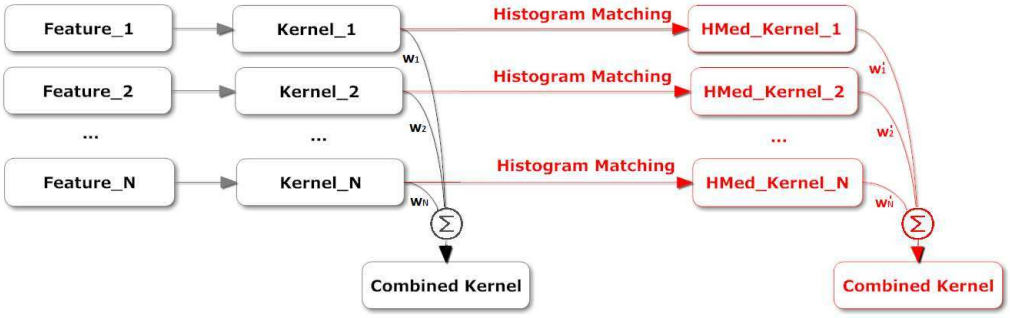

FIGURE 3.1: Typical feature combination methods always represent features into their kernel forms. These kernels are then combined. Traditionally, the kernels are combined directly through one of the methods in (3.2) or (3.3). In this chapter, we proposed to add a histogram matching module before these kernels are combined by one of the methods in (3.2) or (3.3).

The new average K^* kernel is formed as:

$$K^*(x, x') = \frac{1}{F} \sum_{m=1}^{F} HM(K_m(x, x')) \tag{3.7}$$

In the case of MKL, the combined kernel K^* is formed as:

$$K^*(x, x') = \sum_{m=1}^{F} \beta_m HM(K_m(x, x')) \tag{3.8}$$

Our histogram matching algorithm is summarized in **Algorithm 1**. It differs from typical histogram matching methods in that the elements in the kernel matrixes are continuous instead of discrete values. Therefore, we need to quantize the kernel values into discrete bins. To reduce the quantization error and maintain the original order, the values are piecewise linear interpolated for each bin. Note that in all our experiments, we use 1500 bins.

Algorithm 1 Piecewise Linear Histogram Matching

Input: template (canonical kernel), orig_kernel, num_of_bins;
Output: HMed_kernel (**H**istogram **M**atched kernel)
Normalize template and orig_kernel to (0,1);
sorted_template = sort(template);
for i=1 to num_of_bins **do**
  cut_point_index = size(find(template < i/num_of_bins));
  cut_point_value = sorted_template[cut_point_index];
end for
for i in orig_kernel **do**
  lower_bound = max(orig_kernel[i] > cut_point_value(:));
  upper_bound = min(orig_kernel[i] < cut_point_value(:));
  HMed_kernel[i] = Linear_interpolate(lower_bound,orig_kernel[i],upper_bound);
end for
Normalize HMed_kernel back to the original range

3.4 Transforming Kernels to Positive Semi-Definite

Note here we should ensure the histogram matched kernels be positive semi-definite. Although there are some previous works [85] showing that even if the kernel matrix is not positive definite, it is still appropriate for SVM based classification. In fact, there exists some research on SVM classifier with non-positive definite matrixes [86, 87], but the research is still very limited [38]. Therefore, we propose a method to transform the indefinite kernel to a positive definite kernel to make it universally applicable for conventional MKL solvers.

In [38], two methods for transforming indefinite matrices to positive semi-definite matrices are proposed, one is to use the second power of the kernel matrix to replace the original matrix, and the other is to use the spectral decomposition and ignore the spectra with negative eigenvalues. The same method is also adopted in [88], where the negative eigenvalues are treated as noise, and are replaced with zero. In our case, if we adopt the first method, we will see that the histogram of the kernel matrix after the multiplication will also change. Therefore, it is not applicable in our scenario. Hence we prefer to use the second method.

Consider the spectral decomposition of the kernel on the training set:

$$K_{train} = VDV^T \tag{3.9}$$

where D is a diagonal matrix contains all the eigenvalues and V is an orthogonal matrix, whose column contains the corresponding eigenvectors. Suppose D has some negative values, then we replace these negative values with 0 and obtain a new D which we named as D^*. Denote V^* as the eigenvectors corresponding to the non-negative eigenvalues. Then we can obtain a new positive semi-definite matrix K^*_{train} which is computed as:

$$K^*_{train} = VD^*V^T = V^*D^*V^{*T} \tag{3.10}$$

It will replace the original indefinite kernel K_{train}. In essence, the above procedure performs exactly the same as Principal Component Analysis (PCA), as we only retain the positive eigenvalues. Notice here that as the kernel on the training set has changed, we also need to change the corresponding kernel on the test set. If the training kernel is full rank, then its eigenvectors will be a complete basis, which means the equation $VV^T = I$ holds, where I is the identity matrix. Hence we will have:

$$K_{test} = K_{test}VV^T \tag{3.11}$$

However, as the original K_{train} is not positive definite, the eigenvector V will not form a complete basis. As we have already projected the original K_{train} to the space spanned by the non-negative eigenvectors, we also need to do this projection for K_{test}. Recall that V^* represents the eigenvectors corresponding to the non-negative eigenvalues, then the projected test kernel K^*_{test} can be computed as:

$$K^*_{test} = K_{test}V^*V^{*T} \tag{3.12}$$

3.5 Automatically Choosing the Canonical Kernel Histogram

An important question in this method is finding the canonical kernel histogram which is likely to be dataset dependant. Using a cross validation technique is not a principled method. In this section, we propose two measures that are suitable for selecting the canonical kernel histogram in a systematic and principled manner. The first one is to use the kernel alignment score proposed in [89]. Consider a two-class labeled dataset $S = (x_i, y_i)_{i=1}^{l}$ with $y_i \in \{+1, -1\}$, then the kernel alignment score between a kernel K_p and the kernel K_q over the sample S is defined as:

$$\hat{A}(S, K_p, K_q) = \frac{\langle K_p, K_q \rangle_F}{\sqrt{\langle K_p, K_p \rangle_F \langle K_q, K_q \rangle_F}} \tag{3.13}$$

where $\langle K_p, K_q \rangle_F = \sum_{i,j=1}^{l} K_p(x_i, x_j) K_q(x_i, x_j)$. This alignment score can be considered as the cosine of the angle between two bi-dimensional vectors K_p and K_q. If we consider an ideal kernel $K_q = \mathbf{y}\mathbf{y}^T$, where $\mathbf{y} = [y_1, \ \dots, y_l]^T$, then we can use the alignment score between this ideal kernel and any kernel K to judge the quality of K.

Previous works have only considered the two class case for the ideal kernel. In fact, this kind of definition of ideal kernel can be easily extended to multi-classes through following definition [90]:

$$K_{ij} = \begin{cases} 1 & \text{if } \ y_i = y_j \\ -1/(C-1) & \text{otherwise} \end{cases} \tag{3.14}$$

where C is the number of classes.

Another criterion that we considered is to use the norm of the classification hyperplane. Consider a standard multi-class MKL setting as shown in (2.7), the term $\sum_{m=1}^{F} \beta_m \|\mathbf{w_m}\|^2$ is usually considered as a regularization term. Besides,

it also reflects the margin between the classification hyperplane and the data points. From the decision function (2.6), we could see that the MKL classifier is a linear classifier in the high dimensional kernel space. Recall that for a typical linear classification problem[2] $y = wx + b$, if the data points are linearly separable, then there exist two hyperplanes: $wx + b = 1$ and $wx + b = -1$, between which there doesn't exist points. By using geometry, the distance between these two hyperplanes is $\frac{2}{\|w\|}$. Therefore, by minimizing $\|w\|$, SVM is in fact maximizing the margin between the two hyperplanes. This conclusion holds for linear SVM, it also holds for MKL, as MKL still adopts a linear classifier. The only difference is that MKL classifier is performed in the kernel space instead of the original feature space. Based on the above analysis, we propose the second criteria for selecting the canonical kernel histogram, which is to use the norm $\sum_{m=1}^{F} \beta_m \|w_m\|^2$.

More specifically, the procedure to utilize the above two measures for choosing the canonical kernel histogram is as follows: for any kind of kernel histogram, it is firstly chosen as the canonical kernel histogram. All the kernels are transformed according to this chosen kernel's histogram. Then we can adopt any typical MKL algorithms to learn the optimal combination. After that we could obtain the norm score and the Kernel Alignment (KA) score. Based on these scores, we can choose the optimal canonical kernel histogram which produces the smallest norm score or the largest KA score.

In summary, our new systematic baseline feature combination method can be summarized as follows:

Step 1: Normalize the training kernel and the corresponding testing kernel;

Step 2: Based on the Kernel Alignment score or the norm score, pick a target kernel's distribution;

[2] http://en.wikipedia.org/wiki/Support_vector_machine

Step 3: Transform all other kernels to this target kernel distribution by the histogram matching algorithm shown in Algorithm 1;

Step 4: Transform the indefinite kernel to a positive semi-definite kernel;

Step 5: Combine the kernel by using any MKL algorithms.

In step 1, when normalizing the kernel, each K_{train} is normalized to (0,1) by $K_{train} = (K_{train} - K_{min})/(K_{max} - K_{min})$, where K_{max} and K_{min} are the overall minimum and maximum of the matrix K_{train}. Note that as K_{train} is normalized, we also need to normalize K_{test} by still using the same K_{max} and K_{min} value: $K_{test} = (K_{test} - K_{min})/(K_{max} - K_{min})$.

3.6 Experimental Results

3.6.1 Adaptive Kernel Combination

3.6.1.1 Experiment Setup

To systematically test the performance of our proposed method, we have tried to address the following questions:

- Firstly and most importantly, can the proposed method boost the performance of the original MKL?
- Can the proposed measures for automatically selecting the canonical kernel histogram perform as well as the cross validation technique?
- Do we have to select one of the existing kernels as the canonical kernel histogram? Is there any other possible forms of the canonical kernel histogram?

To answer the first question, we will do experiments across several datasets to prove the efficiency of our method. To answer the second question, we will list the final combined kernel accuracy as well as the measure score of each possible kernel, in order to see whether the proposed measure is highly correlated with the final accuracy. To answer the third question, we have considered settings when we choose a pre-specified uniform distribution or a gaussian distribution as the canonical kernel histogram. As these distributions are data independent, we have also considered another data dependent form which is detailed as follows:

For every kernel, we divide it into two parts, the intra class kernel $K_{intra} = \{K_{ij}, \forall i, j : y_i = y_j\}$ and inter class kernel $K_{inter} = \{K_{ij}, \forall i, j : y_i \neq y_j\}$. Then we calculate their histograms separately. In practice, as there are always limited number of training samples for each class, which will make the size of the intra kernel much smaller than the inter kernel, and it will make the estimation of the intra-kernel histogram not robust. Therefore, we prefer to use regression method to get a smoother version of the histogram. We have considered using parametric regression method, and we have tried gamma distribution and weibull distribution, but they didn't exhibit satisfactory results. After that, we decided to choose the nonparametric regression method: Gaussian Process Regression [91] to regress the data. A typical example is shown in Fig.3.2. Then we combine the smoothed inter- and intra- histogram as a potential candidate for the canonical kernel histogram: $H_i = max(Inter_i, Intra_i)$.

3.6.1.2 Oxford Flowers Dataset

The Oxford flowers dataset [92] contains 17 different kinds of flowers. Each class contains 80 samples, 40 for training, 20 for validation, and the remaining 20 for testing. The authors of [92] have also made the distance matrixes

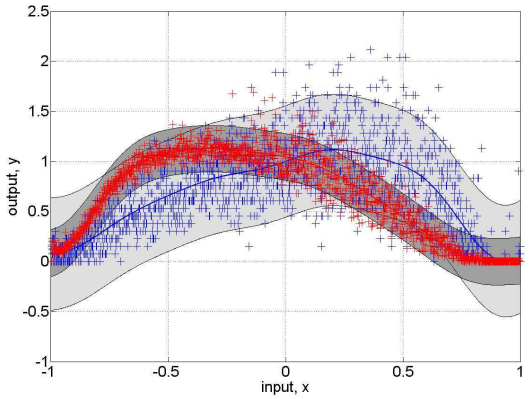

FIGURE 3.2: The blue points correspond to the histogram of intra kernel, while the red points correspond to the inter kernel. The regression results are shown as the blue line and the red line, and the shaded area corresponds to the area in one standard deviation. To accelerate the regression speed, the histogram bin index, which ranges from 1 to 1500, are linear mapped to (-1,1). Ideally, the intra kernel value should be larger than inter kernel value; therefore the combined histogram should be double peaked.

they used publicly available[3]. Following [93], these distance matrixes are transformed to kernels using $k = \exp(-\gamma^{-1} \cdot d)$, where γ is the mean of the distance matrix, and d is the distance between samples. The kernel histogram of these 7 features are shown in Fig.3.3. For the MKL solver, we had chosen a state-of-art solver named OBSCURE [93].

Table.3.1 list the results when we choose different feature histograms as the canonical kernel histogram. As the original OBSCURE only gets an accuracy of 84.9±1.4, we can see that in 4 out of 7 features, we can get a better performance. This partially answers the first question in section 3.6.1.1. From Table.3.1, we can also see that the 2 smallest norm scores correspond to those best performed two kernels, and their corresponding KA score are ranked 3 and 1 respectively. This means the two criteria we proposed are indeed useful for selecting the canonical kernel histogram, and the norm score criterion works slightly better

[3]http://www.robots.ox.ac.uk/ vgg/data/flowers/17/index.html

than KA score on this dataset. This answers the second question. In answering the third question, the low performance of histogram equalization and the predefined gaussian distribution indicates that there is little chance of finding a universal canonical kernel histogram, and the canonical kernel histogram should be data dependant. What is also listed in Table.3.1 is the performance when we choose the intra-inter histogram as the canonical kernel histogram. It exhibits similar performance when we directly choose the original histogram.

TABLE 3.1: Experimental results on Oxfordflower. OBSCURE[93] is one of the state-of-art MKL solvers. HE is short for Histogram Equalization, and Gaussian represents a predefined $\mathcal{N}(0,1)$ normal distribution. From the table, we could see that the best performed two kernels: K_color and K_hsv, have the 2 smallest norm score and their KA score are ranked as 3 and 1 respectively.

	OBSCURE	HE	Gaussian
Accuracy	84.9±1.4	57.9±3.3	60.9±0.3
Norm score	425.8	364.9	489.0
KA score	0.19	0.12	0.07

	K_color	K_hog	K_hsv	K_shape	K_siftbdy	K_siftint	K_texture
Accuracy	**87.5±0.9**	84.7±1.3	**86.9±1.1**	86.9±0.9	83.8±1.0	86.4±1.3	79.0±2.3
Norm score	**389.6**	466.9	**379.9**	431.6	446.7	432.9	430.7
KA score	**0.24**	0.10	**0.26**	0.15	0.25	0.19	0.11
Intra-inter Accuracy	86.9±0.6	84.3±2.1	85.5±1.1	86.6±1.3	84.9±1.5	**87.4±1.8**	77.0±3.0
Norm score	384.8	451.0	369.2	423.8	385.7	405.7	423.5
KA score	0.22	0.10	0.23	0.15	0.30	0.20	0.11

3.6.1.3 MSRC21 Dataset

Next, we consider another example in semantic segmentation area. MSRC21 is a well-known dataset which contains 591 images. Each image has pixel level ground truth labels from 21 semantic classes. Following [9], these 591 images are split into 276 for training, 59 for validation, and the remaining 256 for testing.

Here, our objective is to evaluate feature combination rather than aiming to achieve state of the art semantic labeling performance. Therefore, we simplify the original semantic segmentation problem into a region labeling problem, i.e. we assume the image has already been segmented into regions, and our task is to assign each region a semantic label. To avoid bias, we directly use the ground truth segmentation. We adapted the code from [66], where they used three features to represent each region. These three features are texton histograms, color histograms and pyramids of HOG. Their respective kernel histograms are shown in Fig.3.3. To avoid the bias of different MKL solvers, we still adopt OBSCURE as our MKL solver. Results are shown in Table 3.2, where we have presented results based on four different evaluation criteria. For a detailed explanation of these four evaluation criteria, please refer to [9, 12].

TABLE 3.2: Experimental Results on MSRC21. Accuracy1 = Per-class region-wise accuracy. Accuracy2 = Overall region-wise accuracy. Accuracy3 = Per-class pixel-wise accuracy. Accuracy4 = Overall pixel-wise accuracy. From the table, we could see that K_texton performs best, and it also has the lowest norm score and the highest KA score among the three kernels.

	Baseline	Feature Histogram					Intra-inter Histogram		
	OBSCURE	HE	Gaussian	K_texton	K_color	K_hog	K_texton	K_color	K_hog
Accuracy1	77.2	49.8	80.6	**82.6**	78.0	80.6	79.1	64.0	77.2
Accuracy2	84.3	63.0	85.5	**87.3**	83.8	85.5	84.8	76.1	82.7
Accuracy3	81.5	46.4	82.5	**84.4**	80.3	82.5	82.5	68.2	80.6
Accuracy4	90.8	61.6	91.0	**92.4**	89.6	91.0	90.9	81.8	89.0
Norm score	1379.8	1971.8	1992.1	**1004.0**	2022.6	1992.1	1886.9	3612.1	3349.8
KA score	0.14	0.15	0.13	**0.14**	0.14	0.13	0.12	0.12	0.12

From Table.3.2, we could see that the performance of OBSCURE is boosted by introducing the histogram matching module. The kernel corresponds to the texton histogram exhibits the highest performance which can be predicted from either the norm score or the KA score. Histogram equalization didn't perform well while the predefined gaussian distribution performs comparably. The intra-inter histogram didn't perform as well as the original histogram.

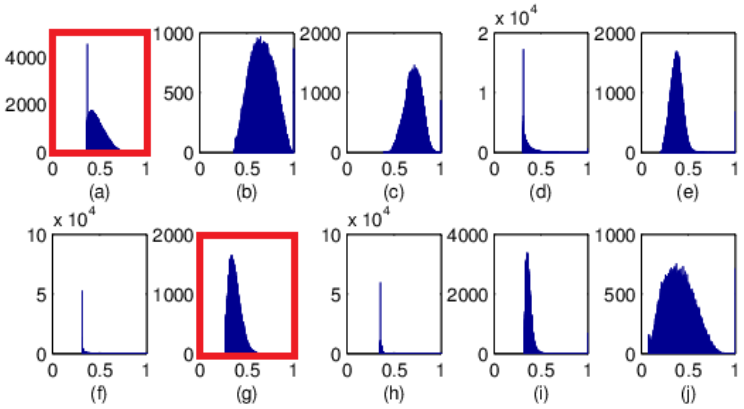

FIGURE 3.3: Kernel histograms of three features used in MSRC21 ((a) to (c)) and seven features ((d) to (j)) used in Oxfordflowers. The histogram in the red box is chosen as the standard histogram.

3.6.1.4 UCI Machine Learning Repository

We have also done experiments on the UCI repository datasets. On the Sonar and Breast dataset, we did experiments exactly following [94]. Three kernels are used: a quadratic kernel, an RBF kernel and a linear kernel. We report mean test accuracy across ten random replications of three-fold cross validation. For the MKL solver, we still chose to use OBSCURE. The results are shown in Table 3.3.

From Table.3.3, we could see that we get the highest accuracy when the RBF kernel is chosen as the canonical kernel histogram, and it outperforms the original OBSCURE. Both the norm score and the Kernel Alignment (KA) score have correctly predicted this. Besides, both the histogram equalization and gaussian distribution didn't perform well on the sonar, although they exhibited a remarkable performance on the breast. However, their corresponding norm score and KA score are not in accordance with their respective performance. Regarding to the intra-inter histogram, it didn't perform well on the sonar dataset, but performs slightly better on the breast.

TABLE 3.3: Experimental Results on UCI datasets. The above table shows the results on Sonar dataset while the bottom is for breast. K_rbf performs best across the two datasets, and it also has the lowest norm score and the highest KA score among the three kernels.

	Baseline	Prefixed histogram		Feature Histogram		
Sonar	Obscure	HE	Gaussian	K_quadratic	K_rbf	K_linear
Accuracy	88.0±4.3	72.6±4.4	74.2±4.2	86.2±4.2	**88.2±4.0**	80.0±3.6
Norm score	50.08	34.8	52.1	60.20	**50.06**	60.49
KA score	0.1172	0.0538	0.0429	0.0700	**0.1174**	0.0519
Breast	Obscure	HE	Gaussian	K_quadratic	K_rbf	K_linear
Accuracy	96.7±5.0	97.6±1.0	97.3±0.9	96.6±1.0	**97.6±0.9**	97.2±0.9
Norm score	27.8	26.6	44.1	54.6	**28.7**	42.8
KA score	0.34	0.33	0.19	0.31	**0.34**	0.31

3.6.1.5 Results Summary

Our results demonstrate that statistical kernel transforms can always boost the performance of the original state of the art MKL solver. Transforming the possibly non-positive definite matrix into positive semi-definite, which will make the method theoretically justified, will not deteriorate performance in practice. Both the norm score and the KA score are predictive in selecting the canonical kernel histogram, and we have also found that norm score performs slightly better than the KA score on the Oxfordflower and MSRC21 datasets. It seems very difficult, if not impossible, to find an universal canonical kernel histogram, and the canonical feature histogram should be data dependant. The relatively low performance of the intra-inter histogram suggests that we should better directly use the original kernel histogram. Although the idea of pursuing a double peaked histogram seems appealing(this is exactly what the intra-inter histogram pursues), as it makes it easier for the kernel classifier to do its classification job, our results so far have not shown this is helpful.

3.6.2 Non-adaptive Kernel Combination

Although MKL algorithm shows superior performance in some datasets, some researchers found it not well suited for some particular datasets [95]. In some cases, the complicated nature of the problem itself or the relatively small available training data makes even non-adaptive kernel combination outperforms its adaptive counterpart. Examples include the work presented in [1, 84]. In these circumstances, the non-adaptive kernel combination method is usually adopted. In the following experiments, we show that in these circumstances, our method can also boost the performance than traditional non-adaptive kernel combination method, such as 'average'.

3.6.2.1 Corel5K Dataset

In [1], the authors studied the problem of image annotation. They showed that by simply adding the distances of different features, they can achieve superior performance on the corel5K benchmark image annotation dataset. They used features representing color and texture, and the distances of each feature channel are equally weighted. They called their algorithm Joint Equal Contribution (JEC). In [46], the authors proposed to use another 15 kinds of features including global and local features. They reported similar results to JEC. They have also released their features[4] used in the experiments. We did experiments directly based on these features. Different metric measures [46] are adopted to calculate the distances in each feature channel. The histograms corresponding to the distances of those 15 kinds of features are shown in Fig.3.4. From there we can see an obvious difference between different feature channels.

[4] http://lear.inrialpes.fr/people/guillaumin/data.php

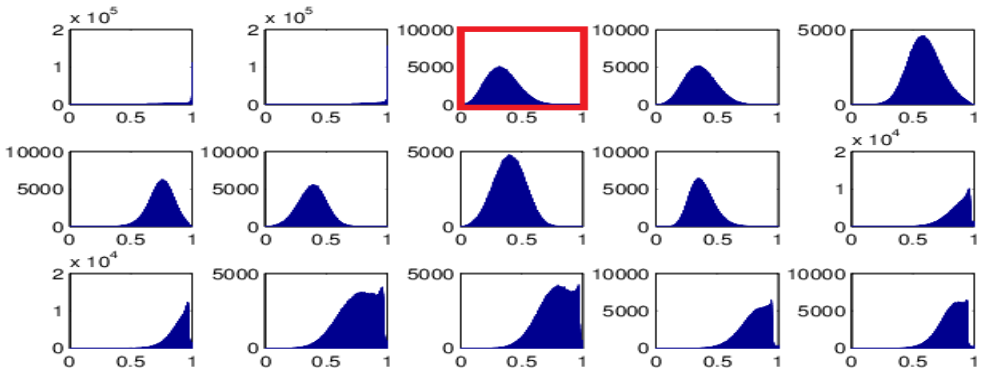

FIGURE 3.4: Kernel histograms of the features used in [46]. The histogram in the red box is chosen as the standard histogram, which corresponds to the feature of hue descriptors extracted at Harris-Laplacian interest points

As there is no theoretical guidance on how to choose a standard histogram, we use cross validation to choose one from these 15 features as the canonical feature. The histograms after histogram matching are shown in Fig.3.5.

Those distances after histogram matching are added together. Based on this added distance, the K nearest neighbors for each test sample are retrieved from the training set. The tags of each test sample are solely determined by these K nearest neighbors. In predicting the tags from these K neighbors, we also adopted the label transfer strategy used in [1]. Precision and recall are used to evaluate the performance and the results are shown in Table 3.4. From there we can see a performance boost by introducing the histogram matching module. It is important to point out that the purpose here is not to compete with the state of the art image tagging performances but rather to demonstrate that by calibrating the kernels using simple histogram matching before combining them can improve performance.

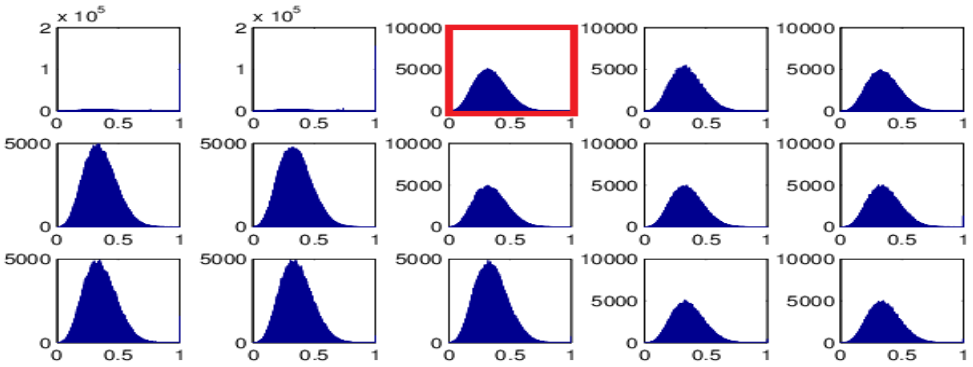

FIGURE 3.5: Histograms in Fig.3.4 after histogram matching.

TABLE 3.4: Image Annotation Performances on Corel5K Dataset. HM is short for **H**istogram **M**atching. Rate+ is the number of tags whose recall is above zero.

models	Prec	Recall	Rate+
HPM [96]	0.25	0.28	136/260
JEC [1]	0.27	0.32	139/260
JEC-15 [46]	0.28	0.33	140/260
JEC-15 + HM	0.30	0.36	150/260

3.6.2.2 Caltech101 in 39 Kernels [84]

In [84], the authors thoroughly studied the problem of feature combination. One of their important findings is that a simple average kernel may outperform sophisticated MKL algorithms. They have also released their code and the gram matrixes[5] used in their experiments. The best result they got was based on a combination of 39 kernels. These different kernels are mainly based on 5 different kinds of features: LBP, PHOG, SIFT, Region covariance and Gabor filter banks. Those features are assembled in different layouts, resulting in a total of 39 kernels. In their work, they have already compared their results with typical MKL algorithms, including SILP [97] and SimpleMKL [98]. In some cases, simple average kernel may outperform these complicated MKL methods.

[5] http://people.ee.ethz.ch/ pgehler/projects/iccv09/caltech/

We did experiments directly on these publicly available gram matrixes. Some of the representative histograms are shown in Fig.3.6. Experimental results are shown in Fig.3.7. Again, we can see a performance boost by introducing the histogram matching module before combining the kernels. Once again it is worth mentioning that our goal here is not to compete with state of the art object categorization techniques but to demonstrate the effectiveness of introducing histogram matching for feature combination.

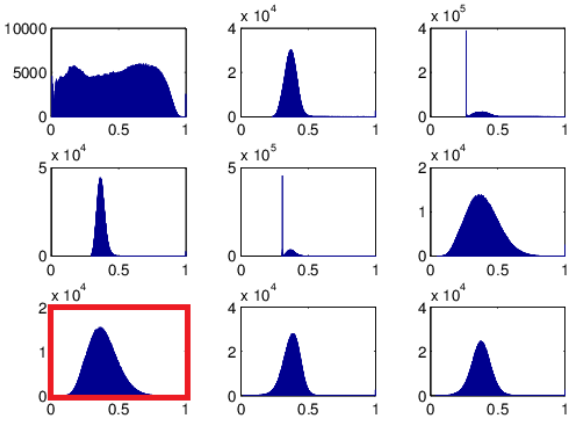

FIGURE 3.6: Some representative kernel histograms among the 39 kernels used in [84], the one in the red box is the best performing canonical kernel histogram.

3.6.2.3 Results Summary

These experiments clearly showed that our method is not only applicable for supervised MKL cases, it also fits for unsupervised scenarios, where there is a need to sum different similarity measures together. In common practice, one only need to make the mean (1st order moment) or the variance (2nd order moment) of different similarities to be the same [99]. Our method, which tries to make the histograms to be the same, is actually standardizing the higher order moments. Therefore it could be considered as a new baseline.

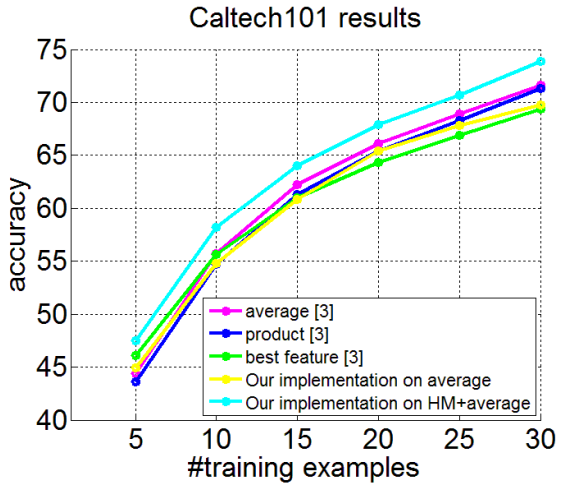

FIGURE 3.7: The classification results on Caltech101. From the figure, we can see the average of histogram matched kernels can always perform better than averaging original kernels. Note that the author of [84] report results on five random splits of the dataset. However, they have only released their gram matrixes of one split. We did experiments only on this split. This results in the slight difference between our implementation on average and the average accuracy reported in [84].

3.7 Concluding Remarks

In this chapter, we have proposed a new feature combination method which calibrates the kernels to have the same histogram before linearly combining them. Experiments on various datasets have shown the effectiveness of this simple strategy. This method can be used in the unsupervised scenario where it consistently performs better than the average baseline. In supervised cases, it can be seamlessly combined with state of the art multiple kernel learning algorithms and we have shown it again can boost performance.

Besides the experiments shown in this chapter, actually we have also done experiments based on the work of [4, 100] on caltech101 and scene15 datasets. We can get subtle improvements over their works: on scene15 dataset, we can boost the accuracy from 89.15% [100] to 89.57% under 100 training image for each category; on caltech101, the performance gain is from 78.5% [4] to 78.6%.

As we further analyze the reason why we didn't get significant improvements is probably because the features used in [100] is very comprehensive[6], and the authors use a finely tuned weight for each feature channel: four times of each feature channel's individual performance. Thus the baseline method has already achieved a very high performance. In [4], the histograms of the 10 kernels[7] used in the experiments are already very similar, thus our method can not get significant improvements. But still, we believe that histogram matching should be an important module in future feature combination systems. Even if in the worst case, it will not deteriorate the performance.

[6] 14 kinds of features in combination with different kinds of kernels, resulting in total 59 different kernels

[7] http://www.robots.ox.ac.uk/ vgg/software/MKL/

Chapter 4

Integrating Low-level and Semantic Features for Object Consistent Segmentation

In this chapter, we studied the problem of pixel-level semantic image understanding. Different from most of the previous methods that chose pixel or superpixel as the processing primitive, we advocate to choose regions as processing primitives. We first experimentally showed that using ground truth segments as processing primitives can boost semantic segmentation accuracy, and then proposed a novel method to produce regions that resemble the ground truth regions, which we termed as object-like regions. We achieve this by integrating state of the art low-level segmentation algorithms with typical semantic segmentation algorithms through a novel semantic feature feedback mechanism. We present experimental results on the publicly available image understanding dataset MSRC21 and the stanford background dataset, showing that the new method can achieve relatively good semantic segmentation results with far fewer processing

primitives. Furthermore, we have also proposed to integrate semantic segmentation with an interactive segmentation module, and some qualitative results have shown the promise of our approach.

In this chapter, we first introduce some background on pixel level semantic image understanding in section 4.1. Then in section 4.2 we describe our algorithm on integrating semantic segmentation with low-level segmentation. We present our experimental results on two well known datasets in section 4.3. Section 4.4 describes our approach for combining semantic segmentation with interactive segmentation, and some concluding remarks will be given in section 4.5.

4.1 Introduction

Just as we discussed in Chapter 2, to understand the semantic meaning of an image at the pixel level, there are possibly two strategies: the first one is to adopt the segment-then-recognize strategy, i.e. first segmenting an image into regions, then recognizing each region one by one. The other strategy is to choose every single pixel as the processing primitive, and directly assign a semantic label to it.

By comparing these two strategies, it is not difficult to see that the second strategy will produce object non-consistent results (as shown in Fig.4.1), and each object is just a group of pixels or superpixels with the same semantic labels. In contrast, the first strategy will remedy this drawback as long as we can design an appropriate segmentation algorithm that can produce more object-consistent regions.

Therefore, the biggest challenge in adopting the first strategy is how to make the segmentation algorithm capable of producing regions that are more object-consistent. Taking the image in Fig.4.1 as an example, the head of the sheep

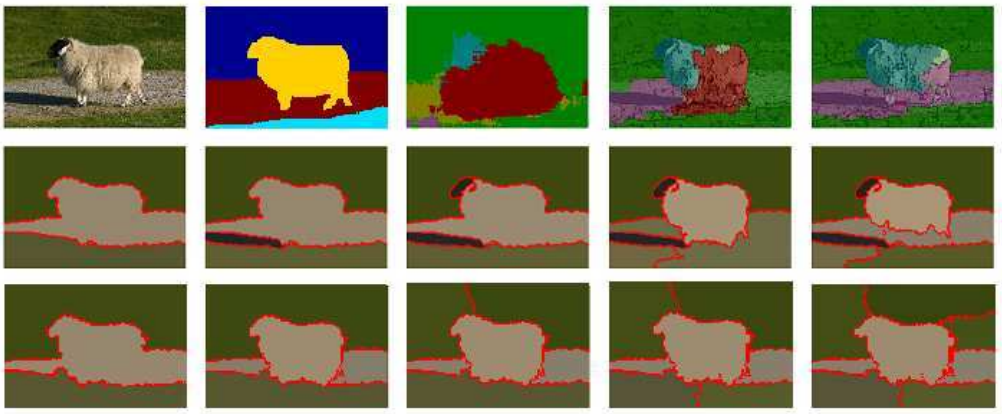

FIGURE 4.1: Top row: (left to right) original image, ground truth segmentation, pixel based semantic labeling [9] without CRF [57] post processing, two groups of superpixel based methods [101] by varying the parameters of mean shift; Middle row: a state of the art spectral segmentation method named Full Pairwise Affinity (FPA) [102] which considers only color features to produce K (K=3,4,...7) clustering; Bottom row: our full model by combining color features and semantic features to produce K clustering. It is seen that in all cases our method consider the head and the body of the sheep to be one object.

(black color) and the body of the sheep (white color) are totally different from each other, but still they belong to one object. Therefore, we could not expect that our segmentation module which is purely based on low-level features can produce object-consistent regions.

How can the segmentation algorithm consider the head of the sheep and the body of the sheep to be one object? The only way is to incorporate supervised information into the segmentation module. Suppose we have a set of training images together with their ground truth segmentations, probably there exist, among these training images, a sheep with black head and white body. Thus it is possible to utilize these training images to guide our segmentation algorithm.

4.2 Combining Low-level and Semantic Segmentation for Object Proposal

4.2.1 The Advantage of Choosing Regions as Processing Primitives

Before we introduce our newly proposed framework, we would like to first emphasize the importance and advantage of choosing regions as processing primitives. We choose MSRC21 as our test bed. MSRC21 contains 591 images from 21 semantic classes. Following [9], the dataset is divided into 276 images for training, 59 images for validation, and the remaining 256 images for testing. For the object regions, we directly use the clean ground truth segmentations[1] on this dataset. These ground truth segmentations are directly fed into our region labeling module to be described in section 4.2.6. We obtain an overall global pixel accuracy of 90.8%, outperforming any state-of-art methods [68, 101] on this dataset. This experiment clearly shows the advantage of choosing the regions as our processing primitives. However, the challenge is how to automatically generate such object regions. We present such a method in the following subsections: section 4.2.2 will present a general picture of the whole framework, then several of its important modules will be introduced in the subsequent subsections.

4.2.2 The Proposed Framework

Given an input image, several kinds of low-level features can be extracted, and these features are fed into a low-level spectral segmentation module. These low-level segmentation algorithms can produce low-level consistent regions, which

[1] http://www.cs.cmu.edu/ tmalisie/projects/bmvc07/

we call superpixels here. A typical semantic segmentation algorithm can either choose these superpixels or directly choose the image pixels as the processing primitives, and their outputs can be considered as semantic features. Those semantic features are then combined with the low-level features and fed back to the low-level segmentation algorithm again. This time, as the features contain semantic feature, we call the segmentation module mid-level segmentation. Ideally, they will segment images into regions that are both low-level and semantic-level consistent, i.e. they are object-like regions. Based on those object-like regions, typical classification algorithms can be used to achieve the goal of image understanding. The flow chart of this procedure is shown in Fig.4.2.

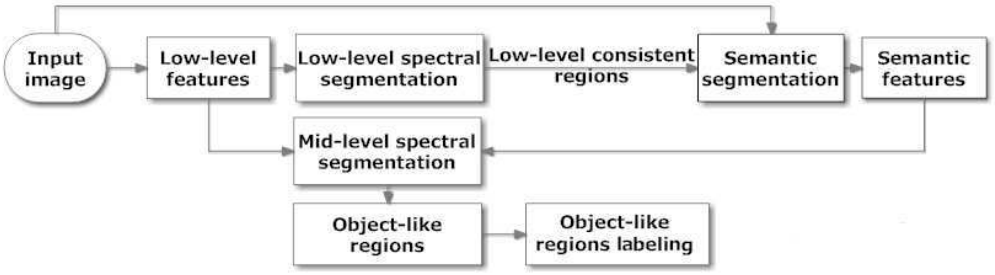

FIGURE 4.2: The flow chart of the proposed framework

In essence, the semantic segmentation algorithm is still based on low-level cues; our algorithm can therefore be considered as having introduced a feedback from the classifier output to the input level. We believe this idea is an important contribution of our work. As we have also noticed similar ideas being adopted in some other scenarios, we believe it is an important philosophy and could be generally adopted in many scenarios.

In [25], the outputs of various object detectors are treated as new mid-level features; in [103], the authors learned a bank of weak classifiers from the web-retrieved images, and the output of these weak learners are also treated as mid-level features. In this theme, the output of semantic segmentation algorithm used in our low-level segmentation module can also be considered as mid-level features. In [104], the authors first learned a classifier on the local patch, and

then the output of the classifier is considered as the context information of the target patch. By concatenating them together, the author retrained the system. A similar strategy is also adopted in [105], where the authors use the geometric context detector to detect the layout of similar images, then the average of those outputs are considered as a prior, based on which the author retrained the system. From those previous works, we came to a conclusion that the output of classifier contains useful information. They can enhance the performance of the original system through a carefully design. This also justifies the applicability of the framework we proposed here.

4.2.3 Spectral Methods for Low-level Segmentation

For the low-level spectral segmentation module, we adopt the multi-layer graph method proposed in [102], as it produces the state of the art results on the BSDS dataset [106]. A multi-layer graph is represented by $G^* = (V^*, E^*)$, where the nodes V^* are a set of pixels and superpixels, and edges E^* exist between neighboring pixels, neighboring superpixels, and also between the superpixel and all the pixels it includes. The edge weights are defined as:

$$w_{ij} = \begin{cases} exp(-\theta_g \parallel g_i - g_j \parallel) & if \;\; i,j \in pixels \\ exp(-\overline{\theta}_g \parallel \overline{g}_i - \overline{g}_j \parallel) & if \;\; i,j \in superpixels \\ const & otherwise \end{cases} \tag{4.1}$$

where g_i is the color value (in Lab space) of pixel i, and $\overline{g}_i$ represents the mean color of all the inner pixels contained in a superpixel i. θ_g and $\overline{\theta}_g$ are constants that control the strengths of the weight. They can be specified either manually or using cross-validation techniques. For details of this algorithm, please refer to [102].

4.2.4 Pixel or Superpixel based Semantic Segmentation

For the semantic segmentation module, we adopt the typical two order Conditional Random Field (CRF) [57] based methods. Just as shown in (2.11), a CRF defines a posterior distribution over hidden random variables $\mathbf{Y}$ (labels), given observed image features $\mathbf{X}$, in a factored form:

$$p(\mathbf{Y}|\mathbf{X}) = \frac{1}{Z}\exp(-\sum_{c\in C}\psi_c(Y_c, \mathbf{X})) \tag{4.2}$$

where Z is a normalizing constant, and C is the set of all cliques.

Given a set of images and its corresponding ground truth labels, the training procedure of a CRF aims to make the energy of the ground truth label assignment corresponds to the minimum of the energy function. After the model is trained, for a new test image, its most probable labeling $\mathbf{Y}^*$ is defined as

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}\in\mathcal{Y}} p(\mathbf{Y}|\mathbf{X}) \tag{4.3}$$

where $\mathcal{Y}$ corresponds to any kinds of possible labeling.

Besides the most probable labeling $\mathbf{Y}^*$, we can also get the marginal posterior distribution $p(Y_i)$ of any node i. We choose to use this marginal distribution as our semantic feature. Compared with directly choosing the MAP assignment as the semantic feature, it clearly enjoys some benefits. For example, for two neighboring nodes, we can not only judge if they are most likely to belong to one specific semantic class, but also we could say whether they are all dissimilar with another semantic class.

In our experiments, for the pixel based semantic segmentation, we adopted the Textonboost method [9] and used their publicly available code[2]; for superpixel

[2]http://jamie.shotton.org/work/code.html

based semantic segmentation, we adapted the code from the STAIR Vision Library[3], which uses piecewise training method [107] to train the CRF and uses max-product propagation [108] for inference.

4.2.5 Mid-level Segmentation with Semantic Feature Feedback

We propose to introduce the semantic features generated by CRF into the above mentioned low-level spectral segmentation module, enabling the spectral segmentation algorithm to produce both low-level and semantically consistent regions.

One intuitive way of achieving this goal is to redefine the edge weight function of equation (4.1). Note that in equation (4.1), the weight between neighboring nodes only depends on their low-level features. By introducing our semantic segmentation module, we can also obtain the semantic feature of each node. Thus, we redefine the weight between neighboring nodes as a combination of their low-level feature similarity and their semantic feature similarity:

$$w_{ij} = \begin{cases} \alpha \exp(-\theta_g \parallel g_i - g_j \parallel) + (1-\alpha)\exp(-\theta_s \parallel s_i - s_j \parallel) \\ \qquad\qquad \text{if } i,j \in \text{pixels} \\ \alpha \exp(-\bar{\theta}_g \parallel \bar{g}_i - \bar{g}_j \parallel) + (1-\alpha)\exp(-\bar{\theta}_s \parallel \bar{s}_i - \bar{s}_j \parallel) \\ \qquad\qquad \text{if } i,j \in \text{superpixels} \\ \text{const} \qquad\qquad \text{otherwise} \end{cases} \tag{4.4}$$

[3] http://robotics.stanford.edu/ sgould/svl/

where s_i corresponds to the semantic feature of node i, here it is equivalent to the marginal probability $p(y_i)$ of node i in our CRF paradigm. $\alpha \in (0, 1)$ is a trade-off parameter between low-level feature and semantic feature. When it equals to 1, (4.4) degrades to (4.1); when it equals to 0, (4.4) will generate exactly the same regions as the semantic output, and our algorithm can be considered as a verification step after the CRF. We empirically set $\alpha = 0.5$ in all our experiments to make the semantic features and low-level features contribute equally to the segmentation module. $\| \ \|$ represents the norm of the vector. We experimentally compared different kinds of norms and choose the $L1$ norm which performs best in our experiments. As has been shown by others, the $L1$ norm tends to produce better results for its robustness against outliers [96].

4.2.6 Object-like Primitives Labeling

Having produced object-like regions using the spectral segmentation algorithm with semantic feature feedback, we now extract features of those object-like regions and build a classifier to label them. As the object-like regions are usually large and contain many pixels, we believe that their histogram features are more robust and discriminative. Similar to [66], several kinds of histogram features, including Texton Histogram, Color Histogram and pHOG Histogram are extracted from each ground truth region among all the training images. All these histograms are obtained by vector-quantizing and pooling the corresponding features. We use χ^2 kernel to measure the similarities among those histograms. These kernels can then be added together, and a plain SVM can be adopted as the classifier based on this combined kernel.

To enhance the performance of the classifier, we have adopted two additional techniques: the first one is to use the Multiple Kernel Learning (MKL) approach to replace the plain SVM.

The second technique we adopted is to augment the training set. We select the regions as processing primitives, and the number of the ground truth regions contained in the training set is always very limited. For example, there are only about 10 regions for some semantic classes in the MSRC21 dataset. Therefore it is very easy for the classifier to get overfitted. This is in contrast to selecting superpixel or pixel as processing primitives, where we can obtain tens of thousands of training samples from the training set. Therefore, it is necessary and beneficial if we can augment the training set. We achieved this through the following procedure: we make the low-level segmentation module to perform on the training images. If they generate some pure regions according to the ground truth semantic labeling, then these regions are also considered as positive training samples and augmented the training set.

This strategy is in the similar spirit with the 'mining hard negatives' technique widely used in the object detection literature [61]. Both methods advocate reevaluating the algorithm on the training set to obtain more training samples. Note that in our scenario, we are not limited to any specific kinds of segmentation algorithm. Instead, we can use any kind of low-level segmentation algorithm, and we can even utilize our full model to perform on the training set. As long as these segmentation algorithms generate some pure regions according to the ground truth semantic labeling, we can augment these regions to the training set.

We will show in the experiment section that both of these two techniques will help to boost performance.

4.3 Experiments

4.3.1 Segmentation Quality

To assess the quality of the regions generated by our model, we adopted the segmentation covering [109] as the accuracy measure. The covering of a segmentation S by a segmentation S' is defined as

$$C(S, S') = \frac{1}{N} \sum_{R \in S} |R| * \max_{R' \in S'} O(R, R') \tag{4.5}$$

where N denotes the total number of pixels in the image, $|R|$ represents the number of pixels in the region R, and O is the overlap.

We still chose MSRC21 dataset as our testbed. For each test image, we firstly segmented it into a fixed number (3 to 12) of regions using different segmentation algorithms. Then we compared these segments with the ground truth segmentation to compute the segmentation covering score. Here we compared Normalized Cut [110] and Full Pairwise Affinity Model in [102] with our full model. The results are shown in Fig.4.3. From there we can see our full model consistently performs better than the other two algorithms. We also noticed that the highest segmentation covering score is achieved when an image is divided into four regions. This is consistent with the fact that the testing images in this dataset has an average of four ground truth regions approximately.

Furthermore, as we don't know how many regions we should partition an image into, we consider all these generated segments as a segmentation pool, and recompute the segmentation covering score. The results are shown in Table.4.1. Again, we can see an obvious advantage of our full model over its counterparts. Besides, the results we achieved is very close to a state of the art method [109]. However, in [109], it generates a hierarchy of segments of each image. Usually

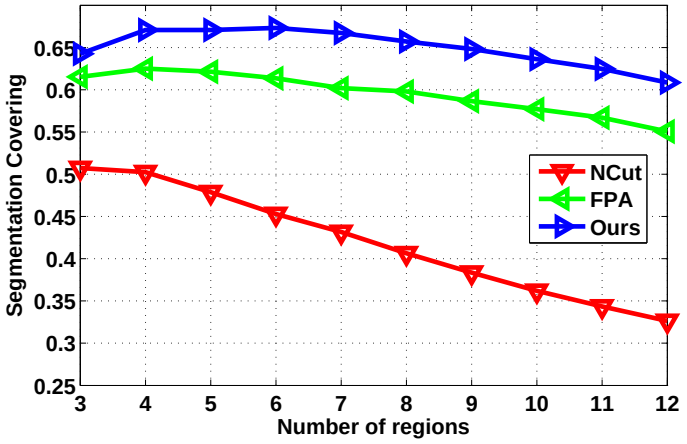

FIGURE 4.3: Segmentation covering score of different methods. X axis represents dividing an image into # regions, while Y axis corresponds to the accuracy. NCut means regions generated by normalized cut, FPA means Full Pairwise Affinity model in [102]

it generates hundreds of segments, whilst our results are obtained based only on a total of 75 (which is the sum of 3+4+...+12) regions.

TABLE 4.1: Segmentation covering score of the segmentation pool

method	NCut	FPA	Ours	gPb-owt-ucm [109]
Coving	0.60	0.73	0.77	0.78

Fig.4.1 shows a typical result of our model. It can be seen there that whilst previous methods have failed to recognize the head and the body of the sheep as belonging to one object, our model has succeeded. Some more qualitative examples are shown in Fig.4.4. We can see there that our algorithm can generate more object-consistent regions than the other segmentation algorithms.

4.3.2 Comparison of Pixel Labeling Accuracy on MSRC21

Based on the regions generated by our full model, we directly use them as our processing primitives and performed region recognition as detailed in section 4.2.6. Statistics of experimental results are shown in Fig.4.5 and Table.4.2. We

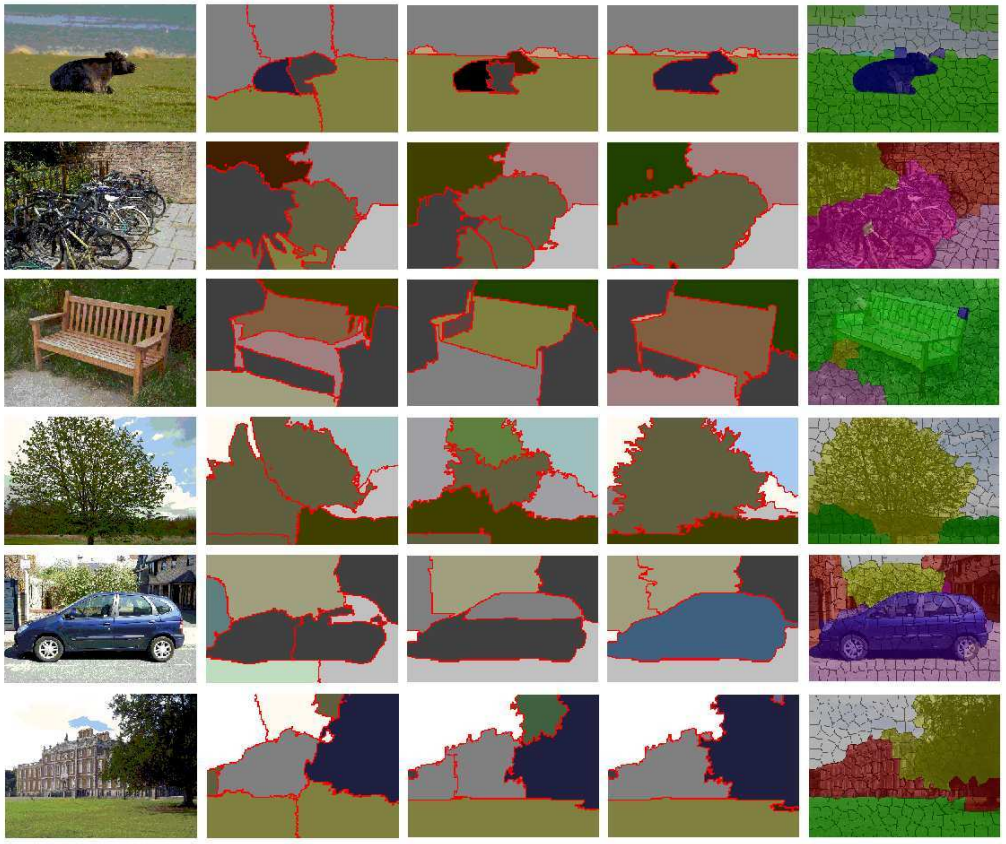

FIGURE 4.4: Some examples of the results of three segmentation methods. Each image is segmented into a predefined 7 regions. From left column to right column: original image; results generated by NCuts, Full Pairwise Affinity (FPA) [102], and our full model. The results of a typical semantic segmentation algorithm [101] is also shown in the last column.

can see that no matter we use the plain SVM classifier or the MKL classifier, our full model consistently produced better results than other two methods. Besides, we could see an improvement on MKL classifier over plain SVM for all the three segmentation methods. This suggests that the usage of an advanced classifier can help to improve the performance.

Compared with most state of the art results ([68]: 77%, [101]: 76.4%) on this dataset, we are getting very close. Although the hierarchical CRF model of [12] demonstrates superior performance: 86%, it should be noted that their pixel-wise classifier can obtain an overall accuracy of 81%, which suggests their use of much more discriminative features [112]. However, the main difference between

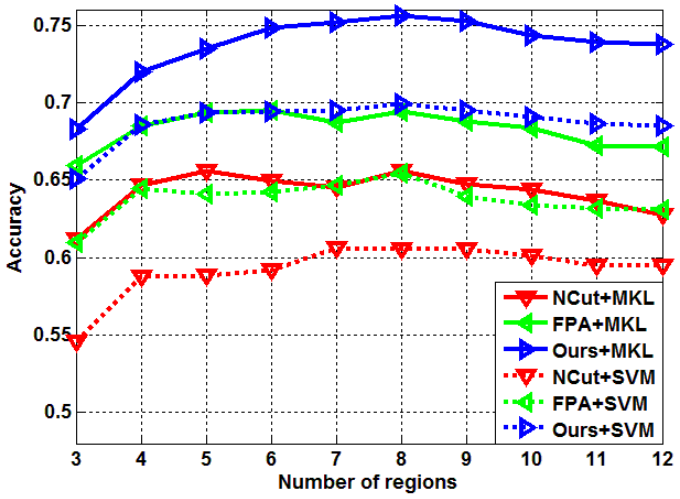

FIGURE 4.5: Global pixel accuracy on MSRC21

TABLE 4.2: Semantic labeling accuracy on MSRC21 when each test image is partitioned into a fixed 8 regions. While 'Global' means the overall pixelwise labeling accuracy, 'Average' means the average of classwise accuracy. The results are shown in terms of percentages.

	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Global**	**Average**
[9]	62	98	86	58	50	83	60	53	74	63	75	63	35	19	92	15	86	54	19	62	7	72	58
[111]	77	93	70	58	64	92	57	70	61	69	67	74	70	47	80	53	73	53	56	47	40	75	65
NCut	55	86	87	56	27	90	30	62	46	53	60	62	57	24	41	36	72	0	21	48	9	67	49
FPA	61	88	87	63	39	93	36	71	42	43	58	68	49	32	48	45	79	21	32	44	12	70	53
Ours	77	89	85	75	52	80	29	76	70	65	75	65	53	33	71	53	81	44	33	62	10	75	61

our algorithm and theirs lies in that their algorithms are superpixel based whilst ours is object-like region based. For a typical image in MSRC21, it is usually segmented into hundreds of superpixels [68, 101]. However, in our case, we only divide the image into a few (3˜12) regions and we achieved similar accuracy. Fig.4.6 shows an illustrative example. It can be seen that whilst superpixel based methods divide the image into many small patches, our model divides it into only a few object like regions. Besides, the results generated by these superpixel based semantic segmentation algorithms are always not object-consistent, just as shown in the last column of Fig.4.4. On the contrary, our algorithm inherently

FIGURE 4.6: From left to right: original image; object-like regions generated by our full model; superpixels commonly used in conventional semantic segmentation methods

enjoys the advantage of producing object-consistent outputs as clearly illustrated in Fig.4.1.

4.3.3 Experiments on Stanford Background Dataset

To further verify the effectiveness of our approach, we also did experiments on the Stanford background dataset [101]. This dataset consists of 715 images belonging to eight semantic classes including sky, tree, road, grass, water, building mountain and foreground object. The images in this dataset are collected from several well-known datasets including PASCAL, LabelMe, and Geometric Context. Each image in this dataset contains about 11 distinct object regions on average, while MSRC21 only contains about 3 object regions. Therefore, this dataset is much more challenging than MSRC21. Following previous work [17, 101] on this dataset, the overall 715 images are divided into 572 images for training and 143 images for testing.

Firstly, we also use the ground truth segmentation to see how much accuracy we could achieve, which can be considered as the upper bound for the subsequent experiments. This time, we achieve a global pixel accuracy of 85.64%, which is again better than any state of the art methods. For example, [101] reported an accuracy of 76.4%, while [17] achieved 77.5%. [113] reported 79.4% which is the highest score on this dataset until the submission of this thesis. All these

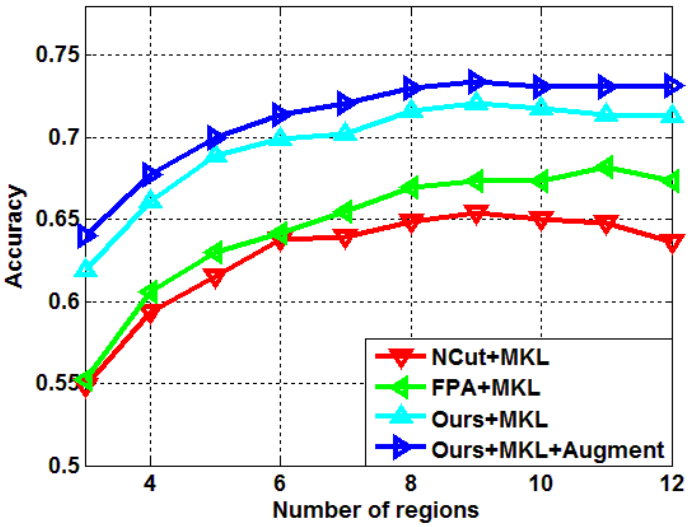

FIGURE 4.7: Semantic class accuracy on Stanford background dataset. 'Ours+Augment' represents results obtained by augmenting the training set.

three methods chose superpixel as the processing primitive and falls into the conventional CRF paradigm. This once again shows the advantage of choosing regions as processing primitives.

The following experiments follow exactly the same procedure as in MSRC21. The results we have achieved are shown in Fig.4.7. In that figure, we have also shown the effect of augmenting the training set strategy which was described in section 4.2.6. Here we have only augmented about 10000 regions for the whole dataset due to the space limitation caused by the kernel SVM, that is about 20 regions per training image. This is still much lower than conventional pixel or superpixel based methods, where the training set is usually as large as hundreds of thousands. From Fig.4.7, we can see that the strategy of augmentation do help improve the performance, and we can expect a further performance boost when augmenting more training samples.

4.3.4 Computational Complexity Analysis

From the above experiments, we could see that our algorithm exhibits a better performance than its counterparts: NCut and FPA. However, this improvement relies on its increased computational burden. For the segmentation algorithm itself, it shares the same complexity with FPA, as we have only redefined the weight function as shown in Equ.4.4. Therefore, the increased computational complexity arises from the computation of semantic features. Although most of the current semantic segmentation algorithm are time consuming, such as the method [9] that we adopted in this work, there do exist some methods that can perform the semantic segmentation in real-time [10]. How to design a faster semantic segmentation algorithm is beyond the scope of this work. It is important to point out that the work we presented in this chapter is a general framework for combining semantic segmentation and low-level segmentation, thus it can be seamlessly combined with any kind of semantic segmentation algorithm.

4.4 Hard Integration

Besides the framework introduced above for fusing semantic segmentation with low-level segmentation, we have also tried another method to achieve this. That is to treat the semantic segmentation module as an unreliable 'teacher', which generates some tokens to guide the low-level segmentation module. Thus the whole system performs in an interactive segmentation fashion. We called this kind of integration **hard integration**, in comparison with the above framework which can be called **soft integration**.

The rationale behind the hard integration is that if the semantic segmentation module consistently considers a relatively large region to belong to one semantic class with high probability, then it has a high chance to be correct. Therefore,

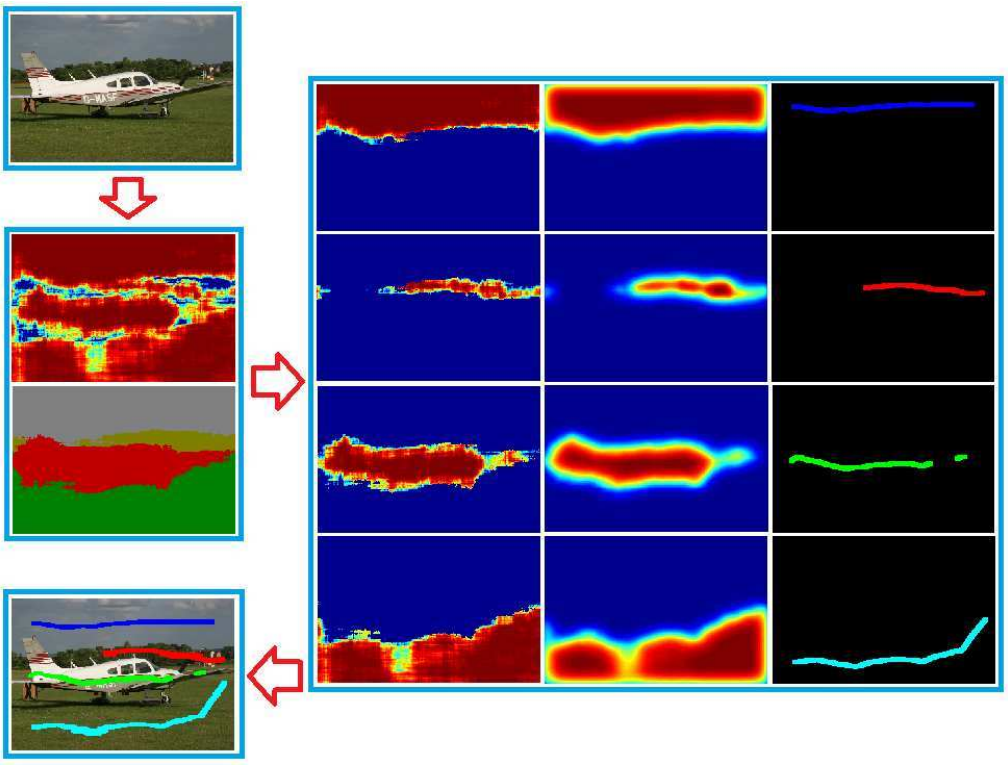

FIGURE 4.8: The procedure for generating the tokens. Based on the outputs from the semantic segmentation algorithm and their predicted semantic classes, the highly probable large regions went through a series of morphological operations including gaussian filtering, threshold, thinning and dilation.

we can treat these large regions as highly reliable regions and generate tokens from it. The procedure of generating the tokens is shown in Fig.4.8. The low-level segmentation module then treats these tokens as 'human' input, and tries to segment images satisfying these token constraints. Here, we adopt the method introduced in [114] to perform the low-level interactive segmentation. Some visual examples are shown in Fig.4.9.

Although the idea of hard integration seems appealing, it cannot perform on its own in practice. This is simply because not all images can generate reliable tokens. Besides this, the interactive segmentation algorithm is still not mature. However, no matter it is hard integration or soft integration, we believe that there still exists a large room for improvement. Future works will try to design new

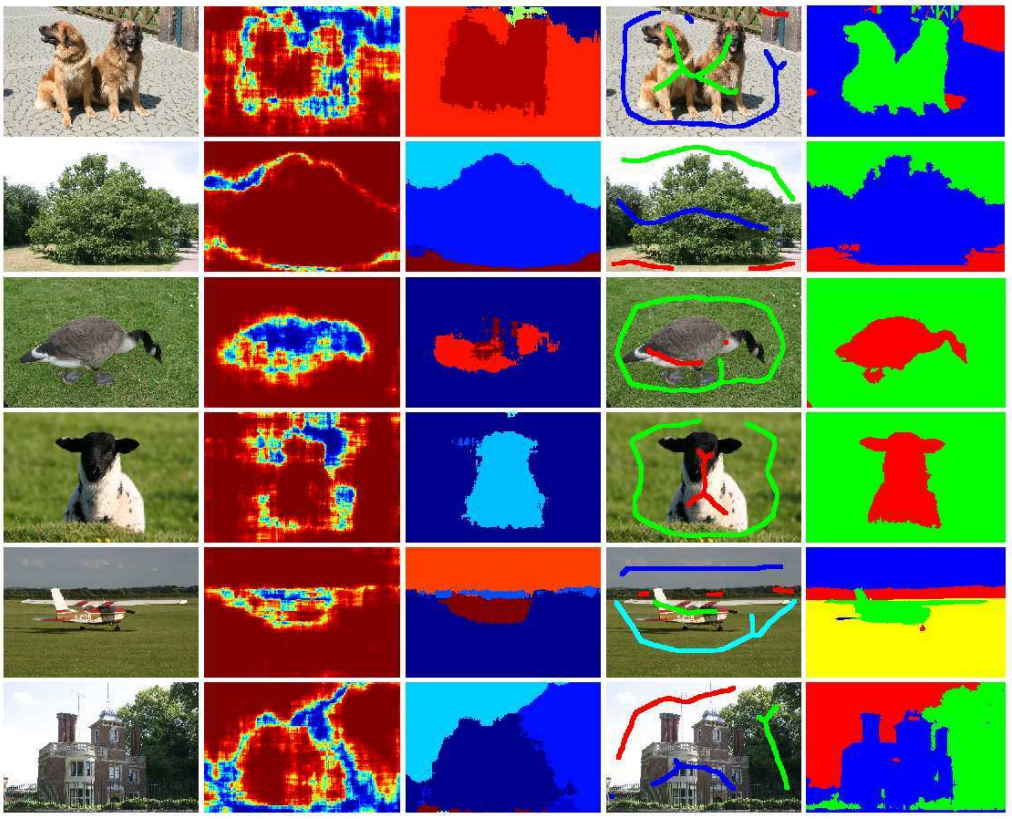

FIGURE 4.9: From left to right: original image; outputs from the semantic segmentation algorithm; the predicted semantic class; automatically generated tokens; segmentation results generated by [114] based on the tokens.

frameworks to make the semantic segmentation module and low-level segmentation module really 'interact' with each other.

4.5 Concluding Remarks

In this chapter, we first experimentally highlighted the importance of choosing regions as processing primitives in pixel-level semantic image understanding. To produce the object-consistent regions, we then propose to unify the state of the art low-level segmentation algorithms with typical semantic segmentation

algorithms by introducing the semantic feature feedback. Experiments on MSRC21 and the stanford background dataset have confirmed the effectiveness of our new method.

There exist some limitations in our model. For example, we are implicitly requiring the semantic features should have a relatively high accuracy, otherwise it will deteriorate the performance of the whole system. Another problem is how to automatically decide the number of objects contained in an image. In the illustrative example of Fig.4.1, the appropriate number of regions should be 4. The Dirichlet Process Mixture Model[4] may be a promising tool to tackle this problem.

We also noticed that there are some recent works [115, 116] that are quite similar to ours. In [115], the authors try to obtain semantic contours (which will result in semantic consistent regions) by utilizing the results from object detectors. [116] performed the low-level over-segmentation first, and then extract multiple kinds of semantic features from each region. These semantic features together with the low-level regional features are then fed to a one-vs-all SVM classifier to perform the final semantic segmentation. By looking at these works together with our own work, we believe that it is the recent trend that researchers tend to combine different tasks together and try to solve them simultaneously. For example, in [117], the authors integrated the task of surface estimation, depth estimation, occlusion boundary estimation and object detection into one framework. Each module is iteratively updated using the output of other modules. A similar idea is also adopted by Heitz [118] who proposed the Cascaded Classification Model (CCM) which integrates the problem of object detection, region labeling and geometric reasoning. Another notable work is recently proposed in [119], where object detection and semantic segmentation are novelly integrated in a CRF framework.

[4] http://people.csail.mit.edu/jacobe/software/dpmm.tar.gz

Chapter 5

GlandVision: A Novel Polar Space Random Field Model for Glandular Biological Structure Detection

Just as shown in the previous chapter, we believe that the segment-then-recognize strategy is a more suitable choice in pixel-level semantic image understanding, as long as we can design an appropriate segmentation algorithm which can produce regions that are more object-consistent. In this chapter, we studied a problem in medical imaging named gland recognition. We follow the segment-then-recognize strategy, and found that this strategy works very well in this scenario, as long as we can devise a suitable algorithm which can produce regions resembling the glands. More specifically, we first transform the image from Cartesian space to polar space and introduce a novel random field model with an efficient inference strategy that uses two simple chain graphs to approximate a circular graph to infer possible gland boundaries. We then develop a visual feature based support vector regressor (SVR) to verify if the inferred contour corresponds to a gland. We then combine the outputs of the random field and the regressor to form the GlandVision algorithm for the detection of glandular structures. In the

experiments, we treat the task as object (gland) proposal, detection and segmentation problems respectively and show that our new technique outperforms state of the art computer vision algorithms developed for these tasks.

This chapter is structured as follows: in section 5.1 we introduce some background for gland detection and we briefly review related literature in section 5.2. We present our new polar space random field model and a novel efficient inference solution in section 5.3. In section 5.4 we introduce a support vector regression model to verify the potential glandular structures. In section 5.5 we present our overall glandular structure detection method and section 5.6 presents extensive experimental results and comparison with state of the art.

5.1 Introduction

Tissue diagnosis is an important part of modern day medicine. Where disease is suspected, tissue samples can be taken from the patient and viewed under the microscope by a Pathologist. In many human tissues, cells are organized into complex anatomical units called *glands*. In many disease states the glands are disrupted, often in a characteristic fashion. If automated image analysis is to be used to facilitate tissue diagnosis, then recognition of glands is essential.

In this study we sought to devise an algorithm for the automated detection of glandular structures in human tissues. A typical microscopic image of the human colon and the glands contained in it are shown in Fig.5.1. It can be seen there that a gland is composed of a group of cells who sit side-by-side and form the boundaries. Depending on the way the tissue has been sectioned, the shape of a gland can vary hugely and this poses significant challenge to computational algorithms for automatic gland detection.

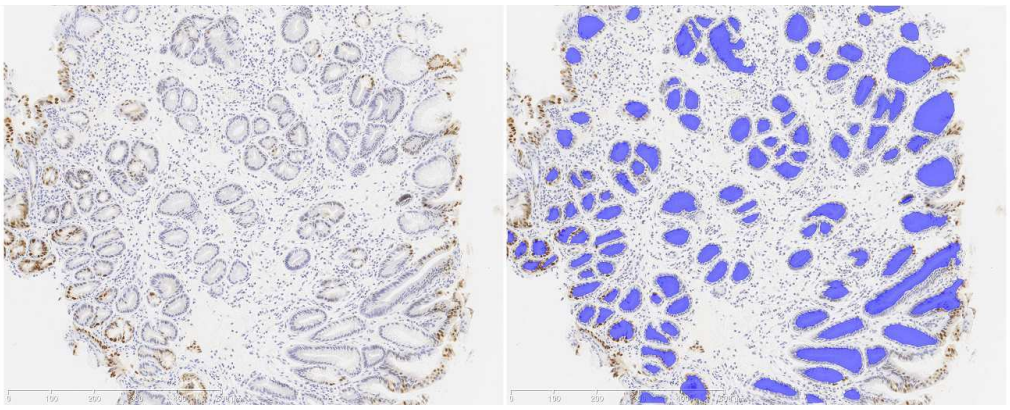

FIGURE 5.1: A microscopic image of human colon tissue containing glands. The glands are manually annotated in blue solid color as shown in the right image.

5.2 Related Work

Since gland exhibits irregular shapes on tissue sections, most of the previous works tackle the gland detection problem by focusing on *lumen* detection. Lumen is the region in the center of a gland. Previous methods assume that lumen can be identified by their color [120, 121] or texture [122, 123]. Once the lumen area is detected, it is considered as the seed region, and some segmentation methods, such as region growing [124], active contour [125], level-set [126], etc, can be adopted to segment out the glands.

Because color can vary significantly in different microscopic images, these methods sometimes will require human intervention [125] to label the lumen regions for model construction. In contrast, our method is totally color-free and processes the grey-scale image. Besides this, we also believe that texture is not an informative cue in this kind of image as can be seen in our experimental section 5.6.3 where a state-of-art semantic segmentation algorithm which utilizes texture features can only obtain a relatively low performance.

5.3 A Novel Random Field Model for Gland Proposal

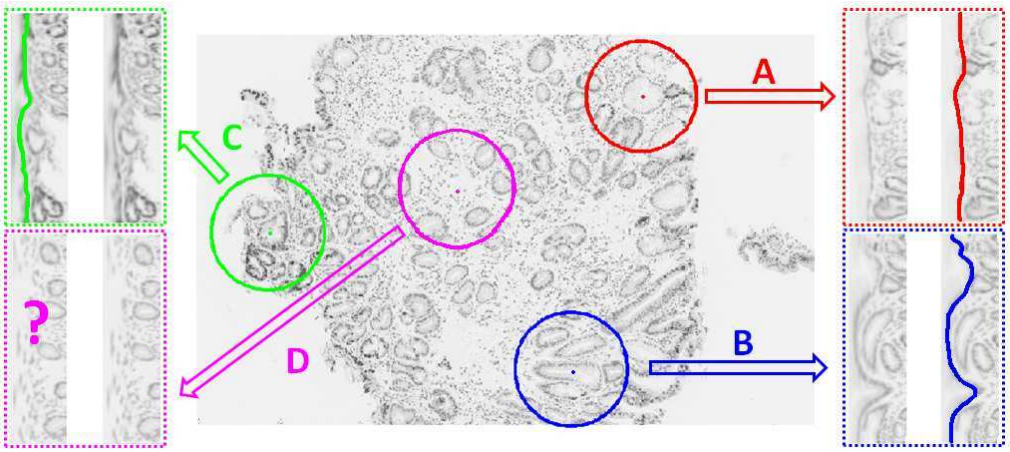

FIGURE 5.2: For the polar image, its horizontal axes corresponds to the distance to the origin, while the vertical axes corresponds to the angle ranges from 1 to 360. A, B and C correspond to cases where the polar space's origin is inside a gland while D corresponds to the case where the polar space's origin is not inside a gland. We can clearly see a continuous line structure in A, B and C, while this kind of structure cannot be seen in D.

One of the most distinctive properties of a gland is that they usually exhibit a closed shape structure. If we place our viewpoint inside the gland, we will see a closed contour, which means if we place the co-ordinate's origin inside a gland and transform the gland to the polar space, we will see a continuous line structure along vertical direction in the polar image. Some examples are shown in Fig.5.2. It is seen that if the origin is inside a gland, we can see an obvious line structure in their corresponding polar image (e.g., regions circled as A, B, and C); if the origin is outside a gland there is no such line structure in its polar image (e.g., the region circled as D). Based on this observation, the problem of detecting glands can be formulated as the problem of detecting those line structures in the polar image. In this section, we have developed a novel Conditional Random Field (CRF) model to achieve this.

5.3.1 A Novel Random Field in the Polar Space

At each point (x_0, y_0) in the Cartesian space, we crop a sub-image with (x_0, y_0) at its center. A point (x, y) in the sub-image is transformed to the polar space (r, θ) using:

$$\begin{cases} r = \sqrt{(x - x_0)^2 + (y - y_0)^2} \\ \theta = \arctan \frac{y - y_0}{x - x_0} \end{cases} \tag{5.1}$$

In this work, θ is discretized to 360 units corresponding to 360 degrees. For a practical system, we only need to consider a limited range of $r \in (1, r_{max})$, which means that we assume the diameter of the maximum size of the gland is $2 * r_{max}$ (to detect larger glands, we can down-scale the image and still use the same r_{max} value). After this transformation, a circular region with radius of r_{max} in the original image is transformed to a fixed size polar image of 360 rows $\times$ r_{max}-columns.

To detect the gland contours in the polar image, we developed a Conditional Random Field (CRF) model [57]. A CRF formally consists of a random variable $\mathbf{X}$ over the observed data and a set of random variables $\mathbf{Y} = \{Y_1, Y_2, ..., Y_n\}$ over the labels to be inferred. All components Y_i of $\mathbf{Y}$ are usually assumed to range over a finite label alphabet $\mathcal{Y}$. In our case, we assign each row of the polar image a label Y_i, which indicates the position of the gland contour at each row. Accordingly, the label alphabet $\mathcal{Y}$ equals to $\{1, 2, ..., r_{max}\}$. The graphical model of our CRF contains only 360 nodes in total and is illustrated in Fig.5.3(a).

The energy function of our CRF consists of two terms: the unary potential ψ_i and the pairwise potential ψ_{ij}. It is formally defined as:

$$E = \sum_i \psi_i(Y_i \mid \mathbf{X}) + \sum_{i,j} \psi_{ij}(Y_i, Y_j \mid \mathbf{X}) \tag{5.2}$$

where $\mathbf{X}$ represents the polar image data. It can be represented as $\mathbf{X} = \{X_1, X_2, ..., X_{360}\}$, where X_i corresponds to the ith row in the polar image, as illustrated in Fig.5.3(b).

We then made two further assumptions about the unary potential and the pairwise potential. For the unary potential $\psi_i(Y_i \mid \mathbf{X})$, we assume it only depends on X_i instead of the whole $\mathbf{X}$, and we assume the pairwise potential $\psi_{ij}(Y_i, Y_j \mid \mathbf{X})$ only depends on X_i and X_j. Thus our energy function can be re-written as:

$$E = \sum_i \psi_i(Y_i \mid X_i) + \sum_{i,j} \psi_{ij}(Y_i, Y_j \mid X_i, X_j) \tag{5.3}$$

The factor graph of our CRF is shown in Fig.5.3(c). The definition of our unary potential and the pairwise potential will be described in section 5.3.3 and section 5.3.4.

We also noticed that there exists previous works that also utilize the polar image [127] or the polar property [128] and adopted a random field model for image segmentation. However, the structure of their random field is quite different from ours. In their work, they assign each pixel a random variable which can take two labels corresponding to 'inside' and 'outside'. For a polar image of 360 rows $\times$ r_{max}-columns, their definition will produce $2^{360*r_{max}} = (2^{r_{max}})^{360}$ possible states, whilst our definition will only produce $(r_{max})^{360}$ states, which is much smaller than theirs. Besides, our structure of the random field automatically satisfy the star-convexity property which is considered as an extra constraint in [128].

5.3.2 Inference

The inference procedure of our CRF is to find the optimal $\mathbf{Y}$ that can achieve the lowest energy E. From the graph shown in Fig.5.3(a), we could see that it contains a loop structure, which means it is very time consuming, if not impossible, to perform exact inference, and we had to resort to approximate inference

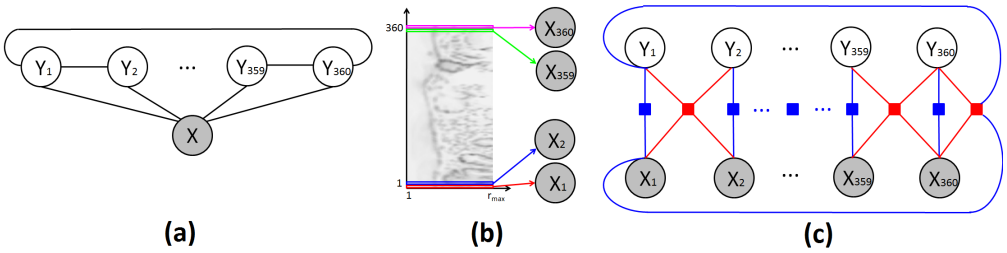

FIGURE 5.3: (a) The graphical model of our CRF model; (b) Each row of the polar image is assigned a random variable; (c) The factor graph of our CRF model.

methods, such as loopy belief propagation [129] which is known to be computationally intensive. We noticed that the graph structure in our scenario is a loop structure only if it contains one more edge which links Y_1 and Y_{360}, otherwise it will be a chain, and there exists very efficient inference methods, such as dynamic programming, for chain structure. To avoid the influence of this extra edge, we have developed a novel efficient solution that uses two chain structures to approximate this circulate graph. More specifically, we generate two polar images $I(r, \theta)$, one with a θ ranges from 0 to 2π, the other with a θ ranges from π to 3π. This is shown in Fig.5.4(a). For these two graphs, we do not connect their heads and tails; hence they are just two chain graphs. We use the well known Viterbi algorithm for inference in this chain structure. The inference is performed separately on each of these two graphs. Finally, we use Algorithm 2 to combine the two results together.

This 'divide and merge' approach works extremely well in practice, and it only takes a few milliseconds on a 360 $\times$ 100 pixel polar image.

5.3.3 Unary Potential

Now we define the unary potential $\psi_i(Y_i \mid X_i)$ appeared in our energy function. We noticed that the gland's boundary usually corresponds to the darker area of the image (see Fig.5.1 for example). Therefore, a natural strategy is to use the

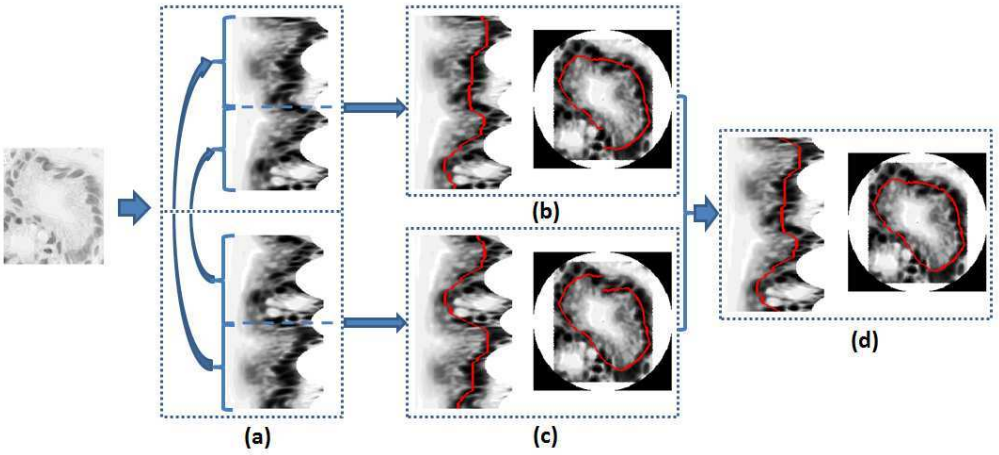

FIGURE 5.4: An Efficient Inference Strategy. (a) An image is transformed into two polar images, one's θ ranges from 0 to 2π, and the other's ranges from π to 3π. The Viterbi inference algorithm is performed separately on these two polar images, and the results are shown in (b) and (c). We can see that the resulted contour is not a closed shape. This is because we use the chain structure to approximate the circular graph. These two results are then combined using Algorithm 2 to generate the final result shown in (d).

pixel intensity as the unary potential. Let $I(r, \theta_i)$ be the grey value at position (r, θ_i) in the polar image, then a simple definition of the unary potential can be written as:

$$\psi_i(Y_i = r) = I(r, \theta_i) \tag{5.4}$$

As illustrated in Fig.5.5(c), such simplistic definition is flawed and produces lots of noise. To make the unary potential more informative, we make another stronger assumption that the boundary position prefers a smaller r value. In other words, we prefer the inner contour than the outer contour. This assumption is reasonable because the inner part of a gland is usually in lighter colors. To achieve this, the polar image is first filtered with a vertical edge filter [-1, 0, 1]. Then we calculate the cumulative sum of the edge image in the horizontal direction, and the values in each line of this cumulative edge map are normalized to

Algorithm 2 Gland Detection

Input: gland image;
Output: gland $Contour$;
Transform the gland image to two polar images: $\theta \in (0, 2\pi)$ and $\theta \in (\pi, 3\pi)$;

$Contour1$ = perform inference on the polar image with $\theta \in (0, 2\pi)$;
if $|Contour1(1) - Contour1(end)| < 5$ **then**
  $Contour = Contour1$;
else
  $Contour2$ = perform inference on the polar image with $\theta \in (\pi, 3\pi)$;
  if $|Contour2(1) - Contour2(end)| < 5$ **then**
    $Contour = Contour2$;
  else
    $Shift\_Contour2(1:180) = Contour2(181:360)$;
    $Shift\_Contour2(181:360) = Contour2(1:180)$;
    $Difference = |Contour1 - Shift\_Contour2|$;
    $[dummy, index1] = \min(Difference(5:180))$;
    $[dummy, index2] = \min(Difference(181:355))$;
    $Contour = Shift\_Contour2$;
    $Contour(index1 + 4 : index2 + 180) = Contour1(index1 + 4 : index2 + 180)$;
  end if
end if
Return $Contour$;

(0,1). Our new unary potential is defined as:

$$\psi_i(Y_i = r) = 1 - (1 - I(r, \theta_i)) * (1 - I_{cumedge}(r, \theta_i))^{\alpha} \tag{5.5}$$

where α is the trade-off parameter between the two terms. Note that only when $I(r, \theta_i)$ and $I_{cumedge}(r, \theta_i)$ both approaches 0, $\psi_i(Y_i = r)$ approaches 0. If either of these two terms approaches 1, $\psi_i(Y_i = r)$ will approach 1. We empirically set $\alpha = 0.5$ in all our experiments. The complete procedure of generating the unary potential is shown in Fig.5.5.

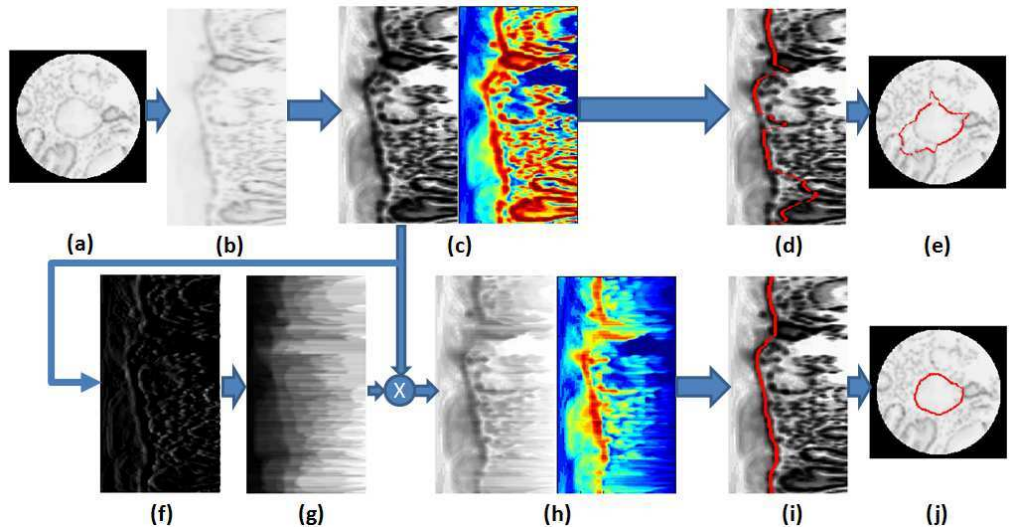

FIGURE 5.5: Illustration of Unary Potential. Given an image as shown in (a), it is firstly transformed into the polar space (b). The histogram equalized polar image is shown in (c) left, and the unary potential as defined in (5.4) is shown in (c) right in pseudo color. This kind of unary potential definition is flawed which will result in the bad detection result as shown in (d) and (e). Instead, we first calculate the cumulative edge map (g) based on the edge map (f). Then this cumulative edge map is combined with the original polar image according to (5.5) to generate our new unary potential shown in (h). Based on this unary potential, we obtain the correct results as shown in (i) and (j).

5.3.4 Pairwise Potential

As shown in Fig.5.3(a), edges only exist among neighbouring nodes. In the simplest case, we can adopt the conventional Gaussian edge potential:

$$\psi_{ij}(Y_i = r_i, Y_j = r_j) = 1 - \exp\left(-\frac{|r_i - r_j|^2}{2\sigma^2}\right) \tag{5.6}$$

where σ is the standard deviation. In practice, we found it is always difficult to set up a suitable σ value. That's one of the reasons why we need to adopt a CRF model instead of a Markov Random Field (MRF) model[1]. On defining the pairwise potential, we adopt similar idea from the contrast sensitive Potts model [11]. We draw a line between (r_i, θ_i) and (r_j, θ_j), and we integrate all the

[1]Please refer to [130] for a more detailed description of the differences between MRF and CRF.

intensity values in this line:

$$I_{sum} = \int_{(r_i,\theta_i)}^{(r_j,\theta_j)} I(r,\theta)\,\mathrm{d}(r,\theta), \qquad (r,\theta) \in line_{(r_i,\theta_i)}^{(r_j,\theta_j)} \tag{5.7}$$

Then we calculate the average intensity along this line $I_{avg} = I_{sum}/length$. Our new data dependent pairwise potential is defined as:

$$\psi_{ij}(Y_i = r_i, Y_j = r_j) = 1 - \exp\left(-\frac{|r_i - r_j|^2}{f(I_{avg})}\right) \tag{5.8}$$

where $f(I_{avg})$ is a function over I_{avg}. It is formally defined as:

$$f(I_{avg}) = c * (\max(1 - I_{avg}, thresh))^{\beta} \tag{5.9}$$

where c, $thresh$ and β are all constants. From the above definition of $f(I_{avg})$, we can see that when I_{avg} approaches 0, $f(I_{avg})$ approaches c, and when I_{avg} approaches $1-thresh$, $f(I_{avg})$ approaches $c*thresh^{\beta}$. Although we can utilize the training data to learn these parameters, we didn't find the learning procedure very helpful. We empirically set $c = 80$, $thresh = 0.5$ and $\beta = 4$ in all our experiments.

5.4 Verification by a Visual Feature based Regressor

Based on the above gland proposal model, we adopt a sliding window style detection strategy on a given whole image. As our random field model directly searches the radius at each angle, and the radius ranges from 1 to r_{max}, it means our algorithm can detect glands whose radius can vary from 1 to r_{max}. Therefore, we do not need to slide the detecting windows at different scales. Instead,

we only need to apply our detection model at each position once. In practice, we sub-sample the pixel locations by 10 pixels, and only apply our model at these positions. In this way, for an image whose size is 1000 pixel width and 1000 pixel height, we will get around 10000 potential gland contours.

The possible gland contours detected by the above random field model still contain much noise, because a true glandular structure is not only defined by its contour but also the structure inside the contour. Therefore, we will design a visual feature based verification module to judge if the contours detected correspond to true glands. We firstly use the energy output of the random field to sort all the potential gland contours. Then a threshold T is set to remove those obvious non-glands. We set T to be a relatively large value in order to maintain a high recall rate. We didn't do much engineering work on the choice of this T value, and T is set to be the same value for all the training and testing images. On average, we will retain about 4000 contours among all the 10000 detected contours for each image. For each of these remaining contours, we extract the PHOG visual feature [131] from the smallest bounding box enclosing it. We also compute a score to measures its quality. This score is defined as the maximum overlap between the detected contour S and any ground truth glands S_i:

$$score = \max_i \frac{|S \bigcap S_i|}{|S \bigcup S_i|} \tag{5.10}$$

We train a regressor to regress this score using the PHOG features. Recent research have shown that regression is sometimes more suitable than classification for object recognition [132, 133]. We utilize the popular LIBSVM toolbox [134] to learn a nonlinear $\epsilon - SVR$. The pyramid match kernel defined in [131] is used to generate the kernel matrix. In training the nonlinear kernel SVR, we also adopt the 'mining hard negative' strategy [61, 133] to incrementally increase

the training set. In each step, we only retain the support vectors and add those samples with the largest regression errors.

5.5 GlandVision: Integrating Random Field with Regressor

To perform the final gland detection, we fuse the outputs of the random field and the regressor using a classifier level fusion paradigm [29]. The random field energy output and the regressor output both have probabilistic meanings. Previous work [29] has shown that adding classifier outputs is a more robust way to fuse two probabilistic outputs. We directly sum the two outputs together. Besides, we have also tried our approach proposed in Chapter 3 in combining these two outputs. The flowchart of our complete gland detection algorithm is depicted in Fig.5.6.

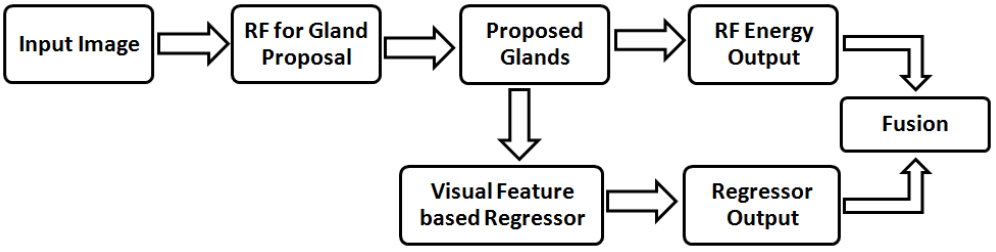

FIGURE 5.6: A Complete GlandVision Procedure

5.6 Experiments

We have collected a dataset consisting of 20 high resolution 1280 $\times$ 1024 pixel microscopic images of human colon tissues. The dataset contains 1072 glands and all have been manually labelled with pixel accuracy. Fig.5.7 shows all the 20 images in this dataset. The dataset and ground truth data has now

been made publicly available at *http://www.viplab.cs.nott.ac.uk/download/Nott-Gland.html*. In the experiments, we randomly chose half of these 20 images for training and the rest for testing. As our dataset contains pixel-level ground truth, we can perform different tasks on this dataset. In the following, we will consider three scenarios: Gland Proposal, Gland Detection and Gland Segmentation.

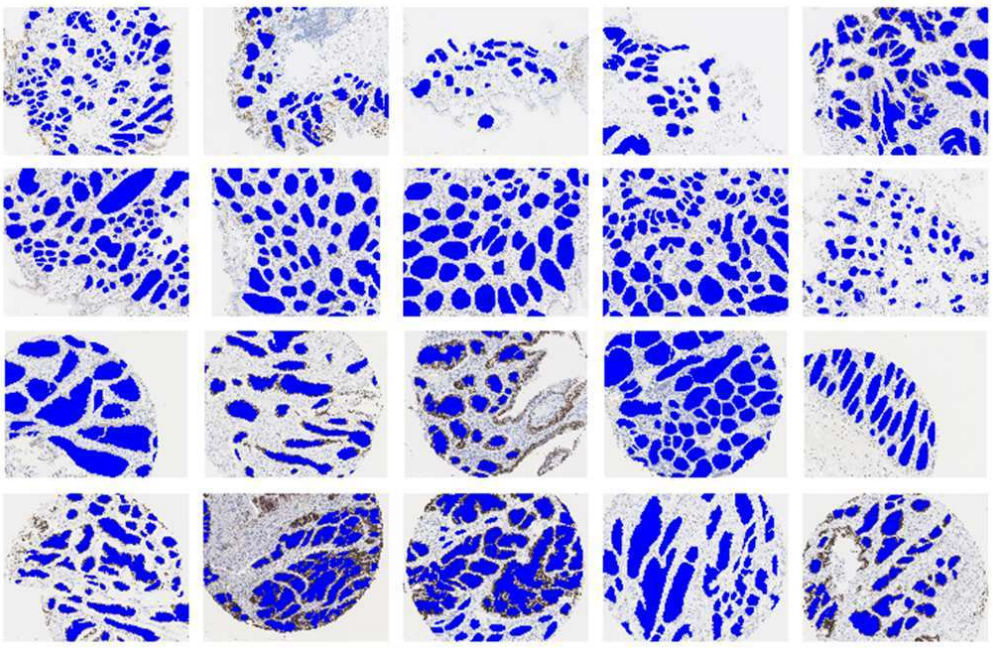

FIGURE 5.7: Nott-Gland dataset.

5.6.1 Gland Proposal Accuracy

It is not until recently that the term of object proposal [69, 135, 136] have attracted researcher's attention which either advocates the segment-then-recognize strategy [69] or can facilitate the sliding window object detector [135]. In this theme, our random field model can also be considered as an object (gland) proposal algorithm. In the first experiment, we consider the problem as a gland proposal problem and evaluate the object proposal accuracy of our algorithm against one of the state-of-art algorithms in object proposal [135]. Given an image, the task of object proposal is to return a ranked list of bounding boxes

that are most likely to contain an object. The performance is evaluated using the Detection-Rate/Signal-To-Noise curve [135]. For the object proposal algorithm in [135], we directly use their public code. The parameters in their method are learned using our training images.

A proposal whose overlap score, as defined in (5.10), exceeds a predefined threshold is considered as a true positive. A common setting is to set the threshold to be 0.5, which is the standard PASCAL criterion [70]. Besides this, we have also considered to set it to 0.8. A higher threshold means that the proposed object and the ground truth must have a higher degree of overlap hence indicating a higher standard to be considered as true positive.

The results we have achieved together with the objectness method [135] are shown in Fig.5.8. From there we can see that our method outperforms objectness by a large margin. This comparison is perhaps not fair, as objectness is designed to detect generic objects, while our random field model is specifically designed for gland proposal. We show the result of objectness here to illustrate that gland proposal is a very distinct problem from generic object proposal, and a method that works well for generic object proposal does not necessarily work well in our scenario. Besides this, we could also tell from the figure that our method can achieve a high detection rate (recall rate), even the overlap threshold is set to 0.8, which means our random field model can accurately propose the glands.

5.6.2 Gland Detection Accuracy

We now consider the problem as a gland detection problem. For the baseline method, we have implemented the part-based object detection model [61] using their public code. This method is considered to be the state of the art method in object detection literature.

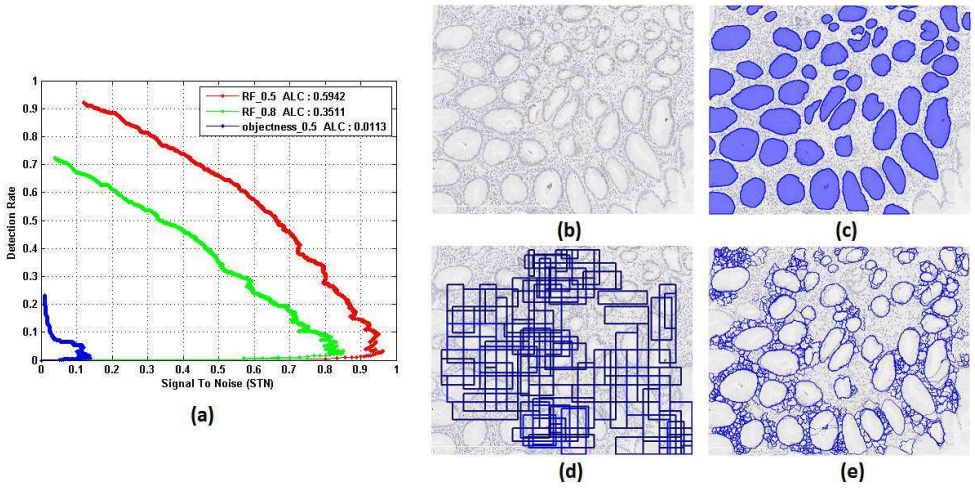

FIGURE 5.8: (a) The DR/STN curve of [135] and our Random Field (RF) model at threshold 0.5 (RF_0.5) and 0.8 (RF_0.8). (b) The original microscopic image. (c) The ground truth image of glands. (d) The results obtained by [135]. (e) The results obtained by our full model.

For the evaluation criterion, we adopt the Mean Average Precision (MAP) which is the standard PASCAL criterion. Fig.5.9(a) shows the detection results achieved by our methods together with results of part-based object detection model. On their own, our Random Field and phog_SVR methods achieve a MAP of 0.50 and 0.54 respectively, which is lower than the results obtained by part-based model. However, when we combine these two methods by simply summing their outputs together, our result exceeds part-based model. Besides this, if we adopt our technique proposed in Chapter 3 to combine these two outputs, our result is further improved. As can be seen from Fig.5.9(b) and (c), the value distribution of 'RF' output and 'phog_SVR' output differs a lot, and it will make more sense if we transform their distribution to be the same before summing them together. We directly use our histogram matching code which is publicly available[2].

Although our method obtains a higher MAP than the part-based model, the advantage is not obvious. A significant advantage of our model is that it can not only detect the glands, but can also **accurately** localize them. Therefore, instead

[2]http://www.viplab.cs.nott.ac.uk/demo&code/FeatureCombination/Feature%20combination.html

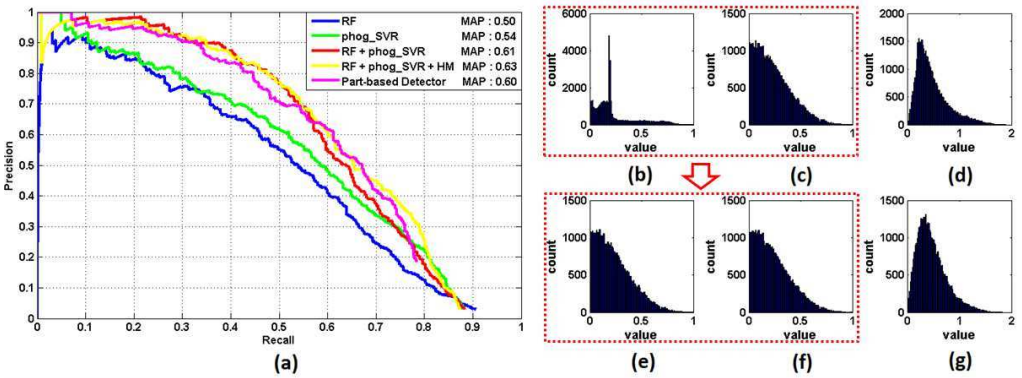

FIGURE 5.9: (a) Precision-Recall curve of different methods. (b)-(d) The histogram of original phog_SVR values, the Random Field (RF) outputs and their summed results. (e)-(g) The histogram of the phog_SVR values, RF outputs and summed value after Histogram Matching (HM) (please refer to Chapter 3). We could see that the histogram of phog_SVR and RF becomes the same and the HM operator.

of setting the overlap threshold to be 0.5, we set it to higher values. Note that a higher threshold means higher degree of overlapping between the detected objects and ground truth hence requiring more accurate localization. We can see from Fig.5.10 that for the part-based detection model, as the threshold increases, MAP drops significantly, whilst for our methods, the drop is not that obvious. When the threshold is set to 0.8, the MAP of part-based detection model drops to 0.02, which means it can hardly detect any glands. But still, our methods can achieve a relatively much higher MAP. Some qualitative examples comparing part-based detection model and our method are shown in Fig.5.11.

5.6.3 Gland Segmentation Accuracy

Just as shown in the previous experiments, our model can accurately localize the glands, which means it can be considered as a segmentation algorithm. In this section, we treat the problem as a semantic segmentation problem, and adopt the PASCAL VOC score $= \frac{TP}{TP+FP+FN}$ as the evaluation criterion. There are in total two semantic classes: 'gland' and 'background'. For the baseline method,

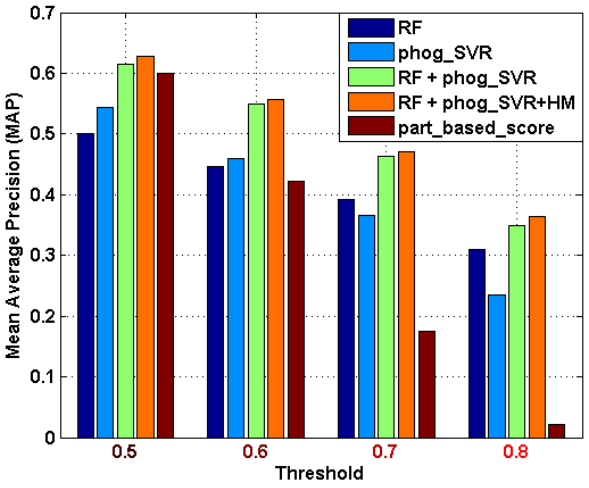

FIGURE 5.10: MAP value at different thresholds.

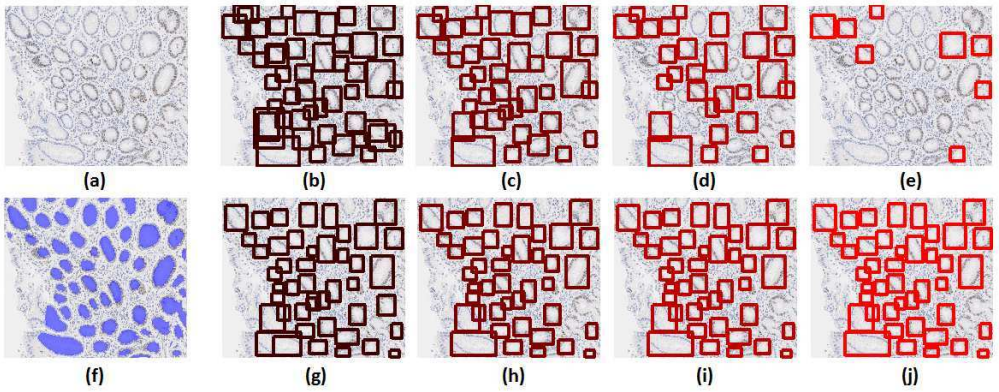

FIGURE 5.11: (a) The original microscopic image. (f) The ground truth image of glands. (b)-(e) The detection results obtained by [61] with threshold varies from 0.5 to 0.8 (g)-(j) The results obtained by by our model. As the threshold increases, the bounding boxes have to be more accurate. As the accuracy requirements increases (higher threshold), the detection result of [61] dropped significantly.

we implemented the hierarchical CRF model [12] using their public code. This method build a hierarchical graph among pixels and superpixels, and has obtained the best result on some famous semantic segmentation datasets including MSRC21 [9].

For our segmentation methods, we simply assign all the pixels contained in a gland proposal to be 'gland' class. Here we need to set a hard threshold to judge if a gland proposal is true positive. For the phog_SVR method, we can simply

select 0.5, as this model is trained to have an output ranges from 0 to 1, but for other models, it is difficult to select an appropriate threshold. Therefore, we count the number of gland proposals whose phog_SVR score is above 0.5, and chose a threshold that retain the same number of proposals for other methods.

TABLE 5.1: Segmentation Accuracy

	RF	phog_SVR	RF+phog_SVR	RF+phog_SVR+HM	[12]
gland	0.521	0.588	0.601	**0.615**	0.119
background	0.845	0.828	0.805	**0.850**	0.785
average	0.663	0.708	0.723	**0.732**	0.452

Table.5.1 shows the results obtained by our methods and the method proposed in [12]. We can see that [12] can only obtain an accuracy of 0.119 for the 'gland' class, which means it is very difficult to distinguish the 'gland' class from the 'background' class only based on the features extracted from the pixel or superpixel level. In fact, the inner region contained in a gland is exactly the same as the regions between glands. It is the circular structure that our model tries to capture that separates the 'gland' class from the 'background' class. An example output of [12] and our method is shown in Fig.5.12.

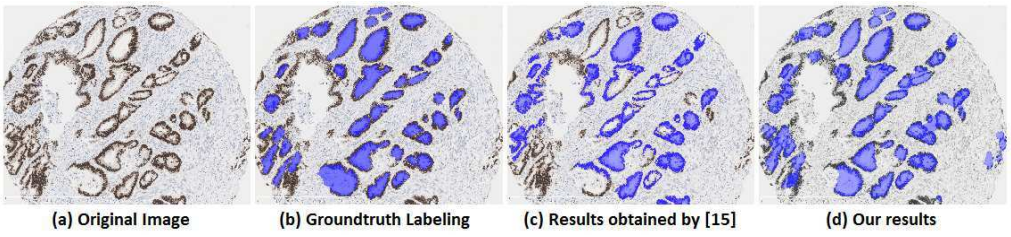

FIGURE 5.12: Segmentation results.

5.7 Concluding Remarks

In summary, we have presented a novel method for detecting glandular structures in microscopic images of human colon tissues in this chapter. We consider

the problem at hand from three different angles, i.e. object proposal, object detection and semantic segmentation, and compare our algorithm with several state of the art methods in the respective fields. Different evaluation criteria have confirmed the effectiveness of our method.

Our method has some limitations. Firstly, due to the utility of the polar image and the structure of the random field model, our method can only detect star-convexity shapes [128]. Future works will try to overcome this limitation. Secondly, the regressor is trained only based on the PHOG feature, it is anticipated that adding more features could boost the performance of the regressor as well as our final model. Thirdly, the glands contained in a microscopic image will never overlap, but we haven't utilized this property in this work. Future works will try to utilize this property and obtain a global consistent result.

Chapter 6

Random Forest for Image Annotation and Retrieval

In the previous two chapters, we have studied problems related to pixel-level semantic image understanding. However, just as mentioned in Chapter 2, the most state of the art pixel level image understanding algorithm can only obtain an accuracy of around 43% on a challenging dataset, which is far from satisfaction. In contrast, there already exists several commercial image search engines, such as google image. The task of image-level semantic image understanding seems to be easier than pixel-level understanding, while it is still of significant piratical meanings. Furthermore, just as some previous works [65, 137] showing that, the results of image-level understanding could be used as a prior and boost the performance of pixel-level understanding.

In this chapter, we have developed a novel usage of random forest for large-scale image-level semantic image understanding. The nature of the random forest structure serves as an efficient data structure for storing and fast browsing the image data. Besides, in generating the random forest, we can effectively utilize the tag information associated with images to tackle the semantic gap problem.

Our random forest differs from traditional usage, which consider each random tree as independent from each other, in that we treat the random forest *as a whole*, and have developed two new concepts named Semantic Nearest Neighbor (SNN) and Semantic Similarity Measure (SSM). These two new concepts enable us to develop novel solutions for the task of image retrieval and image annotation.

In this chapter, we will first introduce some motivations for the usage of random forest in section 6.1. Then we introduce our construction of the random forest and propose the two new concepts of SNN and SSM in section 6.2. In section 6.3 and 6.4, we describe our usage of these two concepts in dealing with the task of image retrieval and image annotation respectively. Experimental results on four datasets will be shown in section 6.5 and some concluding remarks will be given in section 6.6.

6.1 Introduction

We have mentioned in Chapter 2 that Nearest Neighbor (NN) based methods, although simple, have shown its usefulness in many computer vision applications. In fact, as long as you have a sufficiently large repository of images together with human labeled ground truth, then the simple NN method might be able to solve the image understanding problem. The rationale behind this is that we can always find very similar images to the target image when the database is sufficiently large. This idea is illustrated in Fig.6.1. We can see that the nearest neighbors retrieved from a relatively small dataset (contains 2000 images) is not quite similar to the query image. However, when the dataset becomes large (contains 2,000,000 images), the images retrieved look much more similar to the query image. Some researchers [44] named this method brute-force image understanding.

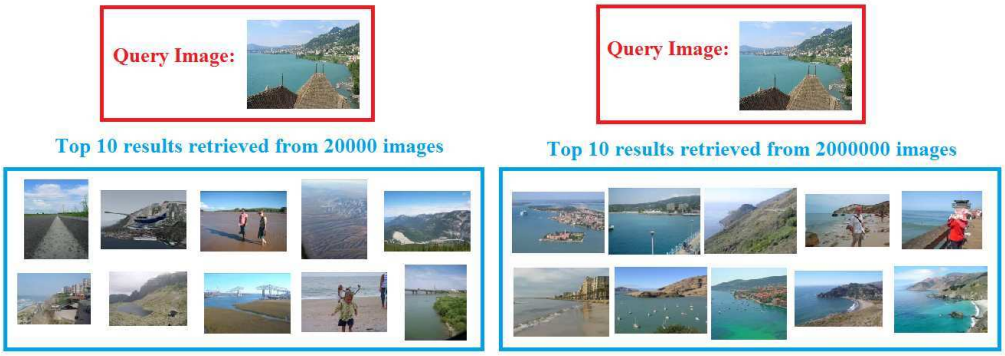

FIGURE 6.1: Illustrative example on the effect for the size of the dataset. This example is adapted from [44].

However, if we want to extend NN methods to deal with large-scale datasets, there are at least two technical issues: the first one is how to design efficient data structures to store and retrieve the nearest neighbors. The second is the semantic gap problem where the nearest neighbors retrieved based on visual feature similarity do not necessarily share the same semantic concepts. Just as we pointed out in Chapter 2, NN is in essence an unsupervised method.

In this chapter, we propose to use random forest to tackle these two problems encountered by NN simultaneously. On the one hand, using the tree structure of the random forest enables the efficient retrieval of nearest neighbors; on the other hand, we can utilize the tag information associated with images as supervising information to guide the generation of the random trees, thus making the images located at the same leaf node share similar semantic concepts.

We now describe our usage of random forest, and show how to use it to tackle image retrieval and image annotation problems.

6.2 The Construction of the Random Forest

It has been common practice [1, 138, 139] to use multiple kinds of features to represent one image. How to efficiently combine different features is a non-trivial task. Existing methods include using an equal weight for different features [1], a distance specific weight for each feature [46], or learn an annotator for each feature and then fuse the outputs of different annotators [139].

In our random tree scenario, the method of fusing different features is correlated with the choice of the split function of the random tree. To narrow down the search space for the split function, we prefer to use a simple split function while simultaneously trying to maintain its discriminative power. To achieve this, we propose to use dimension reduction methods [140] to get a more compact representation for each feature channel. More specifically, we choose to use kernel PCA [141] for each feature channel. Although directly computing kernel PCA is time consuming and hard to extend to large scale, there exist fast approximate methods [142] which make it scalable to large scale datasets. In all our experiments, we use kernel PCA to reduce each feature dimension to a fixed low dimension, thus making each kind of feature have an equal probability to be chosen as the dimension on which the split function will operate. Denote the kernel PCA reduced feature as $\mathbf{F}'$, the split function is defined as:

$$\begin{cases} F'_i \geq thresh & \text{go to left child} \\ otherwise, & \text{go to right child} \end{cases} \tag{6.1}$$

Based on the split function defined above, we can split samples to the left child node or right child node accordingly. We can generate multiple splits by choosing different feature dimensions or different thresholds. We plan to use the tag information to guide the generation of the tree. The idea is straightforward: at each node, after splitting the samples to the left node or the right node, we can

compute their corresponding tag histograms. A good split means the tag histogram of the left node should be quite different from the tag histogram of the right node. From the probabilistic point of view, the tag histograms in the left or right child node can be considered as the probabilities that the left child node or the right child node contains the specific tag. Thus we can use the widely adopted information gain criteria [10, 50], which is defined in (2.10) as the score function.

The whole procedure of growing the tree is summarized in Algorithm 3.

Algorithm 3 The growing procedure of a random tree

Input: Feature(N*M); N: feature dimension, M: Number of samples
Output: Left_node_index, Right_node_index
Dims = randomly select n dims from (0,N)
returned_dim,returned_thresh,best_score=0
for dim in Dims **do**
  min_value = min(Feature(dim,:))
  max_value = max(Feature(dim,:))
  diff = max_value-min_value;
  for thresh = randomly generate m thresholds from (min_value+diff/4,max_value−diff/4) **do**
    compute split score according to score function (2.10)
    if $score > best\_score$ **then**
      best_score = score, returned_thresh = thresh, returned_dim = dim;
    end if
  end for
end for
Left_node_index = Feature(returned_dim,:) $\geq$ returned_thresh
Right_node_index = Feature(returned_dim,:) $<$ returned_thresh

For a test image, we pass it through every random tree. It falls from the root node and keeps falling according to the split function until it reaches the leaf node. It is obvious that every node in its falling trajectory contains important information, but for now we only consider the samples stored in the leaf node and call these samples the *semantic neighbors* of the test sample. The semantic neighbors obtained from different trees together make up the *semantic neighbor set*. Based on this semantic neighbor set, we can draw the important rationale

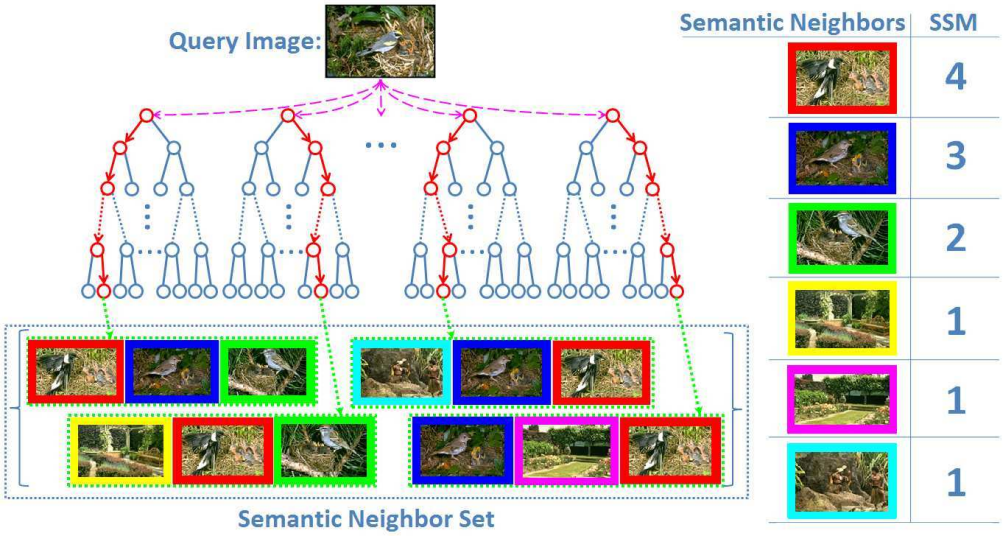

FIGURE 6.2: An example showing the concepts of semantic neighbor set and semantic neighbors. A query image passes through all random trees. The training images stored at the leaf nodes on which the query image falls into form the semantic neighbor set. Based on this, the semantic similarity measure (SSM) between the query and a particular training image is calculated as the number of times that the particular training image appears in the semantic neighbor set. A larger SSM indicates higher similarity.

behind this work: the more often that two images fall into the same leaf node, the more likely they share similar tags. Therefore, the semantic similarity between two images should be monotonically increasing with the number of trees in which they fall into the same leaf node. Thus we can count the number of trees that two images fall into the same leaf node and use this *count* value as the *semantic similarity measure* (SSM) of the two images (we will use *count* value and SSM interchangeably in the rest of this chapter). Based on the SSM, we can sort all the images contained in the semantic neighbor set to retrieve its K semantic nearest neighbors. The concepts of semantic neighbor set and semantic neighbors are also illustrated in Fig.6.2.

6.3 Image Retrieval based on Random Forest

In the realm of image retrieval, there are different ways for a user to query the system, such as uploading an image, using keywords or hand-draw a portrait. In this work, we considered the first two scenarios.

6.3.1 Example-based Retrieval

If the user chose to upload an image, then the image retrieval system should be able to return a ranked list of images that share similar semantic and visual contents. For our random forest, we only need to pass this test image to the forest to obtain its semantic neighbors. Then these semantic neighbors will be ranked according to their SSM values, and those with larger SSM values will be returned to the user. (An example of a ranked list of images according to SSM values is shown on the right of Fig.6.2.)

6.3.2 Keyword-based Retrieval

For the keyword based image retrieval, the user enters a textual word, and the image retrieval system will output images from a large repository which are most likely to contain this word. To perform keyword-based image retrieval, we consider the following settings: suppose a large repository contains two image sets, training and testing: images in the training set are tagged and those in the testing set are not tagged, furthermore, tags in the training set contain all possible keywords that could be asked by the user. Then, given a keyword query, our task is to return a ranked list of images from the testing set that are most likely to contain that specific keyword.

The same as before, we first build a random forest from the training images, and then we pass each image in the testing set to the random forest and obtain its

semantic neighbors. We define the probability that a testing image $\mathbf{I}$ contains tag j as:

$$q_j = \frac{\sum_{i=1}^{K} t_{ij} * c_i}{\sum_{i=1}^{K} c_i} \tag{6.2}$$

where K is the size of the semantic neighbor set of $\mathbf{I}$, t_{ij} is an indicator function which is equal to 1 if tag j exists for the ith semantic neighbor, c_i is the semantic similarity measure (SSM) between the ith semantic neighbor and the testing image $\mathbf{I}$. Based on this equation, we can rank images for a specific tag j according to their q_j values.

6.4 Image Annotation based on Random Forest

6.4.1 Tag Prediction based on Semantic Neighbors

Based on the semantic neighbor set, our semantic nearest neighbor (SNN)-based image annotation method is performed as follows: For a test image, we use the method described above to retrieve its K semantic nearest neighbors. The prediction of the tags for this image totally depends on these K semantic nearest neighbors. At this stage, we can use previously developed methods, like the label transfer method in [1] or the label filter method in [143]. In our case, we can get additional help from the SSM value obtained from our random forest, and therefore we adopt a conventional tf-idf scheme [20].

Denote $\mathbf{I}$ the query image and $\mathbf{Q}$ the probabilities of assigning tags to annotate the image. Let $\mathbf{I_i}$ represents $\mathbf{I}$'s ith semantic neighbor returned by our random forest with its SSM value denoted as c_i, and $\mathbf{T_i}$ represents the ground truth tags of $\mathbf{I_i}$. Suppose there are M tags in total, thus $\mathbf{Q}$ and $\mathbf{T_i}$ can be represented as a M-tuple vector: $\mathbf{Q} = (q_1, q_2, ..., q_M)^T$ and $\mathbf{T_i} = (t_{i1}, t_{i2}, ..., t_{iM})^T$. Here t_{ij} is

an indicator function which is equal to 1 if tag j exists for the ith image. The prediction of $\mathbf{Q}$ totally depends on the $\mathbf{T_i}$ and c_i value:

$$q_j = \log\left(\frac{R}{r_j}\right) * \sum_{i=1}^{K}\left(\frac{t_{ij}}{Z} * f(c_i)\right),\ j \in \{1, 2, ..., M\} \tag{6.3}$$

where the term $\log\frac{R}{r_j}$ is the inverse document frequency [20] obtained from all the training images. R is the number of training images, and r_j is the number of training images which contain the jth tag. Z is a normalizing constant which is equal to $\sum_{i=1}^{K}\sum_{j=1}^{M} t_{ij}$. The term $f(c_i)$ represents a function which should be monotonically increasing with c_i. This term reflects our intention that the neighbor with a larger SSM value should contribute more to the predication of the tags. Based on the computed M-tuple vector $(q_1, q_2, ..., q_M)^T$, we can predict l tags for the test image which correspond to the l largest q_j values.

Possible forms of $f(c_i)$ include $f(c_i) = c_i$, $f(c_i) = {c_i}^2$, etc. However, ad hoc choices of $f(c_i)$ lack predictability. In the next section, we introduce a systematic method to learn $f(c_i)$ from training data.

6.4.2 Image Annotation as Learning to Rank Semantic Neighbors

In conventional random forest literature [52], each tree is considered to be independent and they contribute equally to predict the posterior probability, which is equivalent to setting $f(c_i) = c_i$ in our scenario. However, we will show in our experimental section that sometimes if we choose $f(c_i) = {c_i}^2$, we can obtain a better performance. It is not difficult to understand this. As shown in Fig.6.3, the distribution of the SSM value exhibits a heavy-tailed distribution. That's one of the reasons why the method of JEC [1] is successful by using only the first few nearest neighbors (which correspond to the larger SSM values in our

scenario). It also stimulates our first intuition to use $f(c_i) = {c_i}^2$ to exaggerate the contribution of those large SSM values. However, choosing $f(c_i)$ in an ad hoc manner is not desirable and we need a more systematic approach.

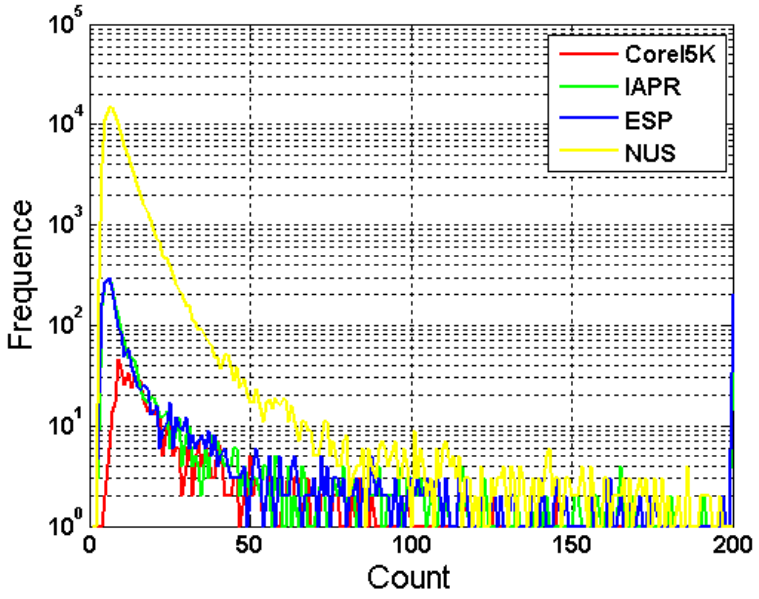

FIGURE 6.3: Distribution of the SSM (i.e. *count*) value across different datasets. Randomly sampled images from each dataset are fed into the random forest. The top K semantic nearest neighbors retrieved with their SSM values are gathered to plot this graph.

Recall that the SSM value c_i can only take discrete values. Therefore we can define $f(c_i) = w_{c_i}$, where w_{c_i} means this variable only depends on c_i value and we can consider it as a look-up table. Suppose we have trained N_T trees in total, then the inequality $c_i \leq N_T$ always holds, and w_{c_i} can have at most N_T different values. We use $\mathbf{W} = (w_1, w_2, ..., w_{N_T})^T$ to denote the vector form of w.

Denote $a_j = \log \frac{R}{r_j} * \frac{1}{Z}$, which is a tag specific constant, then (6.3) can be rewritten as:

$$q_j = a_j * \sum_{i=1}^{K} (t_{ij} * w_{c_i}), \quad j \in \{1, 2, ..., M\} \tag{6.4}$$

Here, we introduce another indicator variable $\mathbf{D} \in \{0,1\}^{K \times N_T}$. Each of its row contains all zero but only one 1 which corresponds to the position of w_{c_i} in $\mathbf{W}$. Thus (6.4) can be written as:

$$q_j = a_j * [t_{1j}\ t_{2j}\ ...\ t_{Kj}] * \mathbf{D} * \mathbf{W} \tag{6.5}$$

Its matrix form is:

$$\begin{aligned} \mathbf{Q} &= \mathbf{A} .* ([\mathbf{T_1}\ \mathbf{T_2}\ ...\ \mathbf{T_K}] * \mathbf{D} * \mathbf{W}) \\ &= (\underbrace{[\mathbf{A}\ \mathbf{A}\ ...\ \mathbf{A}]}_{N_T} .* ([\mathbf{T_1}\ \mathbf{T_2}\ ...\ \mathbf{T_K}] * \mathbf{D})) * \mathbf{W} \end{aligned} \tag{6.6}$$

where $\mathbf{A} = [a_1, a_2, ...a_M]^T$, '$.*$' represents element-by-element multiplication.

Here, we can see that the prediction of tags can be cast as a linear prediction model. In the training procedure, we need to find the largest value in $\mathbf{Q}$, and make it equal to the ground truth annotation. We can cast it as a learning to rank problem [144].

Denote $\{(\mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{N})\}$ as the set of all possible tag pairs in the training set, where $\mathcal{T}_\mathcal{P}$ represents the ground truth tags and $\mathcal{T}_\mathcal{N}$ represents the rest of the tags. Let

$$\mathbf{\Psi_j} = a_j * [t_{1j}\ t_{2j}\ ...\ t_{Kj}] * \mathbf{D} \tag{6.7}$$

then (6.5) can be simplified as $q_j = \mathbf{\Psi_j} * \mathbf{W} = \langle \mathbf{\Psi_j}, \mathbf{W} \rangle$.

Then this learning to rank problem can be formulated as:

$$\begin{aligned} &\min_{\mathbf{W}, \xi_{ij} \geq 0} \quad \frac{1}{2}\mathbf{W^T W} + C \sum_{(i,j) \in \{(\mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{N})\}} \xi_{ij} \\ &s.t.\ \forall (i,j) \in \{(\mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{N})\} : \langle \mathbf{\Psi_i}, \mathbf{W} \rangle \geq \langle \mathbf{\Psi_j}, \mathbf{W} \rangle + 1 - \xi_{ij} \\ &\forall i :\ w_i \leq w_{i+1}, \quad \mathbf{W} \geq 0, \end{aligned} \tag{6.8}$$

This problem differs from the conventional SVM problem [144] in that it has the additional constraints $\mathbf{W} \geq \mathbf{0}$ and $w_i \leq w_{i+1}$, but still it is a convex problem, and we can solve it using off-the-shelf convex problem solvers[1]. These additional constraints represent our belief that the contribution of the retrieved nearest neighbors should be monotonically increasing with the SSM value.

However, the problem defined in (6.8) is still different from our needs. The aim of the learning to rank problem, as defined in (6.8), is to make the rank of positive samples as high as possible, but in our scenario, we only need to predict the top l tags. This means there is no difference between ranking one positive tag as rank $l+1$ or rank $l+100$. Therefore, we introduce a slack variable ξ_i for each positive tag instead of ξ_{ij} for each positive-negative pair. This slack variable motivates the algorithm to make the rank of positive tags outperform all the negative tags. Our new learning to rank problem is defined as:

$$\begin{aligned} \min_{\mathbf{W},\xi_i \geq 0} \quad & \frac{1}{2}\mathbf{W^T W} + C\sum_{i \in \mathcal{T}_\mathcal{P}} \xi_i \\ s.t.\ \forall (i,j) \in \{(\mathcal{T}_\mathcal{P}, \mathcal{T}_\mathcal{N})\} : & \langle \mathbf{\Psi_i}, \mathbf{W} \rangle \geq \langle \mathbf{\Psi_j}, \mathbf{W} \rangle + 1 - \xi_i \\ & \forall i :\ w_i \leq w_{i+1}, \quad \mathbf{W} \geq \mathbf{0}, \end{aligned} \tag{6.9}$$

One obstacle to directly solving (6.9) is that it will generate huge number of constraints. For example, if a dataset contains 5000 images for training, each image is annotated with 4 tags on average and the size of the tag set is 260, then one image will generate 4*256≈1000 constraints, and the whole dataset will generate about 5 million constraints! However, with the help of the additional constraints $\mathbf{W} \geq \mathbf{0}$ and $w_i < w_{i+1}$, we will show that most of the constraints are redundant and we can reduce the size of the constraints by almost two orders of magnitude.

[1] http://cvxr.com/cvx/

Definition Tag m is **superior** over tag n if $q_m = \langle \mathbf{\Psi_m}, \mathbf{W} \rangle \geq q_n = \langle \mathbf{\Psi_n}, \mathbf{W} \rangle$ for every $\mathbf{W} \in \{\mathbf{W} \geq 0, w_i \leq w_{i+1}\}$.

From the above definition, we can see that if tag m is superior to tag n, then we will always prefer tag m over tag n in predicting the tags. If tag m is the ground truth annotation, then we will always be right. On the contrary, there will be no hope to remedy the mistake. Therefore, such constraints can be considered as redundant and there is no need to add them to our optimization problem. Using the following proposition, we can quickly judge if tag m is superior over tag n.

Proposition 6.1. *Denote* $\mathbf{\Psi_j}$ *which is defined in (6.7) as* $[\psi_{j1}\ \psi_{j2}\ ...\ \psi_{jN_T}]$. *Tag* m *is superior over tag* n *if and only if* $\sum_{i=N}^{N_T} \psi_{mi} \geq \sum_{i=N}^{N_T} \psi_{ni}$ *for every* $N \in \{1, 2, ..., N_T\}$.

Proof

Necessary condition: If tag m is superior over tag n, then

$$\forall N \in \{1, 2, ..., N_T\},\ \sum_{i=N}^{N_T} \psi_{mi} \geq \sum_{i=N}^{N_T} \psi_{ni}$$

According to the definition, tag m is superior over tag n means for any $\mathbf{W}$ that satisfies the constraints $\{\mathbf{W} \geq 0, w_i \leq w_{i+1}\}$, the inequation

$$t_{qm} = \langle \mathbf{\Psi_m}, \mathbf{W} \rangle = \sum_{i=1}^{N_T} \psi_{mi} w_i \geq t_{qn} = \langle \mathbf{\Psi_n}, \mathbf{W} \rangle = \sum_{i=1}^{N_T} \psi_{ni} w_i \tag{6.10}$$

always holds.

For every $N \in \{1, 2, ..., N_T\}$, we set $\mathbf{W} = [\underbrace{0\ 0\ ...\ 0}_{N}\ \underbrace{1\ 1\ ...\ 1}_{N_T - N}]^T$. It is clear to see that this $\mathbf{W}$ satisfy the constraints $\{\mathbf{W} \geq 0, w_i \leq w_{i+1}\}$, thus the inequation

(6.10) holds which is equivalent to:

$$\sum_{i=1}^{N_T} \psi_{mi} w_i = \sum_{i=N_T-N}^{N_T} \psi_{mi} \geq \sum_{i=1}^{N_T} \psi_{ni} w_i = \sum_{i=N_T-N}^{N_T} \psi_{ni} \tag{6.11}$$

Thus the proof for the necessary condition is done.

Sufficient condition: If for any $N \in \{1, 2, ..., N_T\}$, the inequation

$$\sum_{i=N}^{N_T} \psi_{mi} \geq \sum_{i=N}^{N_T} \psi_{ni}$$

holds, then tag m is superior over tag n.

Here, we have the following set of inequations:

$$\begin{cases} \sum_{i=N}^{N_T} (\psi_{mi} - \psi_{ni}) \geq 0 & \forall N \in \{1, 2, ..., N_T\} \\ \mathbf{W} \geq 0 & \\ w_{i+1} \geq w_i & \forall i \in \{1, 2, ..., N_T - 1\} \end{cases} \tag{6.12}$$

and our task is to prove:

$$\sum_{i=1}^{N_T} \psi_{mi} w_i = \langle \mathbf{\Psi_m}, \mathbf{W} \rangle \geq \langle \mathbf{\Psi_n}, \mathbf{W} \rangle = \sum_{i=1}^{N_T} \psi_{ni} w_i$$

Based on the above inequations, we can prove the following steps:

$$
\begin{aligned}
\sum_{i=N}^{N_T}(\psi_{mi}-\psi_{ni})w_i &= w_N(\sum_{i=N}^{N_T}(\psi_{mi}-\psi_{ni}) - \sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni})) + \sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&\geq -w_N \sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni}) + \sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&= -w_N \sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni}) + w_{N+1}(\sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni}) \\
&\qquad - \sum_{i=N+2}^{N_T}(\psi_{mi}-\psi_{ni})) + \sum_{i=N+2}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&= (w_{N+1}-w_N)\sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni}) - w_{N+1}\sum_{i=N+2}^{N_T}(\psi_{mi}-\psi_{ni}) \\
&\qquad + \sum_{i=N+2}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&\geq -w_{N+1}\sum_{i=N+2}^{N_T}(\psi_{mi}-\psi_{ni}) + \sum_{i=N+2}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&\ldots \\
&\geq -w_{N_T-2}\sum_{i=N_T-1}^{N_T}(\psi_{mi}-\psi_{ni}) + \sum_{i=N_T-1}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&= (w_{N_T-1}-w_{N_T-2})\sum_{i=N_T-1}^{N_T}(\psi_{mi}-\psi_{ni}) - w_{N_T-1}\sum_{i=N_T}^{N_T}(\psi_{mi}-\psi_{ni}) \\
&\qquad + \sum_{i=N_T}^{N_T}(\psi_{mi}-\psi_{ni})w_i \\
&\geq (w_{N_T}-w_{N_T-1})(\psi_{mN_T}-\psi_{nN_T}) \geq 0
\end{aligned}
$$

In proving the above steps, we utilize $w_N \sum_{i=N}^{N_T}(\psi_{mi}-\psi_{ni}) \geq 0$ to prove the first inequality, and $(w_{N+1}-w_N)\sum_{i=N+1}^{N_T}(\psi_{mi}-\psi_{ni}) \geq 0$ to prove the second inequality. It is obvious to see that these inequations hold according to (6.12).

Based on this proposition, we can use the following procedure to find the redundant pairs and remove them from the constraint set:
1). Let $\mathbf{\Psi} = [\mathbf{\Psi_1}; \mathbf{\Psi_2}; ...; \mathbf{\Psi_M}]$. Rearrange the rows of $\mathbf{\Psi}$ into the **R**ight-**O**rdered form $\mathbf{\Psi_{RO}}$;

FIGURE 6.4: From left to right: the matrix $\mathbf{\Psi} = [\mathbf{\Psi_1}; \mathbf{\Psi_2}; ...; \mathbf{\Psi_M}]$, where $\mathbf{\Psi_j}$ is defined in (6.7); $\mathbf{\Psi_{RO}}$ obtained by rearranging $\mathbf{\Psi}$ in its Right-Ordered form; the cumulative sum of each row of $\mathbf{\Psi_{RO}}$ in the reverse order.

2). Calculate the cumulative sum of each row vector in the reverse order $\mathbf{\Psi_{cum}}$;

3). We can judge that tag m is **not** superior to tag n if either of the following two conditions hold: the position of $\mathbf{\Psi_n}$ in $\mathbf{\Psi_{RO}}$ is higher than $\mathbf{\Psi_m}$ or any items of $\mathbf{\Psi_n}$ in $\mathbf{\Psi_{cum}}$ is bigger than $\mathbf{\Psi_m}$. This procedure is also illustrated in Fig.6.4.

6.5 Experiments

6.5.1 Example-based Image Retrieval

Firstly we will do experiments to mimic the image-based image retrieval scenario. We use Corel5K dataset [81] as our testbed, which contains 5000 images and 373 different tags in total. It is usually split into 4500 images for training and the remaining 500 for testing, resulting in 260 tags in both the training set and the testing set. For the features, we directly use the features in [46] which are publicly available. There are 15 different kinds of features in total, including two kinds of global features: Gist features and color histograms. Local features include SIFT and hue descriptor which are extracted either densely or at the Harris-Laplacian interest points.

For each image in the testing set, we consider it as a querying image and use our random forest to retrieve its SNS, calculate their SSMs and find its K-Nearest

Neighbors based on the K largest SSMs. To quantify the performances, we define the K-Nearest Semantic Measure (KNSM) as

$$KNSM = \sum_{n=1}^{N} \sum_{j \in Q_n} \sum_{k=1}^{K} s_{njk} \tag{6.13}$$

where *N* is the total number of testing images, *K* is the number of nearest neighbors under consideration, Q_n is the set of (ground truth) tags existed in the testing image n. s_{njk} = 1 if querying image n's tag j exists in its kth semantic neighbor, and s_{njk} = 0 otherwise. A larger KNSM value indicates that the K nearest neighbors share more tags with or are semantically more similar to the querying image.

For the baseline method, we considered the Joint Equal Contribution (JEC) method proposed in [1], where the authors used an equally weighted distances of various features to retrieve nearest neighbor images, and showed that these retrieved images perform very well for image annotation task. Fig.6.5 shows the results of our method and those of JEC. We can see that our method can indeed retrieve images with higher semantic similarity.

Some qualitative results are shown in Fig.6.6. It is clearly seen that, whilst the visual appearances of the nearest neighbor images returned by JEC do resemble those of the querying images, their semantics differ significantly. On the other hand, the images returned by our method not only visually resemble the querying images but also share common semantics with them. These examples demonstrate that our new concept of semantic nearest neighbors and semantic similarity measure can indeed be successfully used in image-based image retrieval system and can reduce the semantic gap.

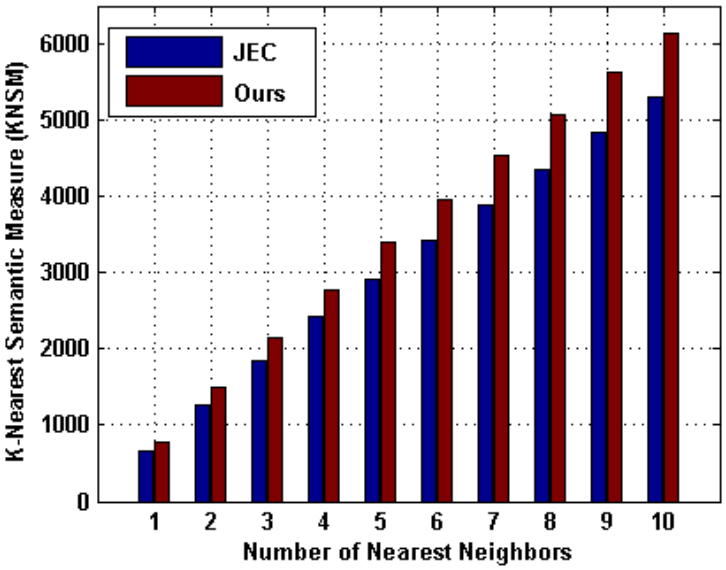

FIGURE 6.5: A comparison of K-Nearest Semantic Measure (KNSM) between our method and JEC [1].

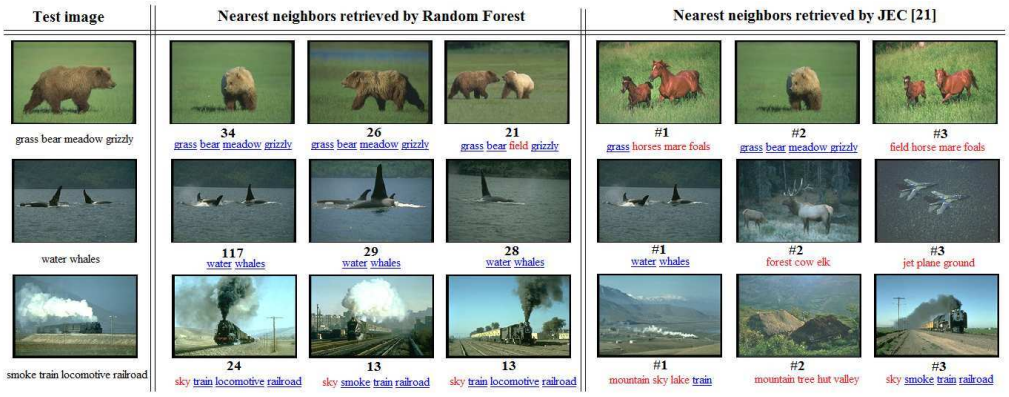

FIGURE 6.6: Some examples of the semantic nearest neighbor images retrieved by our random forest method and by JEC method. The tags associated with each image are also shown beneath each image. The tags which are in accordance with the test image are colored in blue and underlined, while the false tags are colored in red. The numbers underneath the SNN images are the values of SSM.

6.5.2 Keyword-based Image Retrieval

To test keyword based retrieval, we use NUS-WIDE dataset [138] which contains 269,648 images in total, and is usually divided into 161,789 images for training and the rest 107,859 images for testing. Images in the dataset have been manually labeled with 81 different concepts. On average, each image is annotated with 2 concepts. As the concept is relatively sparse on this dataset (about 60000 images contain no concept), previous works [138, 145, 146] have used this dataset for keyword based image retrieval and used the Mean Average Precision (MAP) as the evaluation criterion. Our experiments used the six kinds of visual features released by the creator of the dataset[2].

To construct our random forest, we use kernel PCA to reduce the dimension of each kind of feature to 50 dimensions, resulting in a total of 300 dims. χ^2 distance is used for the bag-of-words feature, and $L2$ distance is used for other five kinds of features. These distances are then transformed to kernels using $k = \exp(-\gamma^{-1} \cdot d)$, where γ is the mean of the distance. As the dataset is quite large, we use an approximation method introduced in [142] to perform kernel PCA. 400 random trees are generated based on these compressed features.

We use (6.2) to predict the probability whether a testing image contains the specific concept/keyword. Based on this probability, all the testing images are ranked for each concept. The MAP accuracy we have achieved with comparison to previous methods are shown in Fig.6.7.

From Fig.6.7, we can see that our method performs much better than its traditional K-NN counterpart, and outperforms most of the previous methods. Although there are some previous works [146, 147] reported higher MAP on this dataset, they all utilize the additional tag information associated with each image. As our method only relies on the visual features, hence these methods are

[2]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

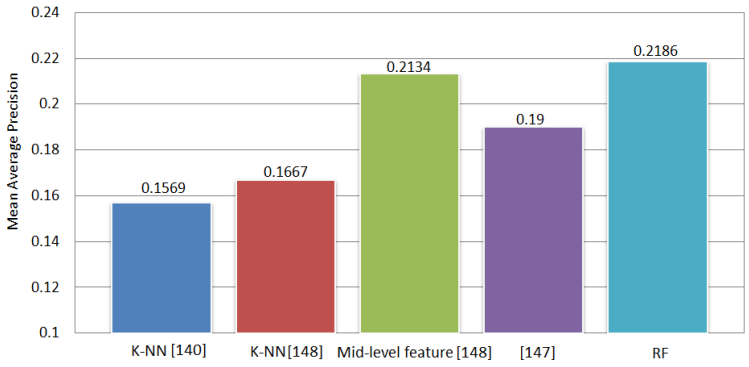

FIGURE 6.7: The MAP achieved by our Random Forest (RF) method with comparison with previously reported results.

not directly comparable. Our aim here is to show the scalability of our algorithm and the advantage of semantic nearest neighbors retrieved by our algorithm over the visual feature nearest neighbors by traditional K-NN method.

6.5.3 Image Annotation Performances

Besides Corel5K dataset, IAPR-TC12 [148] and a subset of ESP-game [46] are another two image annotation datasets which contain approximately 20000 images. More specifically, IAPR-TC12 is usually divided into a fixed number of 17665 images for training and the rest 1962 images for testing, while 291 tags exist both in the training set and testing set; ESP-game contains 18689 images for training and 2081 images for testing. The training set and testing set contain 268 different tags in common.

Much work has been done on these three datasets. In terms of accuracy, Tagprop [46] is the best technique in the literature. The authors of Tagprop have also released the features[3] they used on these three datasets. We directly did experiments based on these features.

[3]http://lear.inrialpes.fr/people/guillaumin/data.php

For all of these three datasets, we generate N_T random trees. Based on the SSM value, we retain K nearest neighbors for each test sample, then the tags are predicted using (6.3) based on these K nearest neighbors. As in other works, five tags are predicted for each image. Average precision, average recall and the number of tags whose recall is above zero ($N+$) are used to evaluate the performance.

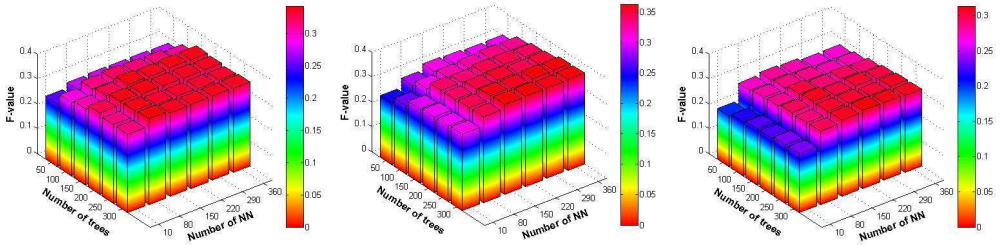

FIGURE 6.8: The relation between the system performance F_value ($F\_value = 2 * Precision * Recall/(Precision + Recall)$), and the parameters of N_T (number of trees) and K (number of Nearest Neighbors used for prediction). From left to right: the results on Corel5K, results on IAPR-TC12 and results on ESP game.

The relation between the performance, and the number of trees N_T and the number of nearest neighbors K are shown in Fig.6.8. From there, we can see that about 200 trees will saturate the performance and we only need to retain about 100 nearest neighbors, and the three dataset exhibit a similar trend. This proves that our algorithm is robust to these parameters.

Before reporting the annotation accuracy, we would also like to qualitatively compare the semantic nearest neighbors retrieved by our method with the visually nearest neighbors retrieved by previous methods, like JEC [1] or Tagprop [46]. Some examples on IAPR-TC12 and ESP game are shown in Fig.6.6.By looking at those visual examples, we noticed that both IAPR-TC12 and ESP game dataset contain some duplicate images which both exist in the training set and testing set. Both our method and JEC can find out these duplicate images. This kind of dataset noise facilitates JEC to obtain a remarkable performance

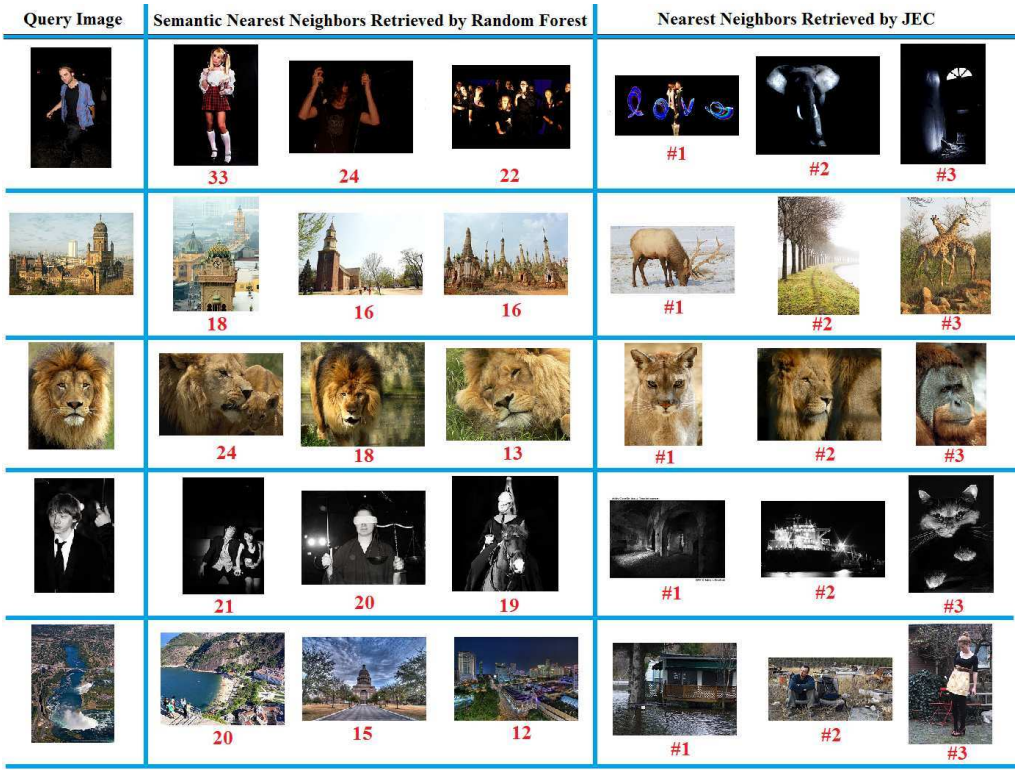

FIGURE 6.9: Some more examples of the nearest neighbor images retrieved by our random forest method and by JEC method. The Semantic Similarity Measure or the rank indexes are shown beneath each retrieved image. The images in the top three rows are from the ESP game, while the bottom three rows are from IAPR-TC12. From this figure, we can see that our method outperforms JEC, and both these two datasets contain some duplicate images that both exist in the training set and testing set.

because it places much emphasis on the first retrieved nearest neighbor in predicting the tags. On the contrary, our method utilizes about 100 nearest neighbors in the prediction; therefore our method is likely to be more robust. For more visual examples, please refer to Appendix A.

These examples demonstrate that our new concept of semantic nearest neighbors and semantic similarity measure can indeed be successfully used to perform nearest neighbor search and reduce the semantic gap of traditional nearest neighbor search. It is this capability to retrieve semantically similar nearest neighbors that has enabled our method to achieve good performances in image annotation.

Table.6.1 shows the average precision and recall performances of our new random forest based image annotation technique and comparisons with state of the art techniques. From Table.6.1, we can see that our methods outperform all previous methods only except Tagprop [46]. However, as mentioned before, the success of Tagprop relies on its sophisticated training procedure and per image per tag optimization, which hinders its extension to large scale datasets. On the contrary, our method can be easily extended to large scale datasets. Besides this, our method is as straightforward as a nearest neighbor method. In this sense, JEC [1] is the most comparable to our method. However, our method shows a clear performance gain over JEC because JEC only relies on the nearest neighbors of visual features, whilst our method finds semantically similar nearest neighbors (also see Fig. 6.6).

TABLE 6.1: Image Annotation Performances on Corel5K, IAPR-TC12 and ESP game. RF represents our Random Forest method. RF_$count$ denote $f(c_i) = c_i$, RF_$count^2$ denote $f(c_i) = {c_i}^2$ and RF_$optimize$ denote $f(c_i)$ is learned based on our optimization framework

	Corel5K			**IAPR-TC12**			**ESP game**		
method	Prec	Recall	N+	Prec	Recall	N+	Prec	Recall	N+
MBRM [149]	0.24	0.25	122/260	0.24	0.23	223/291	0.18	0.19	209/268
JEC [1]	0.27	0.32	139/260	0.28	0.29	250/291	0.22	0.25	224/268
MSC [150]	0.25	0.32	136/260	-	-	-	-	-	-
HPM [96]	0.25	0.28	136/260	-	-	-	-	-	-
M-E Graph [151]	0.25	0.31	-	-	-	-	-	-	-
Tagprop [46]	0.33	0.42	160/260	0.46	0.35	266/291	0.39	0.27	239/268
GS [152]	0.30	0.33	146/260	0.32	0.29	252/291	-	-	-
RF_$count$	0.26	0.36	143/260	0.47	0.22	220/291	0.45	0.24	233/268
RF_$count^2$	0.29	0.41	165/260	0.45	0.31	253/291	0.34	0.27	239/268
RF_$optimize$	0.29	0.40	157/260	0.44	0.31	253/291	0.41	0.26	235/268

6.5.4 Learning to Rank vs. Ad Hoc Annotation Functions

Comparing the results obtained from RF_$count$, RF_$count^2$ and RF_$optimize$ in Table.6.1, we could see that RF_$count^2$ performs best on Corel5K and IAPR-TC12 but performs worst on ESP-game, while RF_$count$ performs slightly better than RF_$count^2$ on ESP-game but worst on the other two datasets. This shows that these ad hoc annotation functions although sometimes can work well but as can be expected, lack consistency. In comparison, the systematic method RF_$optimize$ performs consistently well across all the three datasets. The unpredictable performances of the ad hoc annotation functions across different datasets and the highly consistent good performances of the novel systematic learning to rank algorithm clearly highlighted the value and usefulness of our learning algorithm.

As can be seen the optimization objective of our learning to rank algorithm (6.9) treats each tag equally important. Another useful measurement of performances is to count the total number of correctly predicted tags. Table.6.2 lists the total number of correctly predicted tags for the two ad hoc annotation functions and the systematic learning to rank method. It is seen that the systematic method consistently predicted more correct tags.

TABLE 6.2: The number of correctly predicted tags on each dataset. RF_$optimize$ consistently outperforms the ad-hoc functions.

	Corel5K	IAPR-TC12	ESP game
RF_$count$	993/1756	3957/11053	3778/9774
RF_$count^2$	1004/1756	4242/11053	3724/9774
RF_$optimize$	**1010/1756**	**4254/11053**	**3797/9774**

6.6 Concluding Remarks

In this chapter, we have developed a novel random forest based framework for automatic image annotation and image retrieval. Our new contributions include the use of tag information to guide the generation of the random trees, the introduction of the concept of semantic neighbors and a novel learning to rank framework for systematically learning image annotation models from the semantic neighbor sets. We have presented experimental results which have demonstrated our new method is competitive to the state of the art. We are now planing to test our algorithm on larger datasets which contain millions of images.

Chapter 7

Concluding Remarks

In this thesis, we have studied problems related to semantic image understanding. In this chapter, we will summarize our major contributions, point out their limitations and give some suggestions on how to improve them. We will also discuss some possible directions for future work.

7.1 Main Contributions

Our first contribution in Chapter 3 is to propose the usage of histogram matching in Multiple Kernel Learning. We treat the two-dimensional kernel matrix as an image and transfer the histogram matching algorithm in image processing to kernel matrix. We believe that our proposed method could be generally adopted in Multiple Kernel Learning scenarios and will be a new baseline besides 'average' or 'product'.

Our second contribution is to advocate the segment-then-recognize strategy in pixel-level semantic image understanding. To successfully adopt this strategy, the key is to design a good segmentation algorithm. We can roughly categorize existing segmentation algorithms into three categories: low-level segmentation,

interactive segmentation and semantic segmentation. While low-level segmentation is a purely unsupervised method, interactive segmentation tries to require least human interventions, and semantic segmentation requires full human labeled training data. Each of these three categories has their advantages and disadvantages. Therefore, we have developed in Chapter 4 a hard integration method and a soft integration method. While the hard integration tries to combine semantic segmentation with interactive segmentation, the soft integration method tries to combine semantic segmentation with low-level segmentation. Experiments on two famous datasets have shown that by integrating different segmentation methods, we can indeed obtain a better segmentation algorithm.

We have also successfully applied the segment-then-recognize strategy into medical image analysis in Chapter 5, where we designed a novel polar space random field model for proposing gland-like regions. Experiments have also shown that our method outperforms state of the art methods in the object detection literature and semantic segmentation algorithms which chose pixel or superpixel as their processing primitive.

In the realm of image-level semantic image understanding, our contribution is a novel way to utilize the random forest in Chapter 6. Most of the previous works utilizing random forest store the posterior probabilities at each leaf node, and each random tree in the random forest is considered to be independent from each other. In contrast, we store the training samples instead of the posterior probabilities at each leaf node. We consider the random forest as a whole and propose the concept of semantic nearest neighbor and semantic similarity measure. Based on these two concepts, we devise novel methods for image annotation and image retrieval tasks.

7.2 Limitations and Suggestions for Improvement

We believe that the works presented in this thesis are just a start in their respective fields, and they have either raised new questions or left some room for future improvement.

In Chapter 3, the reason why histogram matching works for MKL is still unclear. We have only empirically confirmed its effectiveness, and there are still no theoretical grounds at this stage. We found that our method works with several existing MKL solvers, and we are directly using these MKL solvers as is. Therefore, these solvers serve as a black box to us. In order to reveal the myth behind our model, we believe that we had to investigate into the MKL solver. A promising way of doing this is to treat the canonical kernel histogram as variables, and directly integrate them into the MKL optimization function.

In our object-consistent segmentation method, an important question is how to automatically decide the number of objects contained in an image. The Dirichlet Process Mixture Model might be a promising tool to tackle this problem. There is also an implicit requirement of our model: the semantic features should have a relatively high accuracy, otherwise it will deteriorate the performance of the whole system. Thus our model is not applicable for scenarios where the semantic segmentation module can only obtain very poor results, such as the PASCAL VOC challenge [70]. Besides, our model is still a two-stage model. No matter it is hard integration or soft integration, the semantic segmentation module will be performed on the image first. Its results are then utilized to help the low-level segmentation module. In this way, these two modules are not really 'interact' with each other. How to design a new framework which can make these two modules really interact is still an open question.

For our GlandVision algorithm in Chapter 5, one of the biggest limitations in our model is that it can only detect star-convexity shapes [128], which is due to

the utility of the polar image and the structure of our random field. Instead of considering this as a limitation, we would rather think it as a characteristic of our model. We would say that our model is specifically designed for detecting star-convexity shapes instead of saying that our model could not detect non star-convexity shapes. If a new algorithm is specifically designed to detect those non star-convexity shapes, then we could combine this new algorithm with our GlandVision. Currently, we are investigating issues to make our model suitable for clinical use. Another possible extension to our model is to combine it with a saliency model [22]. As saliency models can quickly identify regions that are more likely to contain interested objects, our GlandVision algorithm can then be performed only on these salient regions.

We have shown in Chapter 6 that our random forest can efficiently dealt with the problem of image retrieval and image annotation. We are now investigating its applicability in image classification scenario. Besides random forest, semi-supervised hashing [153] and supervised hashing technique [154] is becoming more and more popular recently. A comparison between our random forest and these hashing techniques is necessary if we want to prove the superiority of our random forest method.

7.3 Summary

In summary, we have investigated problems related to semantic image understanding in this thesis. We first developed a novel method in Chapter 3 which can boost the performance of MKL - a state-of-art classifier. This classifier serves as a basis for semantic image understanding problem. We have shown in Chapter 3 that this method can obtain a better performance on the MSRC21 dataset which is our main test bed in Chapter 4. Experiments in Chapter 5 have also confirmed its effectiveness. However, due to the nature of MKL, it is not

applicable for large scale settings, as it is very memory consuming to store the kernel matrixes. Therefore, to tackle with the large scale settings in image-level semantic image understanding, we have developed novel methods in Chapter 6 by using random forest. Our contribution in pixel-level semantic image understanding is to (re)emphasize the importance of segment-then-recognize strategy. Although there are few previous works [133, 155] also adopting this strategy, they usually treat the segmentation algorithm as is. In contrast, we are designing algorithms which can produce more semantic consistent regions in Chapter 4. Chapter 5 is a vivid example which illustrates the superiority of this strategy.

Pixel-level semantic image understanding and image-level semantic image understanding are in fact correlated problems. The results of image-level understanding will serve as a prior to pixel-level understanding, and the results from pixel-level understanding will in turn verify the image-level results. We have made improvements in each individual field in this thesis. Just as some recent works [137] suggest, future work will try to integrate these tasks together and try to solve them simultaneously.

Appendix A

More visual examples

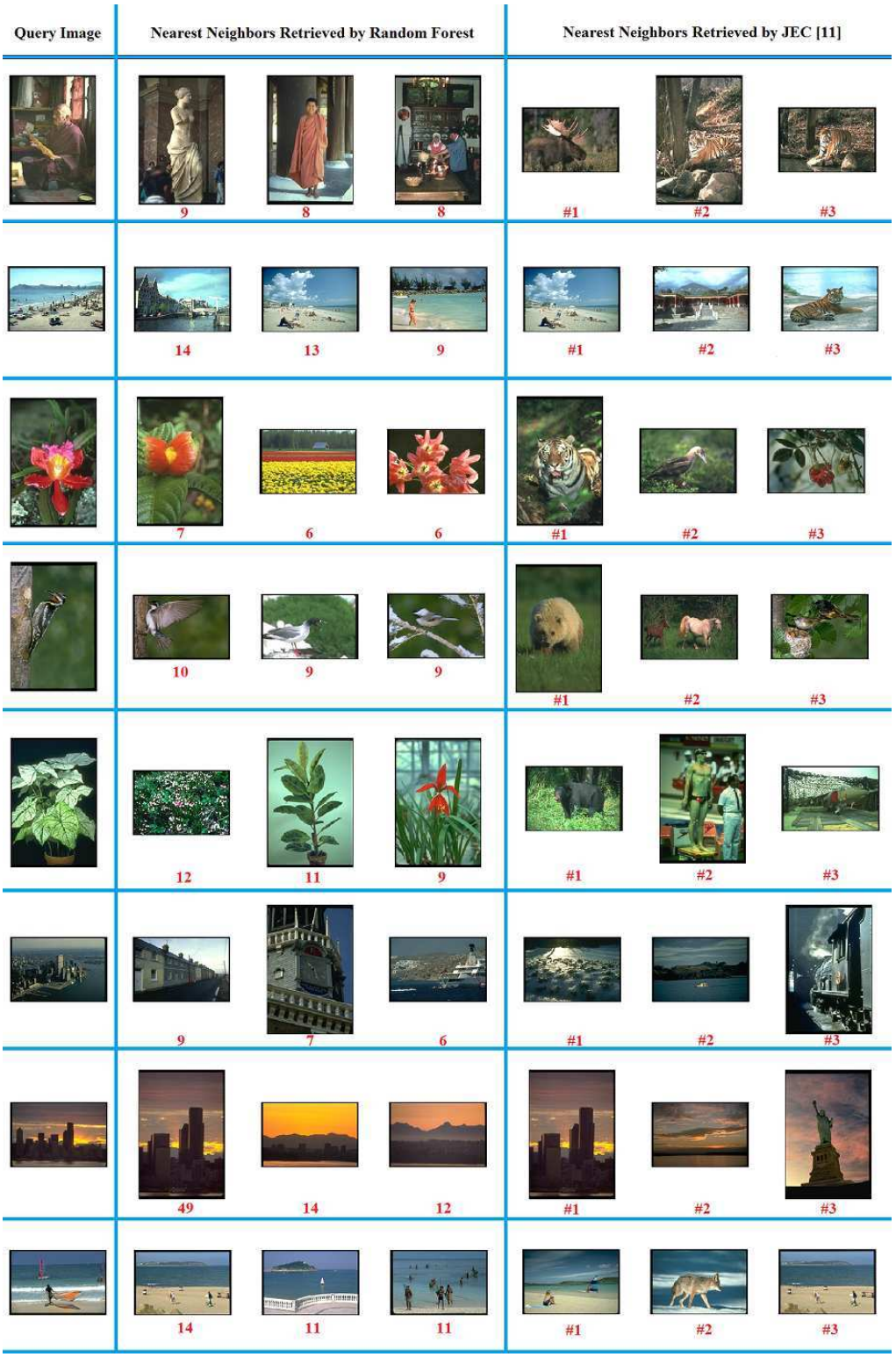

FIGURE A.1: Some examples of the nearest neighbor images retrieved by our random forest method and by JEC method [1], together with their semantic similarity measure (SSM) or the rank index shown under each image.

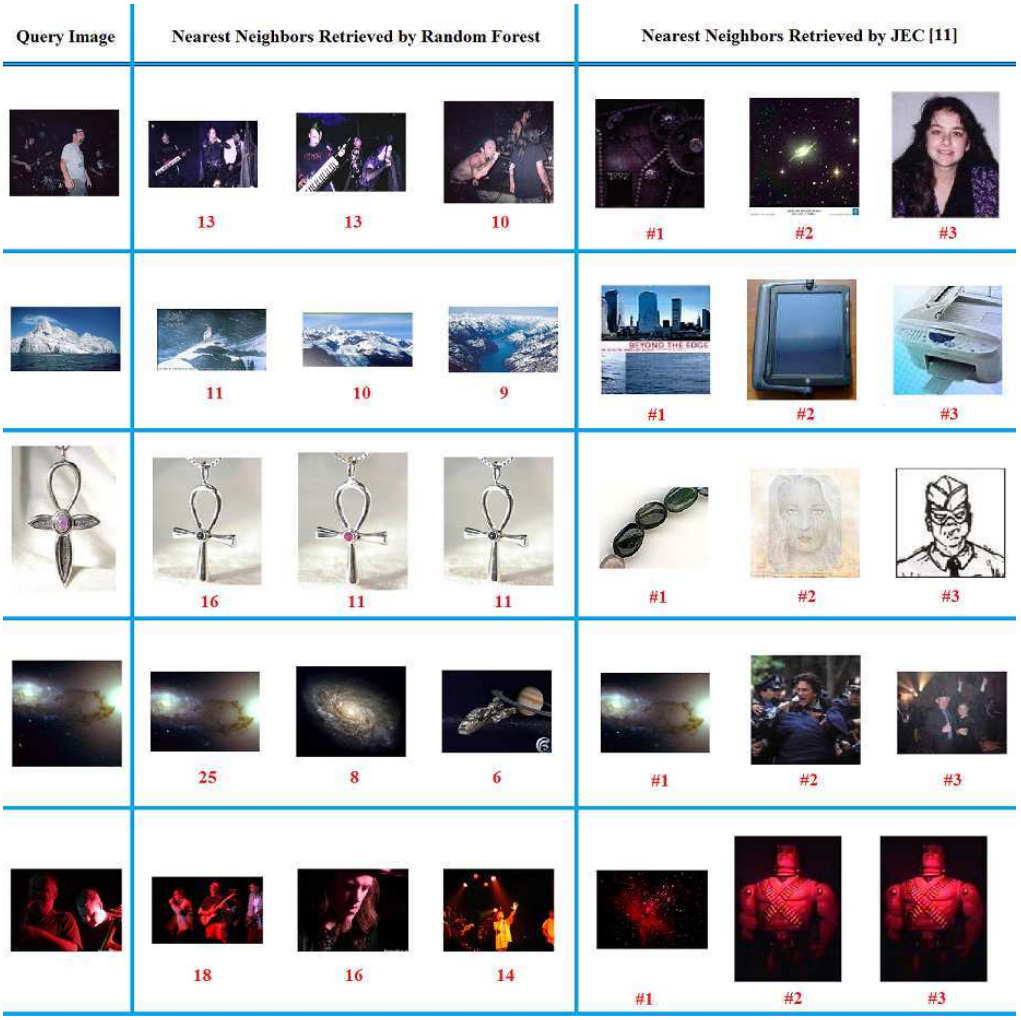

FIGURE A.2: Examples from EPS-game

FIGURE A.3: Examples from IAPR-TC12

Bibliography

[1] Ameesh Makadia, Vladimir Pavlovic, and Sanjiv Kumar. Baselines for Image Annotation. In *ECCV*, 2008.

[2] I Biederman. Recognition-by-components: a theory of human image understanding. *Psychological review*, 94(2):115–47, 1987.

[3] Greg Griffin, Alex Holub, and Pietro Perona. Caltech-256 Object Category Dataset. Technical report, California Institute of Technology, 2007.

[4] Manik Varma and Debajyoti Ray. Learning The Discriminative Power-Invariance Trade-Off. In *ICCV*, 2007.

[5] David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[6] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005.

[7] Manik Varma and Andrew Zisserman. A Statistical Approach to Texture Classification from Single Images. *International Journal of Computer Vision*, 62:61–81, 2005.

[8] Koen E a van de Sande, Theo Gevers, and Cees G M Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1582–96, 2010.

[9] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation. In *ECCV*, 2006.

[10] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008.

[11] Pushmeet Kohli, Lubor Ladický, and PHS Philip H. S. PHS Torr. Robust Higher Order Potentials for Enforcing Label Consistency. In *CVPR*, 2008.

[12] Lubor Ladicky, Chris Russell, Pushmeet Kohli, Philip H S Torr, and L Ladick. Associative Hierarchical CRFs for Object Class Image Segmentation. In *ICCV*, 2009.

[13] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[14] Paul Sturgess, K Alahari, L Ladicky, and P Torr. Combining Appearance and Structure from Motion Features for Road Scene Understanding. In *BMVC*, 2009.

[15] Philipp Krahenbuhl and Vladlen Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *NIPS*, 2011.

[16] Joost Van De Weijer, Cordelia Schmid, and Jakob Verbeek. Learning Color Names from Real-World Images. In *CVPR*, 2007.

[17] Joseph Tighe and Svetlana Lazebnik. SuperParsing : Scalable Nonparametric Image Parsing with Superpixels. In *ECCV*, 2010.

[18] Liefeng Bo, Dieter Fox, Liefeng Bo., and Xiaofeng Ren. Kernel Descriptors for Visual Recognition. In *NIPS*, 2010.

[19] G Qiu. Indexing chromatic and achromatic patterns for content-based colour image retrieval. *Pattern Recognition*, 35:1675–1686, 2002.

[20] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[21] Oncel Tuzel, Fatih Porikli, and Peter Meer. Region Covariance : A Fast Descriptor for Detection and Classification. In *ECCV*, 2006.

[22] Ligang Zheng, Guoping Qiu, Jiwu Huang, and Hao Fu. Salient Covariance for Near-Duplicate Image and Video Detection. In *ICIP*, 2011.

[23] Florent Perronnin, Yan Liu, Jorge Sanchez, and Herve Poirier. Large-scale image retrieval with compressed Fisher vectors. In *CVPR*, 2010.

[24] Regis Behmo, Nikos Paragios, and Veronique Prinet. Graph Commute Times for Image Representation. In *CVPR*, 2008.

[25] Li-jia Li, Hao Su, Eric P Xing, and Li Fei-fei. Object Bank : A High-Level Image Representation for Scene Classification & Semantic Feature Sparsification. In *NIPS*, 2010.

[26] Sreemanananth Sadanand and Jason J Corso. Action Bank : A High-Level Representation of Activity in Video. In *CVPR*, 2012.

[27] Hao Fu and Guoping Qiu. Integrating Low-level and Semantic Features for Object Consistent Segmentation. *Neurocomputing (accpeted)*, 2012.

[28] Aurelien Lucchi, Yunpeng Li, Xavier Boix, Kevin Smith, Pascal Fua, and ETH BIWI. Are Spatial and Global Constraints Really Necessary for Segmentation? In *ICCV*, 2011.

[29] Josef Kittler, Ieee Computer Society, Mohamad Hatef, Robert P W Duin, Jiri Matas, Josef Kirtler, Mohamad Hatefm, and Robert P.W.Duin. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[30] David M.J.Tax and Robert P.W.Duin. Support Vector Data Description. *Machine Learning*, pages 45–66, 2004.

[31] Francesco Orabona, Luo Jie, and Barbara Caputo. Online-Batch Strongly Convex Multi Kernel Learning. In *CVPR*, 2010.

[32] Gert R G Lanckriet, Peter Bartlett, and Michael I Jordan. Learning the Kernel Matrix with Semidefinite Programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[33] Alexander Zien and Cheng Soon Ong. Multiclass multiple kernel learning. In *ICML*, 2007.

[34] Fei Yan, Krystian Mikolajczyk, Mark Barnard, Hongping Cai, and Josef Kittler. lp Norm Multiple Kernel Fisher Discriminant Analysis for Object and Image Categorisation. In *CVPR*, 2010.

[35] Andrea Vedaldi, Varun Gulshan, Manik Varma, and Andrew Zisserman. Multiple kernels for object detection. In *ICCV*, 2009.

[36] Gonen Mehmet and Ethem Alpaydm. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research*, 12:2211–2268, 2011.

[37] Fei Yan, Krystian Mikolajczyk, Josef Kittler, and Atif Tahir. Combining Multiple Kernels by Augmenting the Kernel Matrix. In *In Proceedings of the 9th International Workshop on Multiple Classifier Systems*, 2010.

[38] Isaac Martin Diego, Alberto Muñoz, and Javier M. Moguerza. Methods for the combination of kernel matrices within a support vector framework. *Machine Learning*, 78(1-2):137–174, 2009.

[39] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of Nearest-Neighbor based image classification. In *CVPR*, 2008.

[40] Hao Zhang, A.C. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition. In *CVPR*, 2006.

[41] Regis Behmo, Paul Marcombes, Arnak Dalalyan, and Veronique Prinet. Towards Optimal Naive Bayes Nearest Neighbor. In *ECCV*, 2010.

[42] T Tuytelaars, M Fritz, K Saenko, and T Darrell. The NBNN kernel. In *ICCV*, 2011.

[43] Sancho Mccann and David G Lowe. Local Naive Bayes Nearest Neighbor for Image Classification. In *CVPR*, 2012.

[44] James Hays and Alexei A. Efros. Scene completion using millions of photographs. In *SIGGRAPH*, 2007.

[45] Ce Liu, Jenny Yuen, Antonio Torralba, Josef Sivic, and William T. Freeman. SIFT Flow: Dense Correspondence across Different Scenes. In *ECCV*, 2008.

[46] Matthieu Guillaumin, Thomas Mensink, Jakob Verbeek, and Cordelia Schmid. TagProp: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

[47] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of Exemplar-SVMs for Object Detection and Beyond. In *ICCV*, 2011.

[48] Charles Elkan. Nearest Neighbor Classification. Technical report, 2011.

[49] Marius Muja and David G Lowe. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *VISAPP*, 2009.

[50] Anna Bosch, Andrew Zisserman, and Xavier Munoz. Image Classification using Random Forests and Ferns. In *ICCV*, 2007.

[51] Bangpeng Yao, Aditya Khosla, and Li Fei-Fei. Combining Randomization and Discrimination for Fine-Grained Image Categorization. In *CVPR*, 2011.

[52] Gang Yu, J. Yuan, and Z. Liu. Unsupervised Random Forest Indexing for Fast Action Search. In *CVPR*, 2011.

[53] Juergen Gall and Victor Lempitsky. Class-Specific Hough Forests for Object Detection. In *CVPR*, 2009.

[54] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011.

[55] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *ICML*, 2004.

[56] Sebastian Nowozin and Christoph H Lampert. Structured Learning and Prediction in Computer Vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3-4), 2011.

[57] John Lafferty, Andrew Mccallum, and Fernando Pereira. Conditional Random Fields : Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, 2001.

[58] Geremy Heitz and Daphne Koller. Learning Spatial Context : Using Stuff to Find Things. In *ECCV*, 2008.

[59] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *CVPR*, 2008.

[60] Long Leo, Zhu Yuanhao, Chen Alan, and Yuille William. Latent Hierarchical Structural Learning for Object Detection. In *CVPR*, 2010.

[61] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models.

IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9): 1627–45, 2010.

[62] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(4), 2006.

[63] R. Fergus, P. Perona, and A. Zisserman. A Sparse Object Category Model for Efficient Learning and Exhaustive Recognition. In *CVPR*, 2005.

[64] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined Object Categorization and Segmentation with an Implicit Shape Model. In *ECCV 04 Workshop on Statistical Learning in Computer Vision, Prague*, 2004.

[65] Hedi Harzallah, F Jurie, and Cordelia Schmid. Combining efficient object localization and image classification. In *ICCV*, 2009.

[66] Yong Jae Lee, Kristen Grauman, and Yong Jae Lee. Object-Graphs for Context-Aware Category Discovery. In *CVPR*, 2010.

[67] Stefan Walk, Nikodem Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *CVPR*, 2010.

[68] Josep M J.M. Gonfaus, Xavier Boix, Joost Van De Weijer, Andrew D Bagdanov, Joan Serrat, Gonzalez Jordi, J. van de Weijer, and J. Gonzalez. Harmony potentials for joint classification and segmentation. *International Journal of Computer Vision*, 2010.

[69] Joao Carreira and Cristian Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *CVPR*, 2010.

[70] Mark Everingham, Luc Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2009.

[71] Victor Lempitsky, Andrew Blake, and Carsten Rother. Image Segmentation by Branch-and-Mincut. In *ECCV*, 2008.

[72] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. ClassCut for Unsupervised Class Segmentation. In *ECCV*, 2010.

[73] Sara Vicente, Vladimir Kolmogorov, and Carsten Rother. Cosegmentation Revisited : Models and Optimization. In *ECCV*, 2010.

[74] B Russell, A Efros, Josef Sivic, and WT Freeman. Segmenting scenes by matching image composites. In *NIPS*, 2009.

[75] João Carreira, Fuxin Li, and Cristian Sminchisescu. Object Recognition by Sequential Figure-Ground Ranking. *International Journal of Computer Vision*, 2011.

[76] D. Nister and H. Stewenius. Scalable Recognition with a Vocabulary Tree. In *CVPR*, 2006.

[77] E. Nowak, F. Jurie, and B. Triggs. Sampling strategies for bag-of-features image classification. In *ECCV*, 2006.

[78] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[79] Ritendra Datta, Dhiraj Joshi, J. Li, and J.Z. Wang. Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, 40(2), 2008.

[80] David M. Blei and Michael I. Jordan. Modeling annotated data. In *ACM SIGIR*, 2003.

[81] Pinar Duygulu, Kobus Barnard, Nando de Freitas, and David Forsyth. Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *ECCV*, 2002.

[82] Yu Xiang, Xiangdong Zhou, Fudan Unviersity, Tat-seng Chua, and Chong-wah Ngo. A revisit of Generative Model for Automatic Image Annotation using Markov Random Fields. In *CVPR*, 2009.

[83] G. Carneiro and N. Vasconcelos. Formulating Semantic Image Annotation as a Supervised Learning Problem. In *CVPR*, 2005.

[84] Peter Gehler and Sebastian Nowozin. On Feature Combination for Multiclass Object Classification. In *ICCV*, 2009.

[85] Christian Wallraven and Barbara Caputo. Recognition with local features: the kernel recipe. In *ICCV*, 2003.

[86] Cheng Soon Ong, Xavier Mary, Pierre Larousse, and Alexander J Smola. Learning with Non-Positive Kernels. In *ICML*, 2004.

[87] Bernard Haasdonk. Feature space interpretation of SVMs with indefinite kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):482–92, 2005.

[88] Yin Zhang and Zhi-hua Zhou. Non-Metric Label Propagation. In *IJCAI*, 2009.

[89] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On Kernel Target Alignment. In *NIPS*, 2001.

[90] J. Camargo and F. Gonzalez. A Multi-class Kernel Alignment Method for Image Collection Summarization. In *14th Iberoamerican Congress on Pattern Recognition*, 2009.

[91] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes in machine learning*. 2006.

[92] Maria-Elena Nilsback and Andrew Zisserman. Automated Flower Classification over a Large Number of Classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

[93] Francesco Orabona, Luo Jie, and Barbara Caputo. Multi Kernel Learning with Online-Batch Optimization. *Journal of Machine Learning Research*, 13:165–191, 2012.

[94] Darrin P. Lewis, Tony Jebara, and William Stafford Noble. Nonstationary kernel combination. In *ICML*, 2006.

[95] Matthijs Douze, Arnau Ramisa, and Cordelia Schmid. Combining attributes and Fisher vectors for efficient image retrieval. In *CVPR*, 2011.

[96] Ning Zhou, W. Cheung, Guoping Qiu, and Xiangyang Xue. A Hybrid Probabilistic Model for Unified Collaborative and Content-Based Image Tagging. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:1281–1294, 2011.

[97] Soren Sonnenburg, Gunnar Ratsch, Christin Schafer, and Bernhard Scholkopf. Large Scale Multiple Kernel Learning. *Journal of Machine Learning Research*, 7, 2006.

[98] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, and Yves Grandvalet. More efficiency in multiple kernel learning. In *ICML*, 2007.

[99] Yong Jae Lee, Jaechul Kim, and Kristen Grauman. Key-Segments for Video Object Segmentation. In *ICCV*, 2011.

[100] Jianxiong Xiao, James Hays, K.A. Ehinger, A. Oliva, and Antonio Torralba. SUN database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[101] Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*, 2009.

[102] Tae Hoon Kim, Kyoung Mu Lee, and Sang Uk Lee. Learning Full Pairwise Affinities for Spectral Segmentation. In *CVPR*, 2010.

[103] Lorenzo Torresani, Martin Szummer, and Andrew Fitzgibbon. Efficient Object Category Recognition Using Classemes. In *ECCV*, 2010.

[104] Zhuowen Tu. Auto-context and Its Application to High-level Vision Tasks. In *CVPR*, 2008.

[105] Santosh K. Divvala, Alexei a. Efros, and Martial Hebert. Can similar scenes help surface layout estimation? In *IEEE Workshop on Internet Vision*, 2008.

[106] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. In *ICCV*, 2001.

[107] Charles Sutton and A. McCallum. Piecewise training of undirected models. In *21st Conference on Uncertainty in Artificial Intelligence*, 2005.

[108] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. 1988.

[109] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. From Contours to Regions : An Empirical Evaluation. In *CVPR*, 2009.

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

[111] Lei Zhang, Qiang Ji, and Senior Member. Image Segmentation with a Unified Graphical Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1406–1425, 2010.

[112] Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Stacked Hierarchical Labeling. In *ECCV*, 2010.

[113] MP Kumar and Daphne Koller. Efficiently Selecting Regions for Scene Understanding. In *CVPR*, 2010.

[114] T.H. Kim, K.M. Lee, and S.U. Lee. Nonparametric higher-order learning for interactive segmentation. In *CVPR*, 2010.

[115] Bharath Hariharan, Pablo Arbel, Lubomir Bourdev, Subhransu Maji, Jitendra Malik, U C Berkeley, Adobe Systems, Park Ave, and San Jose. Semantic Contours from Inverse Detectors. In *ICCV*, 2011.

[116] Pablo Arbel, Berkeley Berkeley, Willow Rd, and Menlo Park. Semantic Segmentation using Regions and Parts. In *CVPR*, 2012.

[117] Derek Hoiem and Alexei A Efros. Closing the Loop in Scene Interpretation. In *CVPR*, 2008.

[118] Geremy Heitz, Stephen Gould, A Saxena, and Daphne Koller. Cascaded classification models: Combining models for holistic scene understanding. In *NIPS*, 2008.

[119] Lubor Ladicky, Paul Sturgess, Karteek Alahari, Chris Russell, and Philip H S Torr. What, Where & How Many? Combining Object Detectors and CRFs. In *ECCV*, 2010.

[120] Shivang Naik, Scott Doyle, Michael Feldman, and John Tomaszewski. Automated gland and nuclei segmentation for grading of prostate and breast cancer histopathology. *Surgical Pathology*, 2008.

[121] Kien Nguyen, Bikash Sabata, and Anil K. Jain. Prostate cancer grading: Gland segmentation and structural features. *Pattern Recognition Letters*, 33(7):951–961, 2012.

[122] James Diamond, Neil H. Anderson, Peter H. Bartels, Rodolfo Montironi, and Peter W. Hamilton. The use of morphological characteristics and

texture analysis in the identification of tissue composition in prostatic neoplasia. *Human Pathology*, 35(9), 2004.

[123] Reza Farjam, Hamid Soltanian-zadeh, Kourosh Jafari-khouzani, and Reza A Zoroofi. An Image Analysis Approach for Automatic Malignancy Determination of Prostate Pathological Images. *Cytometry*, 240: 227–240, 2007.

[124] Mount Sinai, One Gustave, and L Levy Place. Segmentation of intestinal gland images with iterative region growing. *Journal of Microscopy*, 220: 190–204, 2005.

[125] Jun Xu, Rachel Sparks, Andrew Janowcyzk, John E Tomaszewski, Michael D Feldman, and Anant Madabhushi. High-Throughput Prostate Cancer Gland Detection , Segmentation , and Classification from Digitized Needle Core Biopsies. *Cancer Imaging*, pages 77–88, 2010.

[126] Shivang Naik, Scott Doyle, Michael Feldman, John Tomaszewski, and Anant Madabhushi. Gland Segmentation and Computerized Gleason Grading of Prostate Histology by Integrating Low- , High-level and Domain Specific Information. In *In Proc. 2nd MICCAI Workshop Microscopic Image Analysis with Appl. in Biology (MIAAB)*, 2007.

[127] Ajay Mishra, Yiannis Aloimonos, and Cheong Loong Fah. Active segmentation with fixation. In *ICCV*, 2009.

[128] Olga Veksler. Star Shape Prior for Graph-Cut Image Segmentation. In *ECCV*, 2008.

[129] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006.

[130] Sanjiv Kumar and Martial Hebert. Discriminative Random Fields. *International Journal of Computer Vision*, 68(2):179–201, 2006.

[131] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, 2007.

[132] Matthew B Blaschko and Christoph H Lampert. Learning to Localize Objects with Structured Output Regression. In *ECCV*, 2008.

[133] Fuxin Li, Joao Carreira, and Cristian Sminchisescu. Object recognition as ranking holistic figure-ground hypotheses. In *CVPR*, 2010.

[134] Chih-chung Chang and Chih-jen Lin. LIBSVM : A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3), 2011.

[135] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. WHAT IS AN OBJECT? In *CVPR*, 2010.

[136] Ian Endres and Derek Hoiem. Category Independent Object Proposals. In *ECCV*, 2010.

[137] Jian Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012.

[138] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE : A Real-World Web Image Database from National University of Singapore. In *CIVR*, 2009.

[139] Xirong Li, CGM Cees G M Snoek, and Marcel Worring. Unsupervised Multi-Feature Tag Relevance Learning for Social Image Retrieval. In *ACM ICIVR*, 2010.

[140] Hideki Nakayama, Tatsuya Harada, and Yasuo Kuniyoshi. Evaluation of dimensionality reduction methods for image auto-annotation. In *BMVC*, 2010.

[141] Bernhard Scholkopf, Alexander Smola, Klaus-Robert Muller, B Schölkopf, and KR Müller. Kernel principal component analysis. In *ICANN*, 1997.

[142] Kai Zhang, Ivor W. Tsang, and James T. Kwok. Improved Nystrom Low-Rank Approximation and Error Analysis. In *ICML*, 2008.

[143] Jiwei Hu, Kin Man Lam, and Guoping Qiu. A Hierarchical Algorithm for Image Multi-labeling. In *ICIP*, 2010.

[144] Thorsten Joachims. Training Linear SVMs in Linear Time. In *ACM KDD*, 2006.

[145] Xiangyu Chen, Yadong Mu, Shuicheng Yan, and Tat-Seng Chua. Efficient Large-Scale Image Annotation by Probabilistic Collaborative Multi-Label Propagation. In *ACM MM*, 2010.

[146] Shenghua Gao, Liang-Tien Chia, and Xiangang Cheng. Understanding tag-cloud and visual features for better annotation of concepts in NUS-WIDE dataset. In *1st workshop on Web-scale multimedia corpus*, 2009.

[147] Gang Wang, Tat Seng Chua, Chong-Wah Ngo, and Yong Cheng Wang. Automatic Generation of Semantic Fields for Annotating Web Images. In *International Conference on Computational Linguistics*, 2010.

[148] Hugo Jair Escalante, Carlos a. Hernández, and Jesus a. Gonzalez. The segmented and annotated IAPR TC-12 benchmark. *Computer Vision and Image Understanding*, 2010.

[149] S Feng, R Manmatha, and V Lavrenko. Multiple Bernoulli Relevance Models for Image and Video Annotation. In *CVPR*, 2004.

[150] Changhu Wang, Shuicheng Yan, Lei Zhang, and Hong-jiang Zhang. Multi-label sparse coding for automatic image annotation. In *CVPR*, 2009.

[151] Dong Liu, Shuicheng Yan, Yong Rui, and Hong-Jiang Zhang. Unified Tag Analysis With Multi-Edge Graph. In *ACM MM*, 2010.

[152] Shaoting Zhang, Junzhou Huang, Yuchi Huang, Yang Yu, Hongsheng Li, and D Metaxas. Automatic Image Annotation Using Group Sparsity. In *CVPR*, 2010.

[153] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-Supervised Hashing for Scalable Image Retrieval. In *CVPR*, 2010.

[154] Wei Liu, Jun Wang, Rongrong Ji, and Yu-gang Jiang Shih-fu Chang. Supervised Hashing with Kernels. In *CVPR*, 2012.

[155] Chunhui Gu, JJ Lim, P Arbeláez, and Jitendra Malik. Recognition using regions. In *CVPR*, 2009.