Learning Discriminative Features and Structured Models for Segmentation in Microscopy and Natural Images

THÈSE N^{O} 5836 (2013)

PRÉSENTÉE LE 6 SEPTEMBRE 2013

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

LABORATOIRE DE VISION PAR ORDINATEUR

PROGRAMME DOCTORAL EN INFORMATIQUE, COMMUNICATIONS ET INFORMATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Aurélien LUCCHI

acceptée sur proposition du jury:

Prof. P. Dillenbourg, président du jury
Prof. P. Fua, directeur de thèse
Dr G. Knott, rapporteur
Dr P. Kohli, rapporteur
Dr V. Lempitsky, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2013

Abstract

Segmenting images is a significant challenge that has drawn a lot of attention from different fields of artificial intelligence and has many practical applications. One such challenge addressed in this thesis is the segmentation of electron microscope (EM) imaging of neural tissue. EM microscopy is one of the key tools used to analyze neural tissue and understand the brain, but the huge amounts of data it produces make automated analysis necessary.

In addition to the challenges specific to EM data, the common problems encountered in image segmentation must also be addressed. These problems include extracting discriminative features from the data and constructing a statistical model using ground-truth data. Although complex models appear to be more attractive because they allow for more expressiveness, they also lead to a higher computational complexity. On the other hand, simple models come with a lower complexity but less faithfully express the real world. Therefore, one of the most challenging tasks in image segmentation is in constructing models that are expressive enough while remaining tractable.

In this work, we propose several automated graph partitioning approaches that address these issues. These methods reduce the computational complexity by operating on supervoxels instead of voxels, incorporating features capable of describing the 3D shape of the target objects and using structured models to account for correlation in output variables. One of the non-trivial issues with such models is that their parameters must be carefully chosen for optimal performance. A popular approach to learning model parameters is a maximum-margin approach called Structured SVM (SSVM) that provides optimality guarantees but also suffers from

two main drawbacks. First, SSVM-based approaches are usually limited to linear kernels, since more powerful nonlinear kernels cause the learning to become prohibitively expensive. In this thesis, we introduce an approach to "kernelize" the features so that a linear SSVM framework can leverage the power of nonlinear kernels without incurring their high computational cost. Second, the optimality guarentees are violated for complex models with strong inter-relations between the output variables. We propose a new subgradient-based method that is more robust and leads to improved convergence properties and increased reliability.

The different approaches presented in this thesis are applicable to both natural and medical images. They are able to segment mitochondria at a performance level close to that of a human annotator, and outperform state-of-the-art segmentation techniques while still benefiting from a low learning time.

Keywords: Image processing, computer vision, electron microscopy, image segmentation, kernel methods, mitochondria, statistical machine learning, structured prediction, segmentation, superpixels, supervoxels, shape features.

La segmentation d'images est un défi important qui a attiré beaucoup d'attention dans différents domaines de l'intelligence artificielle et présente de nombreuses applications. Un de ces défis abordé dans cette thèse est la segmentation d'images de tissu neural acquises avec un microscope électronique (ME). Ce type de microscopie est l'un des principaux outils utilisés pour analyser le tissu neural et pour comprendre le fonctionnement du cerveau, mais des quantités énormes de données sont produites, ce qui nécéssite l'automatisation de l'analyse.

En plus des défis spécifiques aux données ME, les problèmes qui proviennent de la segmentation d'images doivent aussi être pris en compte. Ces problèmes comprennent l'extraction de caractéristiques visuelles des données et la construction d'un modèle statistique à partir d'annotations des images. Bien que les modèles complexes paraissent être plus attrayants car ils sont plus expressives, ils menent aussi à une complexité de calcul plus élevée. En revanche, les modèles simples s'accompagnent d'une complexité plus basse mais ne peuvent pas représenter fidèlement la réalité. Par concéquence, une des tâches les plus exigeantes dans le domaine de la segmentation est la construction de modèles expressifs qui ont aussi une complexité raisonnable.

Dans ce travail, nous proposons plusieurs approches de partitionnement de graphes automatiques qui traitent ces problèmes. Ces méthodes réduisent la complexité de calcul en éffectuant des opérations sur des supervoxels au lieu de voxels ainsi que par l'intégration de caractéristiques visuelles capable de décrire la forme 3D des objets cibles et par l'utilisation de modèles structurés pour pour tenir compte de la corrélation des variables de sortie. Un des problèmes non négligeable de ces modèles est que leurs paramètres doivent être soigneusement choisi pour une performance optimale. Une approche populaire pour apprendre ces paramètres est appelée "Machine structuréé à vecteurs de support" (SSVM) et offre des garanties d'optimalité, mais souffre aussi de deux inconvénients principaux. Premièrement, SSVM est généralement limité à des noyaux linéaires, puisque des noyaux non linéaires

engendrent des coûts de calcul prohibitif. Dans cette thèse, nous introduisons une approche qui transforme les caractéristiques visuelles afin qu'un SSVM linéaire puisse tirer parti de la puissance de noyaux non linéaires, sans encourir leur coût de calcul. Deuxièmement, les garanties d'optimalité sont violées pour les modèles complexes avec de fortes inter-relations entre les variables de sortie. Nous proposons une nouvelle méthode à base de sous-gradient qui est plus robuste et permet d'améliorer les propriétés de convergence et de fiabilité.

Les différentes approches présentées dans cette thèse sont applicables aux images naturelles ainsi que médicales. Elles sont capables de segmenter des mitochondries et atteignent un niveau de performance proche de celui d'un annotateur humain et surpasse les techniques de segmentation de pointe, tout en bénéficiant d'une faible temps d'apprentissage.

Mots-clés: traitement d'image, vision par ordinateur, microscopie électronique, segmentation d'images, les méthodes de kernel, mitochondries, apprentissage statistique automatique, prédiction structurée, segmentation, superpixels, supervoxels, caractéristiques visuelles de forme.

Acknowledgements

The work presented in this thesis was accomplished during four wonderful years at the Computer Vision Laboratory at EPFL under the supervision of professor Pascal Fua. I greatly benefited from his guidance and his feedback on my work. I also would like to congratulate Pascal for creating such a great work place filled with brilliant people. I am also very grateful as I had the opportunity to do two internships in outstanding places during my PhD. In 2011, I spent a summer at Google New York working with Dr. Jason Weston on new machine learning algorithms for image ranking. In 2012, I spent another summer away at Microsoft Research in Cambridge working in the Medical and Machine Learning groups, under the supervision of Dr. Darko Zikic and Dr. Antonio Criminisi.

During my four years at CvLab, I met fantastic people who had a very positive influence on my life and I wish to say that I am grateful to all of them. To my former office mates Mustafa Özuysal and Appu Shaji for the invaluable help I received during my first years in the PhD program. To Raphael Sznitman for the interesting work we accomplished together. To Engin Türetken for the hard yet rewarding time we spent together during the first year in the PhD program. To Karim Ali for interesting and passionate discussions about anything from science to politics. To German Gonzalez for providing valuable source code. To my co-authors Radhakrishna Achanta and Xavier Boix for their assistance and the good and stressful time we shared during the preparation of our publications. To Bohumil Maco for painfully annotating data and testing our software. To my current office-mate Anne Jorstad for carefully proof-reading parts of this thesis. To all my co-workers and friends at CvLab, Fethallah Benmansour, Carlos Becker, Mario Christoudias, Natasa Pjescic, Tomasz Trzcinski and many others for making CvLab such an unforgettable place. Great thanks to Josiane for taking care of all of us.

This thesis could not have been completed without the data provided by Dr. Graham Knott who also accepted to be a member of my thesis committee. Graham has always been very patient and understanding. I also thank the other members of my thesis committee, Dr. Pierre Dillenbourg, Dr. Pushmeet Kohli and Dr. Victor Lempitsky, for accepting to evalute this work.

I would like to thank two very special individuals who assisted me during my PhD. I had the great pleasure to work with Dr. Kevin Smith who was a post-doctorate at CvLab during the first two years of my PhD. I am well aware that working with a first year PhD student was not always easy but Kevin did a good job at steering me in the right direction. I was also pleased to work with Dr. Yunpeng Li from who I learned a lot about structured prediction and many other topics.

Finally, I would like to thank my family, especially my parents who supported me all along my academic studies.

CONTENTS

List of Figures **v**

List of Tables **vii**

1 Introduction **1**

1.1 The segmentation problem 1

1.2 Image representation 2

1.3 Trends in image segmentation 5

1.3.1 Active contour 5

1.3.2 Graph-Partitioning approaches 7

1.3.3 Classification based approaches 8

1.3.4 Structured prediction models 9

1.4 Medical imaging 10

1.5 Contributions of this thesis 11

2 Background on graphical models **17**

2.1 Markov random fields 18

2.1.1 Example: a 2D MRF for images 20

2.1.2 Labeling problem 21

2.1.2.1 MAP inference 22

2.1.2.2 Parameter learning 22

2.1.3 Conditional random fields 23

2.2 Inference 23

2.2.1 Variational methods 24

2.2.2 Mean field . . . 26
2.2.2.1 Mean field approximation . . . 26
2.2.2.2 Image denoising with mean field . . . 28
2.2.3 Bethe approximation . . . 29
2.2.4 Belief propagation . . . 30
2.2.4.1 Sum-product algorithm . . . 30
2.2.4.2 Max-Product algorithm . . . 31
2.2.5 More advanced variational algorithms . . . 32
2.2.6 Graph-cuts . . . 32
2.3 Parameter Learning . . . 33
2.3.1 Maximum Likelihood Estimation . . . 33
2.3.2 Approximation methods . . . 35
2.3.2.1 Pseudo-likelihood . . . 35
2.3.2.2 Contrastive divergence . . . 36
2.3.3 Loss-sensitive training . . . 36
2.3.4 Maximum-margin learning . . . 37

3 Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features 39

3.1 Introduction . . . 39
3.2 Related Work . . . 41
3.2.1 Mitochondria Segmentation . . . 41
3.2.2 Machine Learning in EM Imagery . . . 42
3.2.3 Segmentation by Graph-Partitioning . . . 43
3.3 Method . . . 44
3.3.1 Supervoxel Over-segmentation . . . 44
3.3.2 Feature Vector Extraction . . . 48
3.3.2.1 Ray Descriptors . . . 49
3.3.2.2 Histogram Features . . . 52
3.3.3 Graph Cuts with Learned Potentials . . . 52
3.3.3.1 Energy Function . . . 52
3.3.3.2 Learned Shape Cues in the Unary Term . . . 53
3.3.3.3 Learned Boundary Appearance in the Pairwise Term . . . 55

3.4 Results . . . 56
3.4.1 Experimental Setup . . . 56
3.4.2 Parameters and Implementation Details . . . 57
3.4.3 Experiments and Evaluation . . . 57
3.4.4 Discussion . . . 62
3.4.4.1 Computational Advantage of SLIC Supervoxels . . . 62
3.4.4.2 Benefits of Ray Descriptors . . . 63
3.4.4.3 Learning the Pairwise Term . . . 64
3.4.4.4 Comparing against a state-of-the-art method . . . 65
3.4.4.5 Failure modes . . . 66
3.5 Conclusion . . . 67

4 Structured Image Segmentation using Kernelized Features 69
4.1 Introduction . . . 69
4.2 Related Work . . . 71
4.3 Learning a CRF with Kernelized Features . . . 73
4.3.1 CRF for Segmentation . . . 73
4.3.2 Learning the CRF Using SSVM . . . 74
4.3.3 Linearizing the CRF Energy Function . . . 76
4.4 Kernel-transformed Features . . . 77
4.5 Results . . . 80
4.5.1 Competing Methods . . . 80
4.5.2 MSRC Dataset . . . 81
4.5.3 Synaptic Gap Dataset . . . 84
4.5.4 Mitochondria Dataset . . . 85
4.5.5 Discussion . . . 86
4.6 Conclusion . . . 87

5 Learning for Structured Prediction Using Stochastic Descent with Working Sets 89
5.1 Introduction . . . 89
5.2 Related work . . . 90
5.3 Formulation . . . 92
5.4 Max-margin Learning of CRFs . . . 92

5.4.1 Discriminative Learning 92
5.4.2 Max-margin Formulation 93
5.4.3 Stochastic Subgradient Descent 94
5.5 Estimating Subgradient Using Working Sets 95
5.6 Experimental Results 98
5.6.1 CRF for Image Segmentation 98
5.6.2 Methods 99
5.6.3 MSRC Dataset 99
5.6.4 Electron Microscopy Dataset 102
5.6.5 Time analysis 104
5.7 Conclusion 106

6 Enhanced techniques for the segmentation of mitochondria 107
6.1 Introduction 107
6.2 Encoding geometric interactions 108
6.3 Results 109
6.4 Semi-automated splitting of merged regions 115
6.5 Conclusion 116

7 Conclusion 119

8 Appendix 123
8.1 Quadratic complexity of non-linear Structured SVM 123
8.2 Supplementary: Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets 125
8.2.1 Convergence properties of the t^{th} parameter vector 125

Bibliography 129

LIST OF FIGURES

1.1 Segmentation examples 2
1.2 Heuristics for image segmentation 3
1.3 Segmentation results of level sets methods 6
1.4 FIB-SEM (Focused Ion Beam - Scanning Electron Microscope) 11
1.5 FIB-SEM data sets 12
1.6 Images segmented using SLIC 14

2.1 An example of an MRF with pairwise potentials 20
2.2 KL divergence 25
2.3 Mean-field approximation 27
2.4 Mean-field applied to image denoising 29
2.5 Message passing for belief propagation 31

3.1 Segmenting an image stack into supervoxels 43
3.2 Supervoxel size and compactness 46
3.3 Ray feature function 49
3.4 Rotation invariant 3D Ray descriptor 50
3.5 Segmentation of mitochondria from FIB-SEM image stacks and 3D reconstructions 59
3.6 Segmentation results 60
3.7 Thresholding unary SVM predictions vs. our learned pairwise approach 64
3.8 Visual comparison of our results vs. Ilastik 66

LIST OF FIGURES

4.1 Two nearly isotropic stacks of neural tissue acquired using EM microscopy annotated for training and testing . . . 70
4.2 Illustration of our kernel-transform approach . . . 75
4.3 Comparison of a linear SVM, RBF-SVM, and a linear SVM trained on feature vectors kernelized using the support vectors of the RBF-SVM . . 79
4.4 Example segmentations from the MSRC dataset . . . 81
4.5 Segmentation results on the synaptic gap dataset . . . 85
4.6 Segmentation results on the EM dataset . . . 86

5.1 A nearly isotropic stack of neural tissue acquired using EM microscopy annotated for training and testing . . . 100
5.2 Example segmentations from the MSRC dataset . . . 100
5.3 Segmentation results on the EM dataset . . . 103
5.4 Evolution of the training and test scores as a function of the number of iterations . . . 105

6.1 Merging issue . . . 108
6.2 Multi-layer model . . . 109
6.3 Example segmentations produced by different methods on the Striatum EM dataset . . . 111
6.4 Example segmentations produced by different methods using kernelized features on the Striatum EM dataset . . . 112
6.5 Example segmentations produced by the Multi-layer method on the Striatum EM dataset . . . 113
6.6 Mistakes made by the Multi-layer method on several EM datasets . . . 114
6.7 Splitting procedure . . . 117
6.8 Mitochondrial volume . . . 118
6.9 Mitochondrial surface area . . . 118

LIST OF TABLES

3.1 Parameters and Settings 58
3.2 Segmentation Results measured by the VOC Score [26] 61

4.1 MSRC segmentation results 82
4.2 Segmentation results for the synaptic gap EM dataset 84
4.3 Segmentation results for the mitochondria EM dataset 86

5.1 MSRC segmentation results 101
5.2 Segmentation performance measured with the Jaccard index for the mitochondria EM dataset 102
5.3 Running time for the EM and MSRC datasets 105

6.1 Segmentation results for the Hippocampus EM dataset. 110
6.2 Segmentation results for the Striatum EM dataset. 110
6.3 Unary weights for splitting procedure. 115

LIST OF TABLES

CHAPTER

ONE

INTRODUCTION

1.1 The segmentation problem

One of the fundamental dreams in artificial intelligence is to design a computer that could understand an image like humans do. Although this seems to be a trivial task for humans, it has proved to be an extremely difficult problem, and has received a lot of interest during the past fifty years. This problem of interpreting the visual content of an image is often referred to as image segmentation. A simple definition of image segmentation is the task of finding groups of pixels that "go together". This is illustrated in Figure 1.1 for a biomedical image (upper row) and a natural image (bottom row) where the colors indicate the group to which each pixel is associated. Early work on image segmentation starting in the 1970s [17, 108] tried to recognize objects by merging image regions that have similar properties. Figure 1.2 illustrates the results of two heuristics known as "phagocyte" and "weakness" proposed by Brice et al. [17]. The "phagocyte" heuristic guides the merging of regions in such a way as to smooth or shorten the resulting boundary while the "weakness" heuristic joins regions on the basis of the strength of the boundary that separates them. Image segmentation has found many different applications over the past 40 years and numerous segmentation algorithms have been proposed, including thresholding, clustering, region growing and graph partitioning methods. One of the important applications studied in this thesis is the segmentation of electron microscopy (EM) images. This type of imagery can

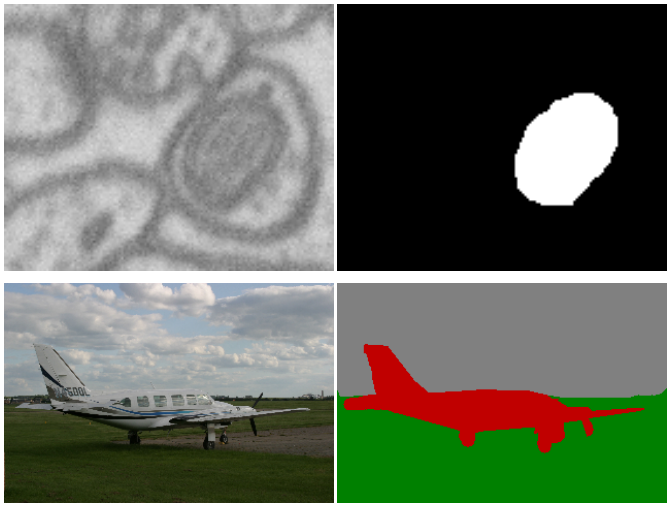


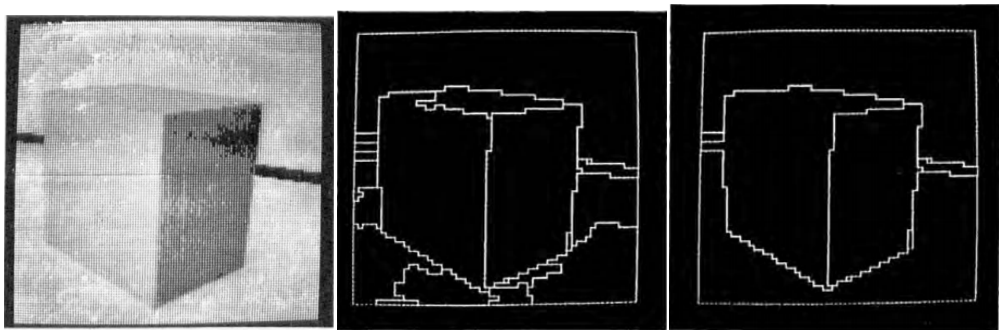
Figure 1.1: *Segmentation examples.* The left image in the first row shows a mitochondrion and its segmentation on the right. The second row is an image from the MSRC database (left) with its segmentation (right).

provide very high resolution images of brain tissue, whose analysis could turn out to be critical for unlocking the mysteries of the brain.

This chapter is organized as follows. We first discuss the way that images are represented and processed on computers. We then present the most common algorithms for image segmentation. We will see that successful algorithms must extract complex features from images to explain the global context of a scene and determine the shape of the objects present in the image. There has been significant research in the field of image segmentation that targets biomedical applications and this will serve as the primary field of application of this manuscript.

1.2 Image representation

Quantized images are represented as sets of pixels encoding color/brightness information in matrix form. Specific structures in image are called "features" and are relevant for solving computational tasks related to a certain application. Image features range from simple structures such as points or edges to more complex structures such as whole objects. Edges are sharp variations of pixel intensities and often indicate important events and changes in world properties. Consequently, the computer vision community



(a) *(b)* *(c)*

Figure 1.2: *(a)* Original image. *(b)* Result produced by the "phagocyte" heuristic. *(c)* Result produced by the "weakness" heuristic. Images taken from [17].

has spent a fair amount of energy to develop numerous edge detection algorithms. A common similarity among these algorithms is the computation of a measure of edge strength, usually the first-order derivative of the image intensity (gradient magnitude). The local maxima of the gradient magnitude are then perceived as meaningful edges.

Properties of edges including gradient and orientation can be extracted with the use of linear filters such as Gabor filters [27]. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. Extracting the edge properties of an image requires convolving the image with a filter. The standard convolution algorithm has quadratic computational complexity, but faster algorithms exist, for example the Fast Fourier Transform has an $O(n \log(n))$ complexity. The extraction of edge features is a time-consuming operation as it requires repetitively convolving the image with a set of Gabor filters with different frequencies and orientations. A faster alternative is to use steerable filters [33] that can be decomposed into simpler forms that require less computation. Extracting a feature for a given orientation does not require the evaluation of new filter responses, but simply multiplying the feature vector extracted at a given canonical orientation by a predefined matrix.

Because not all edges in an image are meaningful, there has been a lot of work focusing on creating image features that are highly distinctive and partially invariant to the variations occurring in an image, such as illumination, 3D viewpoint, etc. The most famous descriptor used by the computer vision community is the SIFT descriptor (Scale-Invariant Feature Transform) proposed by David Lowe [86]. SIFT-based descriptors

have been shown to outperform other local descriptors on both textured and structured scenes, with the difference in performance larger on the textured scene. For further details about local descriptors, we refer the reader to [138].

Although low-level features are very useful for image segmentation, it has been recognized that they alone cannot produce a complete final correct segmentation [120]. While low-level or local features tend to represent small image patches, global features describe an object or an image as a whole. A simple and reliable global image feature for scene recognition is obtained by encoding the organization of color blobs in an image [22]. The GIST descriptor [105] describes the spatial layout of an image using global features derived from the spatial envelope of an image. The image is divided into a 4×4 grid and orientation histograms are computed in each grid element. Although such global descriptors are very effective for natural images [87], they are not useful for the type of biomedical images addressed in this thesis. This is because it is known that certain biological structures like mitochondria or synapses always appear in a biological dataset, whereas it is not known if a cow or a bird will appear in a particular image from a natural image dataset such as MSRC.

Other types of commonly used features include shape features. The shape of an object describes its characteristic outline or contour and is commonly exploited by humans to recognize objects. Most of the existing approaches used to extract shape information can be categorized as region-based, contour-based, or template-based. We here give a simple overview of the existing approaches and refer the reader to a recent survey [151] for further details.

Region-based features include simple geometric features such as center of gravity, axis of least inertia, eccentricity, circularity ratio, rectangularity, convexity, hole area ratio, etc. A more complex description can be extracted with shape context [10] by discretizing a contour into a set of points. For each point, the relative position of the other points is encoded in a histogram.

Contour-based methods exploit shape boundary information. They can be divided into two types: global or structural. Global approaches do not divide shape into subparts but describe the shape from the entire boundary. One example of such features is called shape signatures and include complex coordinates, centroid distance functions, tangent angles, curvature functions, area functions, etc. More complex methods such as Fourier or wavelet descriptors are sometimes desirable as they yield rotation or

translation invariance but can dramatically increase the computational complexity. On the other hand, structural approaches break the shape boundary into segments, called primitives or fragments, using a particular criterion. They then match a segment to a predefined code book of fragments [3, 72] to incorporate some shape information. However, for highly deformable objects, a prohibitively large code book becomes necessary making the approach computationally expensive.

Template-based approaches, such as [1, 32, 100], incorporate a shape template that must be fitted to the image in an alignment or detection step. Such templates can be a contour [32] or silhouette [1, 100] representing target objects, which is learned or painstakingly constructed beforehand. It can be used in conjunction with a CRF model [1, 32, 100]. The complexity of this approach and the problem of aligning multiple templates to the image often limits its applicability to images of a single well-centered object. In Chapter 3, we introduce a new kind of shape signature called "ray features" which are efficient to compute and can also include additional information about an object like the gradient near the object boundary or the orientation of the surface.

1.3 Trends in image segmentation

In the literature of image processing and computer vision, various theoretical frameworks have been proposed for segmentation. This section will present some of the dominant mathematical models such as Markov Random Fields (MRF), active contour models (or deformable models) and learning-based approaches.

1.3.1 Active contour

Image contours play a very important role in the recognition of objects. Contour integration [44] is believed to be a fundamental process by which the human visual system recognizes coherent forms out of a discontinuous sequence of line segments. Although the neuronal mechanisms are still not well understood, psycho-physical experiments have shown that humans are remarkably efficient at integrating contours even if they are jittered or partially occluded [25]. The fundamental role of contours led to the development of PDE-based techniques that can exploit this cue, such as level sets or implicit active contours [106]. The idea behind active contours for image segmentation is the following. An initial guess of the contour is updated by image driven forces to

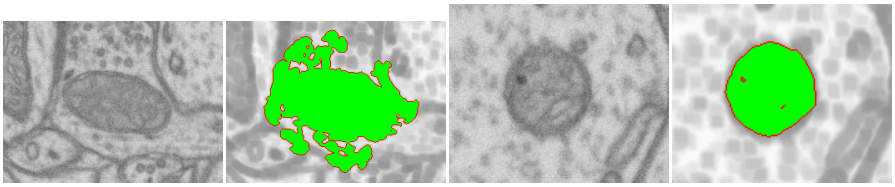


Figure 1.3: *Segmentation results of level sets methods.* The final segmentation shown as a green overlay was obtained with the same set of parameters for both images. The level set implementation used in this example uses multiple parameters that have to be hand-tuned for each object the user wish to segment. As shown on this figure, the segmentation can easily spill over the image boundaries.

the boundaries of the desired objects. Two types of forces are usually considered. The internal forces are computed from the curvature and are designed to keep the model smooth during the deformation process. The external forces, which are computed from the underlying image data, are defined to move the model toward an object boundary or other desired features within the image.

Several approaches within the active contour framework have been developed over the years. The snakes algorithm [58] uses an explicit parametric representation of the curve which makes it robust to noise and boundary gaps as it constrains the extracted boundaries to be smooth. However, it also reduces the degree to which the curve can adapt as no splitting or merging is allowed. In contrast, the implicit active contours or level sets [106], represent the curve as the zero level-set of a characteristic function, which allows them to easily change topology and incorporate region-based statistics. Unlike the parametric form, they are not robust to boundary gaps and suffer from several other deficiencies as well [128].

While active contours and level sets have been successfully applied to many medical imaging problems [107], they suffer from two important limitations: each object requires individual initialization and each contour requires a shape prior that may not generalize well to variations in the target objects. EM image stacks contain hundreds of mitochondria, which vary greatly in size and shape. Proper initialization and definition of a shape prior for so many objects is problematic. As shown in Figure 1.3, an improper initialization of the parameters can easily lead to bad results.

1.3.2 Graph-Partitioning approaches

In recent years, graph partitioning approaches to segmentation have become popular. They produce state-of-the-art segmentations for 2D natural images [28, 120], generalize well, and unlike level sets and active contours, their complexity is not affected by the number of target objects. In 2010, the top two competitors [21, 37] in the Visual Object Classes (VOC) segmentation challenge [26] relied on such techniques. Graph partitioning approaches minimize a global objective function defined over an undirected graph whose nodes correspond to pixels, voxels, superpixels, or supervoxels; and whose edges connect these nodes [2, 14, 18]. The energy function is typically composed of two terms: the *unary term* which draws evidence from a given node, and the *pairwise term* which enforces smoothness between neighboring nodes. Some works introduce supplementary terms to the energy function, such as a term favoring cuts that maximize the object's surface gradient flux [65]. This alleviates the tendency to pinch off long or convoluted shapes, which is important when tracking elongated processes [100]. However, as noted in [59], it cannot entirely compensate for weakly detected membranes and further terms may have to be added. Another shortcoming of standard graph partitioning methods, discussed in further details in Chapter 3, is that most do not consider the shape of the segmented objects. An exception is the Textonboost approach [123]. In contrast to using explicit models to encode object shape they used a boosted combination of texton features which jointly modeled shape and texture. They combine the result of textons with color and location-based likelihood terms in a graph-partitioning approach.

In Chapter 2, we will see how graphical models can be coupled with graph partitioning methods. One type of graphical model commonly encountered in computer vision is the Markov random field (MRF) and a notable variant known as the conditional random field (CRF). Both variants are described in detail in Chapter 2. Graphical models are multivariate statistical models defined on graphs. The graph nodes correspond to random variables and are associated with a probabilistic distribution which corresponds to the unary term of the energy function described previously. The edges are used to represent the interaction between the random variables which correspond to the pairwise term in the energy function. Higher degree interaction terms have also been considered and have been shown to improve results for the problem of multi-class object segmentation [64, 118] but existing approaches still define higher-order potentials

on pre-computed image regions. The accuracy of these approaches is thus restricted by the performance of the method used to compute the regions on which the model operates. A recent approach [69] showed promising results with a fully pairwise connected graphical model defined directly on pixels instead of image regions, but the use of high-order potential at the pixel level is still impractical due to the complexity of inference.

1.3.3 Classification based approaches

Machine learning-based approaches are among the most popular approaches for image segmentation. They have been successfully coupled with level sets or graph partitioning approaches and the combination with graph partitioning approaches currently yields state-of-the art results on most standard benchmark datasets [37].

Machine learning focuses on making prediction from some input data based on known properties extracted from the training data. Machine learning algorithms can be organized into several groups. Supervised learning methods generate a function that maps the input data to desired outputs. The inferred function is also called a classifier for discrete outputs or a regression function for continuous outputs. Examples of supervised learning algorithms include Support Vector Machines (SVMs), neural nets, logistic regression, naive Bayes, memory-based learning, random forests, decision trees, bagged trees, boosted trees and boosted stumps. Other categories of machine learning algorithms include unsupervised learning that tries to find hidden structures in the input data or semi-supervised learning that combines both labeled and unlabeled examples to generate an appropriate mapping function.

Another distinction commonly made in machine learning is between generative and discriminative algorithms. We here consider a set of input feature vectors $\boldsymbol{x}_i \in \Omega$ and associated labels $y_i \in Y$, $i = \{1, \ldots, n\}$. In case of binary classification $Y = \{-1, 1\}$, each sample i is classified as being negative or positive. Generative models consider the joint distribution $p(x, y)$ by trying to model both the likelihood $p(y|x)$ and the prior $p(x)$. On the other hand, discriminative models assume that the prior distribution is not relevant and provides a model of the posterior $p(y|x)$ directly.

An example of a discriminative approach is the SVM algorithm that finds a hyperplane to separate distinct classes so that the distance between the closest data point and the hyperplane (margin) is maximized. The decision function can be expressed as

$f(x) = w^T x + b$ where w is a vector that weights the features in the feature space, and b is a threshold that shifts the hyperplane. The output of SVM can be expressed as the sign of the distance function $f(x)$ (either **1** or **-1** according to which side of the plane x lies on). The training of an SVM reduces to an optimization problem that can be solved using a Quadratic Programming (QP) solver.

Whereas the original problem may be stated in a finite dimensional space, it often happens that the sets to be discriminated are not linearly separable in that space. A common method for solving this problem is to map the original finite-dimensional space to a much higher-dimensional space, with the goal of making the separation in this space easier. In order to learn a correct mapping function, one needs a way to compute the similarity between training instances in the feature space. A linear SVM uses the euclidean distance in the feature space to find a linear separation between the different classes. The similarity metric used in the mapped high-dimensional space is defined in terms of kernel functions. Classical kernels include the polynomial kernel or the Gaussian kernel. This mapping to a high-dimensional space substantially increases the dimension of the feature space which could lead to a much higher computational complexity. Kernel methods circumvent this issue by using an efficient way to compute the similarity between objects known as the "kernel trick". This trick allows us to compute the similarity as a simple dot product without even having to carry the mapping to the high-dimensional space. We refer the reader to [48] for a comprehensive review of kernels.

1.3.4 Structured prediction models

Most common learning approaches such as SVM consider the output labels (or pixels) as being independent. In contrast, a different approach known as graphical models capture the interactions between image pixels or image regions. As explained in Section 1.3.2, graphical models represent a factorization of the joint probability of a set of random variables. While graphical models, such as Markov random fields and conditional random fields, are very attractive for these tasks due to their ability to represent the inter-dependency between variables, efficient learning of such models remains a major challenge, especially at large scales. The most common method used to estimate the parameters of graphical models is Maximum likelihood. This training procedure

chooses parameter values such that the logarithm of the likelihood, known as the log-likelihood, is maximized. The resulting function is concave, guaranteeing convergence to the global maximum, but computing its derivative is intractable for loopy graphs like the ones encountered in most computer vision applications. Approximation methods such as Pseudolikelihood [13] exist, but recent attention has focused on structured prediction methods, which combine the modeling flexibility of graphical models with the training capabilities of supervised classification methods.

A structured prediction model is a factorization or decomposition of structures into parts (nodes, edges or other parts of the structure). It is associated with a scoring function over a set of combinatorial structures. This scoring function is parametrized in terms of the weights associated with each part. Learning this parametric scoring function is an important task. One common supervised technique involves assigning to the correct prediction a score higher than the scores of all the other possible predictions. The most popular regularization function to avoid over-fitting is the squared Euclidean norm. Putting this all together, the problem of finding the parameters of the model can be posed as a quadratic optimization problem known as structured support vector machine (SSVM) [136]. Due to the high number of constraints, this problem is usually solved with a cutting-plane algorithm that involves repetitively finding the highest scoring structure given a current set of parameters. Further details will be presented in Section 2.3.4.

1.4 Medical imaging

The segmentation problem has many applications in medical research. This thesis will mostly focus on biomedical applications but we will show that the methods we developed can also be generalized to other types of images, particularly the ones of the Pascal VOC Challenge or the MSRC dataset. The type of biomedical images we will be dealing with in this thesis were acquired with an electron microscope.

Electron microscopy (EM) is an invaluable tool for mapping the morphology of neural structures. Recent techniques, such as *Focused Ion Beam Scanning Electron Microscopy* (FIB-SEM) depicted in Figure 1.4 can now deliver image stacks at the nanometer resolution in all three dimensions, such as those depicted by Figure 1.5. Such stacks show very fine structures that are critical to unlocking new insights into

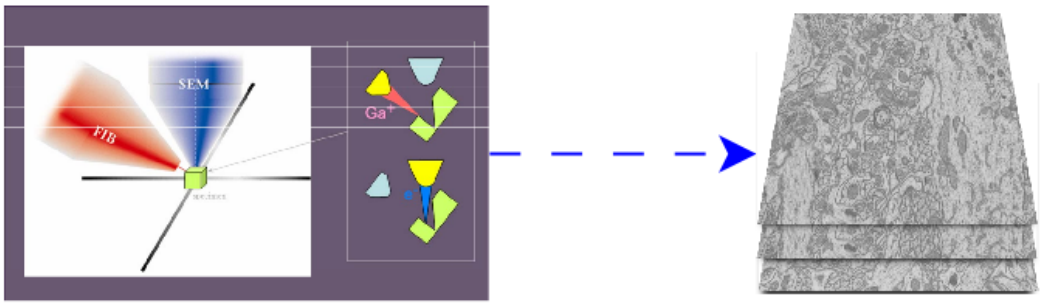


Figure 1.4: *The FIB-SEM (Focused Ion Beam - Scanning Electron Microscope)* technique uses a focused ion beam to create a cut (increments of 20 nm and greater than 100 μm in width) at a designated site in the specimen (a rat brain).

brain function, but are still mostly analyzed by hand, which can require months of tedious labor [95]. As a result, the vast majority of this very high quality data goes unused. Furthermore, although they contain tens of millions of voxels, these stacks span volumes smaller than $10 \times 10 \times 10$ μm, which presents less than a billionth of the volume of the entire mouse brain. If it is ever to be mapped in its entirety, automation will be required.

Manual segmentation remains dominant in part because most state-of-the-art automated algorithms that are reported in the computer vision literature perform well on standard natural image benchmarking data sets such as the Pascal VOC data set [26], but much less well when applied to EM imagery. Furthermore, many automated segmentation algorithms specifically designed to handle EM images tend to work on individual image slices [59, 100] because other EM modalities, such as *Transmission Electron Microscopy* (TEM), deliver image stacks with much lower resolution across slices than within them. As a consequence, they rarely take full advantage of the consistency in all three dimensions. Neither do they usually take into account global 3D geometric constraints.

1.5 Contributions of this thesis

So far, we have stressed the importance of different components for succesfully segmenting images such as designing discriminative image features or constructing a model faithful to the real world while still being tractable. The biomedical application ad-

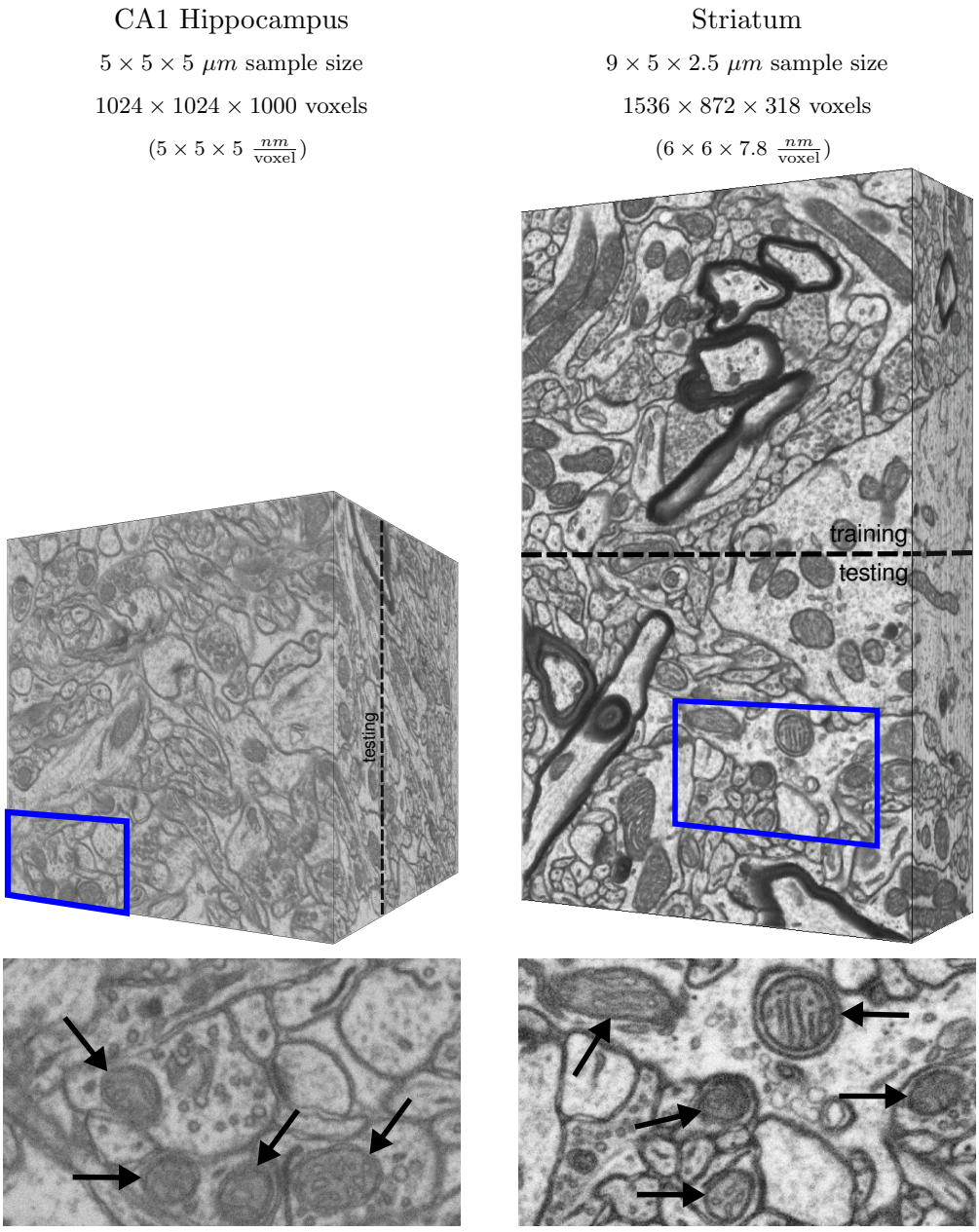


Figure 1.5: *FIB-SEM data sets.* The top row contains 3D image stacks acquired using FIB-SEM microscopy. Details in the bottom row are taken from the blue boxes overlaid on the stacks. Mitochondria, which we wish to segment, are indicated by black arrows. The high resolution allows neuroscientists to see important details but poses unique challenges. FIB-SEM image stack dimensions are orders of magnitude larger than conventional images, which limits the usefulness of many state-of-the-art segmentation algorithms, as discussed in Section 3.4.4.1. Further complicating the problem is the presence of numerous objects with distracting shapes and textures, including vesicles and various membranes. Finally, we can not rely on strong contrasts to indicate object boundaries. Note that the Striatum data is split into training and testing sections, denoted by a dashed line. A separate training stack is used for the CA1 Hippocampus (not shown).

dressed in this thesis implies additional constraints and imposes high quality requirements in order to be useful for a biological study. Our attempt to fulfill all these requirements led us to several innovations that are briefly described in this section and will be further discussed in the following chapters.

SLIC Image segmentation is closely related to the clustering problem, which aims to partition image pixels into clusters, also referred to as superpixels. Superpixel algorithms group pixels into perceptually meaningful atomic regions, which can be used to replace the rigid structure of the pixel grid. They capture image redundancy, provide a convenient primitive from which to compute image features, and greatly reduce the complexity of subsequent image processing tasks. Superpixels have become key building blocks in many computer vision algorithms, such as top scoring multi-class object segmentation entries to the Pascal VOC Challenge [34, 38, 150], depth estimation [156], segmentation [85], body model estimation [97], and object localization [34]. A first contribution made in this thesis is a joint work with R. Achanta [5] et al. that led to the development of a new superpixel algorithm named *simple linear iterative clustering* (SLIC), which adapts k-means clustering to generate superpixels in a manner similar to [156]. It is also a very fast alternative to other superpixel algorithms and yields state-of-the-art results on different image segmentation datasets. In Chapter 3, we present an extension of SLIC to generate supervoxels for 3D volumes. Given the sheer size of the 3D volumes acquired with electron microscopes, the use of supervoxels is of tremendous importance to reduce both the computational complexity and memory footprint of any segmentation algorithm.

Learning the shape of complex objects We have already discussed the importance of extracting discriminative image features that can be leveraged by a machine learning algorithm to learn a meaningful representation of certain object classes. This thesis will introduce a new kind of shape signature called "ray features" that are well-suited to the task of providing non-local shape information because they can be computed efficiently, and provide a compact description without requiring an explicit shape template. A detailed description of the ray features is given in Chapter 3, along with experimental results showing the gain observed on two different EM datasets.

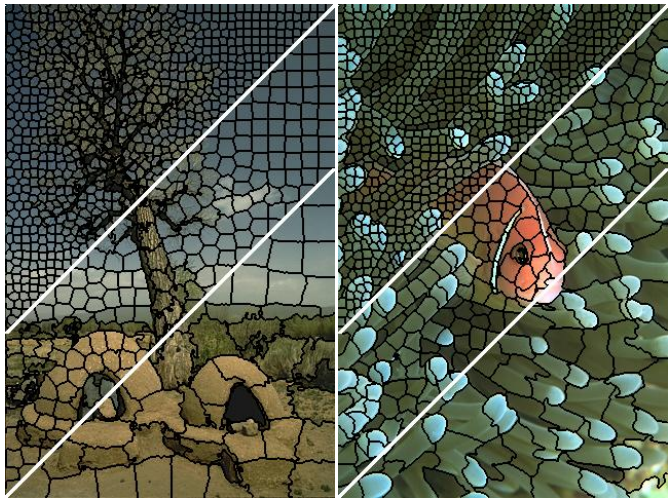


Figure 1.6: Images segmented using SLIC into superpixels of size 64, 256, and 1024 pixels (approximately).

Kernelized features A significant amount of work in the computer vision community has been focused on constructing discriminative features and training structured prediction models that offer the ability to add constraints on the expected output of the segmentation. Kernel methods have also proved very effective for leveraging image features but their application with structured prediction models, while being theoretically possible, suffers from a very high computational complexity. In Chapter 4, we propose to map features to an explicit high-dimensional space so that a linear SSVM framework can leverage the power of non-linear kernels without incurring their high computational cost.

Learning for structured prediction using stochastic descent with working sets In Section 1.3.4, we have seen that training a structured model requires repetitively finding the highest scoring structure, which is equivalent to finding the most violated constraint in a very large set of constraints. We instead propose a working set-based approximate subgradient descent algorithm to minimize the margin-sensitive hinge loss arising from the soft constraints in max-margin learning frameworks, such as the structured SVM. We focus on the setting of general graphical models, such as loopy MRFs and CRFs commonly used in image segmentation. In these cases, exact inference is intractable and the most violated constraints can only be approximated,

voiding the optimality guarantees of the structured SVM's cutting plane algorithm as well as reducing the robustness of existing subgradient based methods. We show that the proposed method obtains better approximate subgradients through the use of working sets, leading to improved convergence properties and increased reliability. Furthermore, our method allows new constraints to be randomly sampled instead of computed using the more expensive approximate inference techniques such as belief propagation and graph cuts, which can be used to reduce learning time at only a small cost of performance. We demonstrate the strength of our method empirically on the segmentation of a new publicly available electron microscopy dataset as well as the popular MSRC data set and show state-of-the-art results.

In the following chapters, we will see how all the innovations described in this section can be combined into a system that can successfully segment both natural and biomedical images. A final and very important contribution of this thesis will be discussed in the last chapter where we show that sufficiently good segmentation results can be used for the analysis of neural structures.

CHAPTER

TWO

BACKGROUND ON GRAPHICAL MODELS

Probability theory is an essential tool to study real world problems that intrinsically involve uncertainty. A good example is the roll of a 6-sided fair die. In probability theory, the space of possible outcomes is denoted $\Omega = \{1, 2, 3, 4, 5, 6\}$ and we talk about an event as a subset of Ω. For example, the event $\{1\}$ represents the case where the die shows 1. We also associate probabilities to each event that determine the degree of confidence we have for this event to happen. For the 6-sided fair die example considered above, each face has a probability of $\frac{1}{6}$.

Another important concept in probability theory is the concept of a random variable. Formally, a random variable is a function defined over the space of possible outcomes. For the die example, the random variable x can take any value in the set Ω and the probability of each event is defined as $P(x) = \frac{1}{6}, \forall x = \{1, \ldots, 6\}$.

Let's now consider an example where one rolls a die twice. The number of possible outcomes to consider is now $6 \times 6 = 36$. We can associate a random variable to each throw and study the joint probability associated to each possible outcome. One might also be interested in marginal probabilities where only a subset of the variables is retained, marginalizing over the distribution of the variables being discarded. We can for example ask about the probability of getting a 1 for the first throw which is computed by marginalizing over all outcomes for the second throw which would give us a probability equal to $\frac{1}{6}$.

We now come to the key concept of probabilistic graphical models presented in this chapter that can be used to describe a probability distribution with a graphical

representation. Let's consider a probability distribution $p(x)$ with a large number of random variables $x = \{x_1, \ldots, x_i, \ldots, x_n\}$ that makes $p(x)$ a hard function to learn. The two extremes are :

- Consider the full joint distribution:
  $p(x) = p(x_1, x_2, \ldots, x_n) = p(x_1|x_2, \ldots, x_n)p(x_2, \ldots, x_n)\ldots$
- Consider all the variables as independent:
  $p(x) = p(x_1)p(x_2)\ldots p(x_n)$

Another solution is to consider some dependencies that can be expressed with a graphical model. A graphical model defines a graph that comprises nodes representing random variables and edges expressing probabilistic relationships between the variables (missing edges imply conditional independence). The benefits of graphical models are a compact representation of a joint probability distribution and the induction of efficient inference algorithms. There exists many different types of graphical models but this chapter will mainly focus on Markov random fields (MRF) [12] and conditional random fields (CRF) [77] widely used in segmentation.

2.1 Markov random fields

A Markov random field (MRF), also known as a Markov network or an undirected graphical model [61], has a set of nodes $x = \{x_1, \ldots, x_n\}$ and a set of edges connecting pairs of nodes. Markov random field theory provides a convenient and consistent way of modeling context dependent entities such as image pixels and other spatially correlated features. This is achieved through characterizing mutual influences among such entities using MRF probabilities. The practical use of MRF models is largely ascribed to the equivalence between MRFs and Gibbs distributions established by the Hammersley-Clifford theorem [55]. This theorem gives necessary and sufficient conditions under which a probability distribution is a valid MRF. It states that a probability distribution that has a positive mass or density satisfies one of the Markov properties with respect to an undirected graph G if and only if it is a Gibbs random field, that is, its density can be factorized over the cliques of the graph[1]. Then the joint distribution is written as

[1] A clique C is defined as a subset of nodes x_c in a graph such that there exists a link between all pairs of nodes in the subset.

a product of potential functions $\phi_C(x_c)$ over the maximal cliques of the graph denoted by $c \in C$.

$$p(x) = \frac{1}{Z} \prod_{c \in C} \phi_C(x_c), \tag{2.1}$$

where Z is a normalizing constant also called the partition function:

$$Z = \sum_x \prod_{c \in C} \phi_C(x_c). \tag{2.2}$$

The choice of the potential function is still an open question. The exponential distribution is often preferred as it is the maximum-entropy distribution consistent with given constraints on expected values. Many common distributions used in statistical modeling, such as the Gaussian, Gamma or Beta distributions are in the exponential family.

The potential functions that belong to the exponential family are of the form:

$$\phi_C(x_c) = \exp\{-E(x_c)\}, \tag{2.3}$$

where $E(x_c)$ is called an energy function.

This representation was first developed in statistical physics where the graph nodes correspond to particles that are described by a spin. Boltzmann's law expresses probability functions in terms of the energy of the system E and the temperature T as:

$$p(x) = \frac{1}{Z} \exp\left\{\frac{-\sum_{c \in C} E(x_c)}{T}\right\} \tag{2.4}$$

A type of MRF that arises in many contexts is that of pairwise MRF, representing distributions with factors over single variables or pairs of variables. Pairwise MRFs are attractive because of their simplicity, and because interactions on edges are an important special case that often arises in practice.

The energy of the system $E(x) = E(x_1, \ldots, x_n)$ for such pairwise MRF is defined as:

$$\begin{aligned} E(x) &= \overbrace{E_d(x)}^{\text{data term}} + \overbrace{E_p(x)}^{\text{prior term}} \\ &= \sum_i E_i(x_i) + \sum_{i,j} E_{ij}(x_i, x_j) \\ &= \sum_i h_i(x_i) + \sum_{i,j} J_{ij}(y_i, y_j) \end{aligned} \tag{2.5}$$

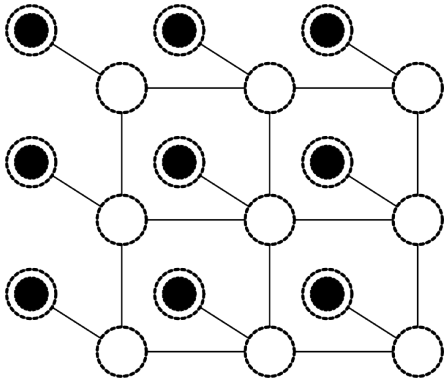


Figure 2.1: *An example of an MRF with pairwise potentials.* The hidden nodes y_i are denoted with white circles and the observed nodes x_i are denoted with filled circles.

where $J_{ij}(x_i, x_j)$ encodes the intuition that neighboring nodes are likely to belong to the same class. In statistical physics, the data term h_i describes an external magnetic field. An example of data term for images will be presented in Section 2.2.2.

2.1.1 Example: a 2D MRF for images

In this example, we consider two sets of random variables $x = \{x_1, \ldots, x_n\}$ and $y = \{y_1, \ldots, y_n\}$ where x_i denotes the observed variables (for example the value of pixel i) and $y_i = \{0, 1\}$ is a binary variable denoting the state of pixel i.

Given these two sets of variables, Equation 2.1 can now be written as :

$$p(x,y) = \frac{1}{Z}\prod_C \phi_C(x_c, y_c) \tag{2.6}$$

$$= \frac{1}{Z}\prod_C \exp\{-E(x_c, y_c)\} \tag{2.7}$$

$$= \frac{1}{Z}\exp\{\sum_C -E(x_c, y_c)\}, \tag{2.8}$$

where $Z = \sum_{(x,y)} \exp\{\sum_C -E(x_c, y_c)\}$.

Note that there exist two types of cliques in the graphical model of Figure 2.1. The first type relates two hidden nodes y_i and y_j and is associated to the prior term. The second type relates a hidden node y_i and an observation node x_i and is associated to

the likelihood term.

$$p(x,y) = \frac{1}{Z}\prod_{i,j}\overbrace{\psi_{i,j}(y_i,y_j)}^{\text{prior}}\prod_{i}\overbrace{\phi_i(y_i,x_i)}^{\text{likelihood}} \tag{2.9}$$

The pairwise potential $\psi_{i,j}(y_i, y_j)$ is typically defined as a function of the difference between the states y_i and y_j.

$$\psi_{i,j}(y_i,y_j) = \rho(y_i - y_j) \tag{2.10}$$

A popular ρ function for binary segmentation problems is based on the Ising model that will be discussed later on. Truncated linear models and truncated quadratic models have also been widely used. For applications like image denoising, the labels y_i correspond to pixel intensities. In that case, an intuitive interpretation of the pairwise potential is that it approximates first order image derivatives.

2.1.2 Labeling problem

The labeling problem consists in assigning a label to each graph node and is often referred to as MAP-MRF problem [35]. It can be solved by maximizing the posterior distribution $p(y|x)$ and the solution is usually referred to as the maximum a posterior (MAP) solution. Since the paper of Geman et al. [35] published in 1984, numerous vision problems have been formulated using this framework. This section reviews related concepts and derives involved probabilistic distributions and energies. For more detailed materials on Bayes theory, the reader is referred to [83].

Notations We are now given $p(y, x)$ in terms of an energy function and the observations x and we want to evaluate the hidden variables y. This process is called an *inference task*.

The posterior distribution $p(y|x)$ is given by applying Bayes' rule :

$$\overbrace{p(y|x)}^{\text{posterior}} = \frac{\overbrace{p(x|y)}^{\text{likelihood}}\overbrace{p(y)}^{\text{prior}}}{\underbrace{p(x)}_{\text{normalization constant}}} \tag{2.11}$$

where $p(x) = \int_y p(x|y)p(y)$ is the normalization constant to make the distribution integrate to 1.

2.1.2.1 MAP inference

To find the most likely solution y^*, we can maximize the posterior term, which is called the maximum a posteriori estimation (MAP).

$$\begin{aligned} y^* &= \arg\max_{y \in \mathcal{Y}} p(y|x) = \arg\max_{y \in \mathcal{Y}} p(x, y) && (2.12) \\ &= \arg\max_{y \in \mathcal{Y}} \frac{1}{Z} \exp\{-E(x, y)\} = \arg\max_{y \in \mathcal{Y}} \exp\{-E(x, y)\} && (2.13) \\ &= \arg\min_{y \in \mathcal{Y}} E(x, y) && (2.14) \end{aligned}$$

Since the different number of configurations $|\mathcal{Y}|$ is in general very high, solving the MAP problem is NP-hard. For some simple networks (i.e. chains or trees), the computation can be done exactly either by summing over all the possibles states or by using dynamic programming methods. Otherwise, an approximate solution to the MAP inference can be computed by coordinate descent, minimizing each unknown iteratively (Iterated Conditional Modes or ICM is an example of such algorithm). This chapter also presents other inference algorithms such as belief propagation and graph-cuts.

Finally, it is worth pointing out that maximizing $p(y|x)$ seeks the exact true labeling. In some cases, a more reasonable approach would seem to be to seek the labeling with the fewest expected number of errors, i.e. choosing $y_i = \arg\max_{y_i} p(y_i|x)\ \forall_i$. This is known as "maximum expected accuracy" [41] or "maximum posterior marginal" inference [93].

2.1.2.2 Parameter learning

In this section we discuss how to estimate the set of parameters of a MRF denoted w. Parameter estimation is typically performed by maximizing the following data likelihood:

$$w^* = \arg\max_{w} p(y, x|w) \tag{2.15}$$

A maximum likelihood estimator coincides with the most probable Bayesian estimator given a uniform prior distribution on the parameters. However, if prior information about the parameters w is available, one could also maximize the joint probability $p(y, w|x) = p(y|x, w)p(w)$, which corresponds to MAP estimation.

2.1.3 Conditional random fields

MRFs suffer from two key limitations with respect to the labeling problem. The first drawback concerns their locality. Generally, due to the complexity of MAP inference (see Section 2.1.2.1) and parameter estimation (see Section 2.1.2.2), only local relationships between neighboring nodes are usually included in the MRF energy function. Hierarchical MRFs offer one way of capturing label relationships at different scales. The second drawback of MRFs lies in their generative nature by which they explicitly attempt to model a joint probability distribution $p(y, x)$ over inputs and outputs. Although there are advantages to this approach, it also has important limitations. Not only can the dimensionality of x be very large, but the features can have complex dependencies, so constructing a probability distribution over them is difficult. In order to construct a generative model, many image samples are needed and resources have to be devoted to learn a prior model, while we are only interested in estimating the posterior over labels given the observed image. This motivates the use of conditional random fields (CRF) proposed by Lafferty et al. [77] that directly model the conditional distribution $p(y|x)$.

$$p(y|x) = \frac{1}{Z(x)} \prod_{i,j} \overbrace{\psi_{i,j}(y_i, y_j, x_i, x_j)}^{\text{prior}} \prod_i \overbrace{\phi_i(y_i, x_i)}^{\text{likelihood}} \tag{2.16}$$

where $Z(x) = \sum_y \exp\{\sum_C -E(x_c, y_c)\}$.

Unlike MRFs, the partition function for CRFs is a function of the input x and sums over all possible labels y. One of the essential differences between MRFs and CRFs lies in the training procedure. As explained in Section 2.1.2.2, learning the parameters of a MRF involves maximizing the likelihood $p(x, y|w)$, while CRFs consider the conditional likelihood $p(y|x, w)$ instead. Further details concerning the training procedure of CRFs can be found in Section 2.3.

2.2 Inference

This chapter discusses the concept of statistical inference that consists in drawing conclusions about the distribution of a random variable. We here present inference to solve the labeling problem introduced in Section 2.1.2 and consisting in assigning a label y to each graph node.

2.2.1 Variational methods

The phrase "variational" is an umbrella term that refers to various mathematical tools for optimization-based formulations of problems. Variational methods are used to determine the function that extremizes a functional. The extremal functions are solutions of the Euler-Lagrange equations that are obtained by setting the first variational derivatives of the functional F with respect to each function equal to zero. Variational methods are of general utility for many domains, such as physics or engineering. This section will provide a description of variational methods as an analytical approximation to the posterior probability of the unobserved variables y, in order to do statistical inference over these variables. The general idea is to convert the inference problem into an optimization problem that can then be relaxed in order to get an approximate solution.

In variational inference, the posterior distribution $p(y|x)$ over a set of unobserved variables denoted y given some data x is approximated by a variational distribution denoted $q(y)$. The distribution $q(y)$ is restricted to belong to a family of distributions of simpler form than $p(y|x)$, selected with the intention of making $q(y)$ similar to the true posterior. In order to make these two distributions similar, we minimize a functional of the two functions p and q that says how close they are. This functionnal denoted as $KL(q||p)$ is called the Kullback-Leibler (KL) divergence and belongs to the family of f-divergences that measures the distance between probability distributions. The KL divergence is written:

$$\begin{aligned} KL(q||p) = & -\sum_y q(y) \log \tfrac{p(y|x)}{q(y)} \\ = & \sum_y q(y) \log \tfrac{q(y)}{p(y|x)} \\ = & \sum_y q(y) \log \tfrac{q(y)p(x)}{p(y,x)} \\ = & \sum_y q(y) \log \tfrac{q(y)}{p(y,x)} + \log p(x). \end{aligned} \tag{2.17}$$

The KL divergence is always non-negative and is equal to zero when the distibutions p and q are identical. It is a non-symmetric measure and $KL(p||q)$ is in general more difficult to optimize as the averaging over p is hard to evaluate.

Equation 2.17 leads to the following decomposition:

$$\log p(x) = KL(q||p) + L(q), \tag{2.18}$$

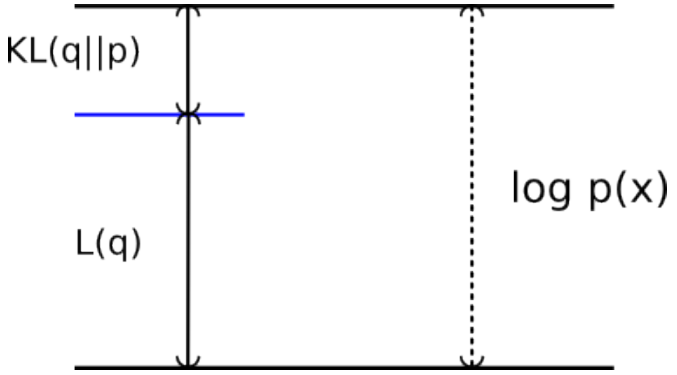


Figure 2.2: Because the KL divergence satisfies $KL(q||p) \geq 0$, we see that the quantity $L(q)$ is a lower bound on the log likelihood function $\log p(x)$.

where $L(q) = \sum_y q(y) \log \frac{p(y,x)}{q(y)}$ is a variational lower bound for $\log p(x)$ as illustrated on Figure 2.2. Note that minimizing the KL divergence is equivalent to maximizing the variational lower bound $L(q)$ with respect to the distribution $q(y)$. The lower bound becomes tight when $KL(p||q) = 0$.

Variational free energy This section will focus on one particular exponential family, namely the Boltzmann distribution which is written as $p(y|x) = \frac{1}{Z} \exp\{-E(y;x)\}$ [1]. Minimizing Equation 2.17 is intractable since evaluating $p(y|x)$ pointwise is hard as it requires computing the normalization constant Z.

[1] Boltzmann's Law is often includes a variable β that is inversely proportional to the temperature. The posterior is then written $p(y|x) = \frac{1}{Z} \exp\{-\beta E(y;x)\}$. The variable β controls how peaky the distribution is. For $\beta \to \infty$, it focuses on the MAP labeling while we get a uniform distribution for $\beta \to 0$.

In order to minimize the KL divergence, note the following decomposition:

$$\begin{aligned} KL(q||p) &= \sum_y q(y) \log \frac{q(y)}{p(y|x)} && (2.19)\\ &= \sum_y q(y) \log \frac{q(y)Z(x,y)}{\exp\{-E(y;x)\}} && (2.20)\\ &= \sum_y q(y) \log \frac{q(y)}{\exp\{-E(y;x)\}} + \sum_y q(y) \log Z(x,y) && (2.21)\\ &= \sum_y q(y) \log q(y) + \sum_y q(y) \log \exp\{E(y;x)\} + \log Z(x,y) && (2.22)\\ &= \overbrace{\sum_y q(y)E(y;x)}^{\text{U(q)}} + \overbrace{\sum_y q(y) \log q(y)}^{\text{-H(q)}} + \log Z(x,y). && (2.23) \end{aligned}$$

The third and fourth lines follow from the properties of the log function ($\log ab = \log a + \log b$ and $\log \frac{a}{b} = \log a - \log b$) and the following equality: $\sum_y q(y) = 1$.

The variational free energy $F(q)$ is equal to:

$$F(q) = U(q) - H(q). \tag{2.24}$$

where $U(q)$ is the average energy and $H(q)$ is the entropy. As stated earlier, evaluating the exact solution $F(p)$ is intractable but $F(q)$ is easier to compute. It should be noted that the expression $U(q) - H(q)$ is sometimes called with different names in the literature (e.g. Gibbs Free energy). In any case, the most important result is that this quantity is minimal for $U(q) = H(q)$, and is at this point equal to $\log Z$.

Update equations Our goal is minimizing the variational energy with respect to the parameters, which is equivalent to minimizing the KL divergence (or maximizing the lower bound) :

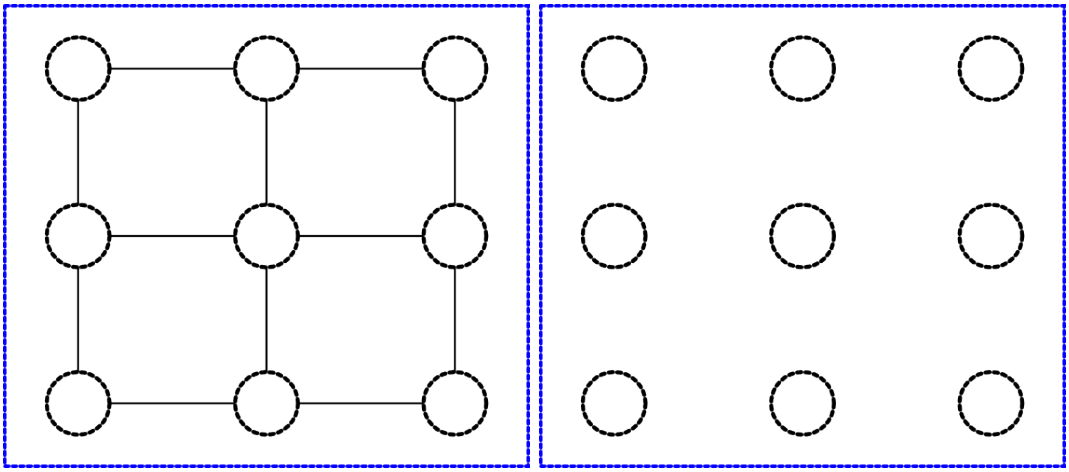
$$arg\min_q KL(q||p) = arg\min_q F(q) \tag{2.25}$$

In the next section, we will derive the optimal distribution q^* by setting $\frac{\partial}{\partial q} F(q) = 0$.

2.2.2 Mean field

2.2.2.1 Mean field approximation

The idea of mean field is to calculate the expected value of the node, conditioned on the mean values of the direct neighbors. Thus, the true distribution $p(y|x)$ is approximated



(a) Original graph. (b) Graph as approximated by mean-field.

Figure 2.3: Mean-field approximation

by a distribution $q(y)$ which has a simple form and factors over pixels:

$$q(y) = \prod_i q(y_i) \tag{2.26}$$

We then plug this factorized form in the free energy equation $F(q)$ and obtain the mean-field free energy :

$$\begin{aligned} F_{MF}(q(y_i)) &= U(q(y_i)) - TH(q(y_i)) && (2.27)\\ &= \sum_i \sum_{y_i} q(y_i)E(y_i) + T\sum_i \sum_{y_i} q(y_i)\log q(y_i) && (2.28)\\ &= -\sum_{i,j}\sum_{y_i,y_j} J_{ij}(y_i,y_j)q(y_i) - \sum_i \sum_{y_i} h_i(y_i)q(y_i) && (2.29)\\ &+ T\sum_i \sum_{y_i} q(y_i)\log q(y_i) && (2.30) \end{aligned}$$

We then compute $q(y_i)^*$ that minimizes Equation 2.30 by setting $\frac{\partial F_{MF}(q(y_i))}{\partial q} = 0$.

$$-\sum_j \sum_{y_j} J_{ij}(y_i,y_j) - h_i(y_i) + T(1+\log q_i(y_i)) = 0 \tag{2.31}$$

Then we get :

$$q_i(y_i) = \exp\left\{\frac{\sum_j \sum_{y_j} J_{ij}(y_i,y_j) + h_i(y_i) - 1}{T}\right\} \tag{2.32}$$

As explained in Section 2.2.1, $q(y_i)^*$ can also be derived by minimizing the KL-divergence $KL(q||p) = E_q(log(q) - log(p))$. By setting the derivative of the KL divergence to zero, we can show that the optimal $q(y_i)^*$ satisfies the equation [1]:

$$q(y_i)^* = \exp\left\{\beta \sum_j \sum_{y_j} q(y_j) \log \psi_i(y_i, y_j) + \beta \log \phi_i(y_i) - 1\right\} \tag{2.33}$$

2.2.2.2 Image denoising with mean field

We illustrate the use of mean field inference for the task of image denoising [2] that consists in retrieving an image $y = \{y_1, \ldots, y_n\}$ from a given corrupted image $x = \{x_1, \ldots, x_n\}$ with white noise denoted ϵ such that $x = y + \epsilon$. Each pixel in the image corresponds to a random variable $y_i = \{-1, +1\}$. We here assume a simple graph connectivity where each pixel is connected to its 4 neighbors (top, bottom, left and right).

Prior We first define a simple prior over the image y as:

$$p(y) = \frac{1}{Z_1} \exp\{-\sum_{i,j} C(y_i - y_j)^2\}, \tag{2.34}$$

where Z_1 is a normalizing constant which ensures that $p(y)$ is a true probability distribution. Here the sum is over all pixels y_i and their four neighbors y_j so the sum contributes $4C$ for each time y_i disagrees with a neighbor y_j. In other words, the constant C controls the magnitude of the penalty for pixels which disagree with their neighbors.

Likelihood We will use a very simple model to describe how observed images are generated from true images:

$$p(x|y) = \frac{1}{Z_2} \exp\{-\sum_{i} D(x_i - y_i)^2\}, \tag{2.35}$$

where Z_2 is the normalizing constant. This expression says that the observed image x is produced from the true image y by adding independent Gaussian random noise at each pixel. The constant D controls the variance of the Gaussian.

[1] The proof involves the transition from a sum over all possible random field configurations y to a sum over local configurations y_i which is given in [126]

[2] This example is adapted from Geoffrey Gordon's web page.

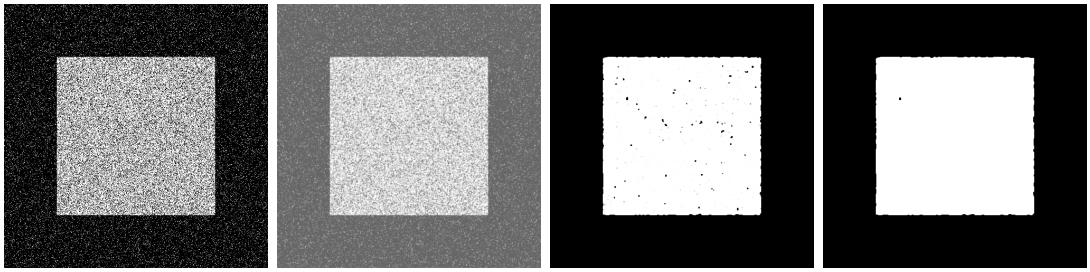


Figure 2.4: *Mean-field applied to image denoising* The left figure represent the noisy image. The other figures (from left to right) show the denoised image after 1,10 and 50 iterations.

Posterior Using Bayes rule given in Equation 2.11, we can write the posterior over images y as :

$$p(y|x) = \frac{1}{Z} \exp\{-(\sum_{ij} C(y_i - y_j)^2 + \sum_i D(x_i - y_i)^2)\} \tag{2.36}$$

Approximating the posterior Since evaluating the true posterior $p(y|x)$ is intractable, we compute an approximate distribution $q(y|x)$ using the mean field distribution which factors over pixels $q(y|x) = \prod_i q_i(y_i|x)$. The denoised image is shown in Figure 2.4.

2.2.3 Bethe approximation

The mean-field approximation described earlier makes use of one-node beliefs $q_i(y_i)$ only. The next step taken by the Bethe approximation is to introduce two-node beliefs $q_{i,j}(y_i, y_j)$ as well. This results in the following trial distribution:

$$q(y) = \prod_{i,j} q_{ij}(y_i, y_j) \prod_i q_i(y_i) \tag{2.37}$$

By combining Equation 2.37 and 2.24, we obtain the following equation for the Bethe free energy:

$$F_\beta(q) = \sum_{ij}\sum_{x_i,x_j} q_{ij}(y_i,y_j)E_{ij}(y_i,y_j) + \sum_i \sum_{y_i} q_i(y_i)E_i(y_i) \quad (2.38)$$

$$+ T(\sum_{ij}\sum_{y_i,y_j} q_{ij}(y_i,y_j)\ln q_{ij}(y_i,y_j) \quad (2.39)$$

$$- \sum_i (d_i - 1)\sum_{y_i} q_i(y_i)\ln q_i(y_i)) \quad (2.40)$$

where d_i is the degree of the node i in the graph.

This energy function can be minimized with the Belief propagation algorithm presented in the next section. Yedidia et al. [153] showed that the fixed points of Belief propagation correspond to stationary points of the Bethe free energy. While the Bethe approximation is exact for tree-like models, some concerns have been raised for graphs with loops but good empirical results have been obtained for different applications, including the image segmentation task presented in Chapter 4.

2.2.4 Belief propagation

Belief propagation can be described as a parallel message-passing algorithm where every node sends a probability density to its neighbors. Two major algorithms solving different problems have been proposed. The sum-product algorithm estimates marginals while the max-product algorithm finds the most probable joint states (MAP estimation problem).

The original Belief propagation algorithm is done in the probability domain in which the unary and pairwise potentials are related to the energy function by the following formula:

$$\phi_{ij}(y_i,y_j) = \exp\left\{-\frac{1}{T}E_{ij}(y_i,y_j)\right\} \quad (2.41)$$

$$\phi_i(y_i) = \exp\left\{-\frac{1}{T}E_i(y_i)\right\} \quad (2.42)$$

2.2.4.1 Sum-product algorithm

At each step node i sends a message $m_{ij}(y_j)$ to each neighbor j about what state node j should be in (see Figure 2.5 for an illustration). Each message is a vector of dimension

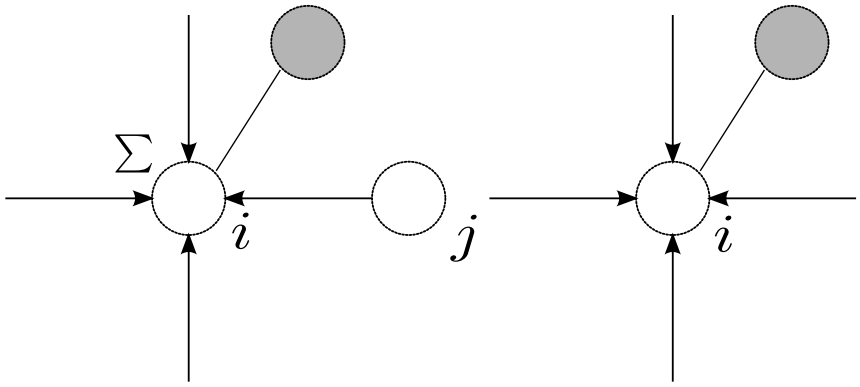


Figure 2.5: (a) Undirected graphical model representing the BP message update rule. The summation symbol indicates a sum over all the possible states of node i. The dark circle indicates the observed variable attached to the hidden node i. (b) Undirected graphical model representing the BP belief equation.

K, the number of possible labels. At each iteration, new messages are computed for each possible state in the following way:

$$m_{ij}(y_j) \leftarrow \alpha \sum_{y_i} \phi_{ij}(y_i, y_j)\phi_i(y_i) \prod_{k \in N(j) \setminus i} m_{ki}(y_i) \tag{2.43}$$

The beliefs $q_i(y_i)$ can then be computed as the product of the local evidence $\phi_i(y_i)$ and all the messages coming into node i:

$$q_i(y_i) = \phi_i(y_i) \prod_{j \in Ni} m_{ji}(y_i) \tag{2.44}$$

2.2.4.2 Max-Product algorithm

The sum-product algorithm presented in the previous section computes the marginal probabilities $p(y_i)$. Another inference problem is concerned in finding the most probable joint state y^* (MAP estimation problem) so that :

$$y^* = \arg\max_y p(y) = \arg\max_{y_1} \max_{y_2} ... \max_{y_M} p(y) \tag{2.45}$$

It can be shown that the max-product algorithm is identical to the sum-product algorithm where summations are replaced by maximizations. This algorithm is usually

defined in terms of probability distribution but equivalent forms have been proposed, minimizing log-probabilities (max-sum algorithm) or negative log-probabilities (min-sum algorithm).

2.2.5 More advanced variational algorithms

Advanced variational algorithms allow a greater accuracy but this comes at a price (running time and memory needs are usually greatly increase). For example, Generalized Belief Propagation algorithms have been developed to minimize Kicuchi free energy (Yedida, Freeman, Weiss, 2004). Note that all these algorithms rely on tractable sum-product messages, hence are limited to Gaussian Markov random fields or discrete random variables. Expectation propagation projections and Monte Carlo approximations to the sum-product messages get around these limitations, but can be unsuitable for dense graphs or can introduce extraordinary computational costs [127].

2.2.6 Graph-cuts

Graph-cuts is a popular graph partitioning approach that has been widely used in the computer vision community to solve the MAP inference problem. The algorithm formulates the labeling problem as a minimum cut of the graph (i.e. a cut of minimum weight to partition the graph). Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed graph including two special nodes denoted s (source node) and t (sink node). An s-t cut $C = (S, T)$ is a partition (or cut) of G into two disjoint subsets S and T such that $s \in S$ and $t \in T$. The max-flow min-cut theorem proves that the maximum network flow is equal to the sum of the cut-edge weights of any minimum cut that separates the source and the sink. The minimum cut problem for positive edge weights can then be solved in polynomial-time with the Edmonds-Karp algorithm.

Each node in the graph has an associated binary random variable $y_i = \{0, 1\}$. The cut in the graph creates a partition of the nodes. A total cost can also be associated to the cut and is equal to the total cost of all the edge costs.

We here use an energy function of the same form as Equation 2.6:

$$E(x, y) = \sum_i h_i(x_i) + \sum_{i,j} J_{ij}(y_i, y_j)) \tag{2.46}$$

, where h_i and J_{ij} are the data and pairwise terms described in Section 2.1.

The energy function can be reparametrized to a canonical form satisfying the following conditions:

$$J_{ij}(0,0) = J_{ij}(1,1) = 0 \tag{2.47}$$

$$J_{ij}(0,1) = J_{ij}(1,0) = \frac{D}{2} \tag{2.48}$$

$$h_i(y_i) \geq 0, \tag{2.49}$$

where $D = J_{ij}(0,0) + J_{ij}(1,1) - J_{ij}(0,1) + J_{ij}(1,0)$.

This reparametrization is valid as long as it does not change the minimum of the energy function. An obvious way to satisfy the third condition is to make $h_i(y_i)$ positive by subtracting the minimum value. In order to satisfy the first two conditions, one can add a constant to the rows or columns to transform J. The submodularity condition requires $D >= 0$. For problems considering only two labels and assuming the energy function satisfies the sub-modularity assumption, graph-cuts returns the exact solution.

2.3 Parameter Learning

This section focuses on the estimation of the parameters of a CRF, briefly mentioned in Section 2.1.2.2. A popular approach is Maximum Likelihood Estimation (MLE) and its approximation variants for loopy graphs, such as piecewise pseudo-likelihood or stochastic gradient methods. Another popular method that gained popularity recently is the structured SVM approach that couples learning and inference.

2.3.1 Maximum Likelihood Estimation

Maximum Likelihood Estimation learns the parameters of a CRF by maximizing the likelihood $p(y|x,w)$. If all the nodes follow an exponential family distribution and are observed during training, this optimization is convex. It can be solved for example using gradient descent algorithms or Quasi-Newton methods, such as the L-BFGS algorithm.

MLE seeks the optimal parameter vector w^* that satisfies:

$$w^* = \arg\max_w p(y|x,w) \tag{2.50}$$

In order to solve this equation, we make two observations. First, since the logarithm is a monotonic function, Equation 2.50 is equivalent to minimizing the negative

log-likelihood function. We then use the principle of maximum conditional likelihood that says to choose a parameter estimate that maximizes the product $\prod_i p(y_i|x_i, w)$. Note that we do not need to assume that the y_i are independent in order to justify the conditional likelihood being a product; we just need to assume that the y_i are independent conditional on x_i.

We then get:

$$\begin{aligned} w^* &= \arg\max_w L(w) = \arg\max_w \log p(y|x, w) && (2.51)\\ &= \arg\max_w \prod_i \log p(y_i|x_i, w) && (2.52)\\ &= \arg\max_w \sum_i \log p(y_i|x_i, w) && (2.53)\\ &= \arg\max_w \sum_i \log \frac{1}{Z(w)} \exp\{-E(y_i; x_i)\} && (2.54)\\ &= \arg\max_w \sum_i -E(y_i; x_i) - \log Z(w) && (2.55) \end{aligned}$$

If we assume a log-linear model, the energy $E(y; x)$ can be written as a dot product between the parameter vector w and a feature function denoted $\psi(x, y)$, i.e. $E(y; x) = -w^T\psi(x, y)$. Plugging this expression in the objective function, we get:

$$L(w) = \sum_i w^T\psi(x_i, y_i) - \log \sum_y \exp\{w^T\psi(x_i, y)\}, \quad (2.56)$$

which can be maximized by computing the derivative of $L(w)$ with respect to the parameters w:

$$\begin{aligned} \nabla_w L(w) &= \sum_i [\psi(x_i, y_i) - \frac{\sum_y \exp\{w^T\psi(x_i, y)\}\psi(x_i, y)}{\sum_y \exp\{w^T\psi(x_i, y)\}}] && (2.57)\\ &= \sum_i [\psi(x_i, y_i) - \sum_y p(x_i, y|w)\psi(x_i, y)] && (2.58)\\ &= \sum_i [\psi(x_i, y_i) - \mathbb{E}_{p(x_i,y|w)}\psi(x_i, y)] && (2.59) \end{aligned}$$

The first term in the summation is the feature function of the correct label while the second term takes the expectation over all labels. Since $L(w)$ is differentiable and convex (it has a positive definite Hessian), the gradient descent of Algorithm 1 will find the global optimum with $\nabla_w L(w) = 0$.

Algorithm 1 Steepest Descent Minimization

INPUT: tolerance ϵ, learning rate η
$w_0 \leftarrow 0$
repeat
  $v \leftarrow \nabla_w L(w)$
  $w \leftarrow w - \eta v$
until $||v|| < \epsilon$

Since overfitting issues are usually associated to MLE, it is common to find the MAP estimate by using a prior on w such as a Gaussian prior $p(w) = \exp\{-\frac{1}{2\sigma^2}\,||w||^2\}$. An alternative is a Laplace prior which leads to what is known as L1 regularization.

For general CRFs, there is still a problem with the computation of $\nabla_w L(w)$ because the number of possible configurations for y is typically (exponentially) large. We then present approximation methods that try to address this computation issue.

2.3.2 Approximation methods

As stated in the previous section, the conditional maximum likelihood leads to an intractable problem due to the computation of the partition function that involves an exponential number of terms. This section will review two approximation methods that have been proposed to address this problem: pseudo-likelihood and contrastive divergence.

2.3.2.1 Pseudo-likelihood

The idea in the pseudo-likelihood approximation is for the training objective to depend only on conditional distributions over single variables. The likelihood probability $p(y|x, w)$ is approximated as a product of individual conditional probabilities for each y_i given its neighbors.

$$p(y|x, w) = \prod_i p(y_i|y_{\mathcal{N}_i}, x, w), \tag{2.60}$$

where $\mathcal{N}_i$ are the neighbors of node i.

Because the normalizing constants for these distributions depend only on single variables, they can be computed efficiently. Assuming that the model family includes

the true distribution and in the presence of infinite amount of data, [49] showed that the pseudo-likelihood is a consistent estimator that will converge to true parameters. Although these assumptions are rarely satisfied, pseudo-likelihood was successfully applied to train CRFs [130, 145].

2.3.2.2 Contrastive divergence

Markov Chain Monte Carlo (MCMC) inference methods can be used to train a CRF by setting up a Markov chain whose stationary distribution is $p(y|x, w)$, running the chain for a number of iterations, and using the resulting approximate marginals to approximate the true marginals in the gradient $\nabla_w L(w)$. MCMC methods suffer from several limitations. First, they require a large number of iterations to reach convergence. Second, many MCMC methods, such as Metropolis-Hastings, require computing a ratio of normalizing constants $\frac{Z(x,w_1)}{Z(x,w_2)}$ for two different set of parameters w_1 and w_2. This presents a severe difficulty for models in which computing $Z(x, w)$ is intractable.

One possibility to overcome these difficulties is contrastive divergence (CD) [45] that uses MCMC but does not require convergence to equilibrium for approximating the model likelihood gradients used for learning. While CD has been mostly applied to models such as restricted Boltzmann machines, it can also be applied to CRFs[43]. Another related method is SampleRank [148] which is a supervised parameter estimation method that performs parameter updates during MCMC inference. Specifically, each pair of consecutive samples is compared, and a parameter update is made if the ranking of the model scores of the samples disagrees with the ranking implied by the labeled data.

2.3.3 Loss-sensitive training

Unfortunately, maximum likelihood and its associated approximations all suffer from the problem that the loss function under which the performance of the CRF is evaluated is ignored. One way to take into account the loss function is to augment the energy of a given training example by including the loss function for that example, producing a Loss-Augmented energy. This idea is similar to the concept of margin re-scaling in structured SVMs, a similarity that has been highlighted previously by [42] and [147].

2.3.4 Maximum-margin learning

An alternative method to training structured linear model is based on the maximum-margin Markov networks [133]. Maximum margin structured learning is a large-margin method for learning the parameters of structured output models, such as CRFs. Given the parameters $\mathbf{w}$, a structured model predicts the labeling $y \in \mathcal{Y}$ for a given input $x \in \mathcal{X}$ by maximizing some score function $S_{\mathbf{w}} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, i.e.,

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} S_{\mathbf{w}}(x, y) = \arg\max_{y \in \mathcal{Y}} \mathbf{w}^T \Psi(x, y) \tag{2.61}$$

The score is usually expressed as a linear function of $\mathbf{w}$ and can be written as $\mathbf{w}^T \Psi(x, y)$, where the vector $\Psi(x, y)$ is the *feature map* or *sufficient statistics* corresponding to the input x and the labeling y. The fundamental properties of random fields imply that the feature map $\Psi(x, y)$ and hence the score $S_{\mathbf{w}}$ decompose into sums over individual nodes and edges for any pairwise CRFs [12].

Discriminative learning uses the labeled training data to learn the CRF parameters so that the inferred labeling of the CRF is "close" to that of the ground truth, defined as yielding a low *loss*. More specifically, given a set of N training examples $\mathcal{D} = ((x^1, y^1), \ldots, (x^N, y^N))$ where $x^i \in \mathcal{X}$ is an input example, such as the image or features associated to it, and $y^i \in \mathcal{Y}$ is the associated labeling, the learning task consists in finding model parameters $\mathbf{w}$ that achieve low empirical loss subject to some regularization. In other words, we seek

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}, \mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \sum_{(x^n, y^n) \in \mathcal{D}} l(x^n, y^n, \mathbf{w}) + R(\mathbf{w}), \end{aligned} \tag{2.62}$$

where l is the surrogate loss function and $R(\mathbf{w})$ is the regularizer (typically minimizing the L2 norm). The most common choice for the surrogate loss l is the hinge loss, as used in [133, 136] and defined as:

$$l(Y^n, Y^*, \mathbf{w}) = [S_{\mathbf{w}}(Y^*) + \Delta(Y^n, Y^*) - S_{\mathbf{w}}(Y^n)]_+ \tag{2.63}$$

Note that the definition of the surrogate loss l depends on the score function $S_{\mathbf{w}}$, since the goal of learning is to make the maximizer of $S_{\mathbf{w}}$ a desirable output for the given input.

The structured SVM approach proposed in [136] formulates the parameter learning problem as a quadratic program (QP) with soft margin constraints:

$$\begin{aligned} &\min_{\mathbf{w},\xi\geq\mathbf{0}} \frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 + C\sum_{n=1}^{N}\xi_n \\ &\text{s.t. } \forall n : S_{\mathbf{w}}(x^n, y^n) \geq \max_{y\in\mathcal{Y}_n}(S_{\mathbf{w}}(x, y) + \Delta(y^n, y)) - \xi_n, \end{aligned} \tag{2.64}$$

where $\mathcal{Y}_n$ is the set of all possible labellings for example n, the constant C controls the trade-off between margin and training error, and the task loss Δ measures the closeness of any inferred labeling y to the ground truth labeling y^n.

The above quadratic program involves a very large, possibly infinite number of linear inequality constraints. In general, the number of inequalities is too large to be optimized over explicitly. Instead, a cutting plane method was proposed in [136]. This method repetitively finds a constraint that is violated and uses it to construct a working set. It then solves the problem with the constraints contained in the working set, and if this solution still does not satisfy all the constraints, another violating constraint is generated and added to the set. This continues until we have accumulated enough constraints so that all constraints are satisfied in the solution. One of the contributions of this thesis presented in Chapter 5 is a new training method based on stochastic subgradient descent.

CHAPTER

THREE

SUPERVOXEL-BASED SEGMENTATION OF MITOCHONDRIA IN EM IMAGE STACKS WITH LEARNED SHAPE FEATURES

3.1 Introduction

In Section 1.4, we emphasized the challenges posed with biomedical images due to noise and other distracting structures present in such images. In this chapter, we will present a graph partitioning method that overcomes these limitations and demonstrate its effectiveness for segmenting mitochondria in 3D EM images.

In addition to providing energy to the cell, mitochondria play an important role in many essential cellular functions including signaling, differentiation, growth and death. An increasing body of research suggests that regulation of mitochondrial shape is crucial for cellular physiology [20]. Furthermore, localization and morphology of mitochondria have been tightly linked to neural functionality. For example, pre- and post- synaptic presence of mitochondria is known to have an important role in synaptic function [78].

Mounting evidence also indicates that there is a close link between mitochondrial function and many neuro-degenerative diseases. Mutations in genes that control fusion and division events have been found to cause neurodegenerative processes [62]. For example, mutations of the gene coding for a protein kinase called PINK1, which is known to regulate mitochondrial division, have been linked to a type of early-onset Parkinson's disease [111].

3. SUPERVOXEL-BASED SEGMENTATION OF MITOCHONDRIA IN EM IMAGE STACKS WITH LEARNED SHAPE FEATURES

Unfortunately, because mitochondria range from less than 0.5 to 10 μm in diameter [19], optical microscopy does not provide sufficient resolution to reveal fine structures that are critical to unlocking new insights into brain function. Recent Electron Microscopy (EM) advances, however, have made it possible to acquire much higher resolution images, and have already provided new insights into mitochondrial structure and function [94]. The data used in this work were acquired by a focused ion beam scanning electron microscope (FIB-SEM, Zeiss NVision40), which uses a focused beam of gallium ions to mill the surface of a sample and an electron beam to image the milled face [63]. The milling process removes approximately $5nm$ of the surface, while the scanning beam produces images with a pixel size of $5 \times 5nm$. Repeated milling and imaging yielded nearly isotropic image stacks containing billions of voxels, such as the ones appearing in Figure 1.5.

Analyzing such an image stack by hand could require months of tedious manual labor [95] and, without reliable automated image-segmentation tools, much of this high quality data would go unused. This situation arises in part from the fact that most state-of-the-art EM segmentation algorithms [59, 100] were designed for highly anisotropic EM modalities, such as *Transmission Electron Microscopy* (TEM). Such data tends to have a greatly reduced resolution in the z-direction, and associated segmentation algorithms often process slices individually to deal with the missing data. Our approach processes large 3D volumes in a single step, which is advantageous for isotropic FIB-SEM stacks. More generic Computer Vision algorithms that perform well on natural image benchmarking data sets such as the Pascal VOC (Visual Object Classes) data set [26] perform poorly on EM data, whether it is isotropic or not. There are several reasons for this. The amount of data in a typical EM stack is a major bottleneck, rendering these approaches intractable both in terms of memory and computation time. Furthermore, these approaches rarely account for important shape cues and often rely only on local statistics which can easily become confused when confronted with the noise and textures found in EM data. Finally, the conventional assumption that strong image gradients always correspond to significant boundaries does not hold, as illustrated in Figure 1.5.

Our approach overcome the challenges presented by EM datasets by combining the following components.

- **Operating on supervoxels instead of voxels.** We cluster groups of similar voxels into regularly spaced *supervoxels* of nearly uniform size, which are used to compute robust local statistics. This reduces the computational and memory costs by several orders of magnitude without sacrificing accuracy because supervoxels naturally respect image boundaries.

- **Including global shape cues.** The supervoxels are connected to their neighbors by edges and form a graph. Most graph segmentation techniques rely only on local statistics to partition the graph, ignoring important shape information. We introduce features that capture non-local shape properties and use them to evaluate how likely a supervoxel is to be part of the target structure.

- **Learning boundary appearance.** EM data is notoriously complex, violating the standard assumption that strong image gradients always correspond to significant boundaries. Spatial and textural cues must be considered when determining where true object boundaries lay. We therefore train a classifier to recognize which pairs of supervoxels are most likely to straddle a relevant boundary. This prediction determines which edges of the supervoxel graph should most likely be cut during segmentation.

We demonstrate our approach for the purpose of segmenting mitochondria in two large FIB-SEM image stacks taken from the *CA1 hippocampus* and the *striatum* regions of the brain. We show that our approach performs close to the level of a human annotator and is much more accurate than a state-of-the-art 3D segmentation approach [125].

3.2 Related Work

In this section, we begin by examining previous attempts to segment mitochondria. We then broaden our discussion to include the use of machine learning techniques for other tasks in EM imagery. Finally, we discuss methods that rely on a graph partitioning approach to segmentation.

3.2.1 Mitochondria Segmentation

As discussed in the introduction, understanding the processes that regulate mitochondrial shape and function is important. Perhaps due to the difficulty in acquiring the

data, relatively few researchers have attempted to quantify important mitochondria properties in recent years. In [146], a Gentle-Boost classifier is trained to detect mitochondria based on textural features. In [101], texton-based mitochondria classification of melanoma cells is performed using a variety of classifiers including k-NN, SVM, and Adaboost. While these techniques achieve reasonable results, they consider only textural cues while ignoring shape information. A recent approach, described in in [125], using state-of-the-art features and a Random Forest learning approach for segmentation has been successfully applied to 3D EM data in [70]. We compare our approach to [125] in Section 3.4.

In [103], shape-driven watersnakes that exploit prior knowledge about the shape of membranes are used to segment mitochondria from the liver. However, this approach is adapted to anisotropic TEM data. Recently, new features have been introduced to segment mitochondria in neural EM imagery. Ray features, first introduced in [124], were applied to 2D mitochondria segmentation in [89]. Inspired by Ray features, Radon-like features were proposed in [73], but have shown to perform significantly worse than Ray features in [139].

3.2.2 Machine Learning in EM Imagery

Besides mitochondria segmentation, machine learning techniques have found their way into other tasks in EM imagery including membrane detection and dendrite reconstruction. We refer the reader to [54] for an excellent survey covering some of these applications. EM data poses unique challenges for machine learning algorithms. In addition to the large number of voxels involved, a variety of sub-cellular structures exist including mitochondria, vesicles, synapses, and membranes. As seen in Figure 1.5, these structures can be easily confused when only local image statistics are considered, especially given the often low signal-to-noise ratio of the data. This is one of the reasons why algorithms that perform well on natural images are far less successful on EM data.

While a large body of research is dedicated to segmenting axons and dendrites from EM data, only a small faction uses a machine learning approach. In [53], a Convolutional Network (CN) performs neuronal segmentation by binary image restoration. This work is extended in [52] by incorporating topological constraints. In [137], CNs are used to predict an affinity graph that expresses which pixels should be grouped together using the Rand index [115], a quantitative measure of segmentation performance. In

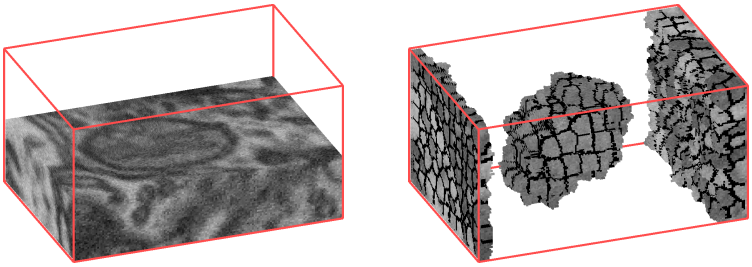


Figure 3.1: *Segmenting an image stack into supervoxels. (left)* A cropped FIB-SEM image stack containing a mitochondrion. *(right)* The cropped stack is segmented using the SLIC algorithm into groups of similar voxels called *supervoxels*. For visualization, supervoxels in the center of the image stack have been removed, leaving supervoxels belonging to the mitochondrion interior and on the caps of volume. Boundaries between supervoxels are marked in black. Notice that voxels with similar intensities are grouped while respecting natural boundaries.

another recent approach [59], a random forest classifier is used in a cost function that enforces gap-completion constraints to segment TEM slices.

Machine learning techniques have also been applied to detect membranes, a common preprocessing step in registration and axon/dendrite reconstruction. In [57], Neural Networks relying on feature vectors composed of intensities sampled over stencil neighborhoods are trained to recognize membranes in TEM image stacks. In [143], an Adaboost classifier is trained to detect cell membranes based on eigenvalues and Hessian features. A hierarchical random forest classification scheme is used to detect boundaries and segment EM stacks in [7].

3.2.3 Segmentation by Graph-Partitioning

While active contours and level sets have been successfully applied to many medical imaging problems [107], they suffer from two important limitations: each object requires individual initialization and each contour requires a shape prior that may not generalize well to variations in the target objects. EM image stacks contain hundreds of mitochondria, which vary greatly in size and shape. Proper initialization and definition of a shape prior for so many objects is problematic.

In recent years, graph partitioning approaches to segmentation have become popular. They produce state-of-the-art segmentations for 2D natural images [28, 120],

generalize well, and unlike level sets and active contours, their complexity is not affected by the number of target objects. In 2010, the top two competitors [21, 37] in the VOC segmentation challenge [26] relied on such techniques. Graph partitioning approaches minimize a global objective function defined over an undirected graph whose nodes correspond to pixels, voxels, superpixels, or supervoxels; and whose edges connect these nodes [2, 14, 18]. The energy function is typically composed of two terms: the *unary term* which draws evidence from a given node, and the *pairwise term* which enforces smoothness between neighboring nodes. Some works introduce supplementary terms to the energy function, including a term favoring cuts that maximize the object's surface gradient flux [65]. This alleviates the tendency to pinch off long or convoluted shapes, which is important when tracking elongated processes [100]. However, as noted in [59], it cannot entirely compensate for weakly detected membranes and further terms may have to be added.

A shortcoming of standard graph partitioning methods, as we will discuss in Section 3.3.3, is that most do not consider the shape of the segmented objects.

3.3 Method

The first step of our approach is to over-segment the image stack into *supervoxels*, small clusters of voxels with similar intensities. All subsequent steps operate on supervoxels instead of individual voxels, speeding up the algorithm by several orders of magnitude. This step is described in Section 3.3.1. Next, a feature vector containing shape and intensity information is extracted for each supervoxel, as described in Section 3.3.2. The final segmentation is produced by feeding the extracted feature vectors to classifiers that define the unary and pairwise potentials of a graph cut segmentation step described in Section 3.3.3. The learning procedure and a list of parameters are provided in Section 3.4.

3.3.1 Supervoxel Over-segmentation

Many popular graph-based segmentation approaches such as graph cuts [14] become exponentially more complex as nodes are added to the graph. In practice, this limits the amount of data that can be processed. EM stacks can contain billions of voxels, making such methods intractable both in terms of memory and computation time. Even

Algorithm 2 SLIC Supervoxels

/ Initialization */*

Initialize cluster centers $C_k = [I_k, u_k, v_k, z_k]^T$ by sampling voxels at regular grid steps S.

Move cluster centers to the lowest gradient position in a $3 \times 3 \times 3$ neighborhood.

Set label $l(i) = -1$ for each voxel i.

Set distance $d(i) = \infty$ for each voxel i.

repeat

/ Assignment */*

for each cluster center C_k **do**

for each voxel i in a $2S \times 2S \times 2S$ neighborhood surrounding C_k **do**

Compute distance δ_{ik} between C_k and voxel i.

if $\delta_{ik} < d(i)$ **then**

set $d(i) = \delta_{ik}$

set $l(i) = k$

end if

end for

end for

/ Update */*

Compute new cluster centers.

Compute residual error E.

until $E \leq$ threshold

/ Post-processing */*

Enforce connectivity.

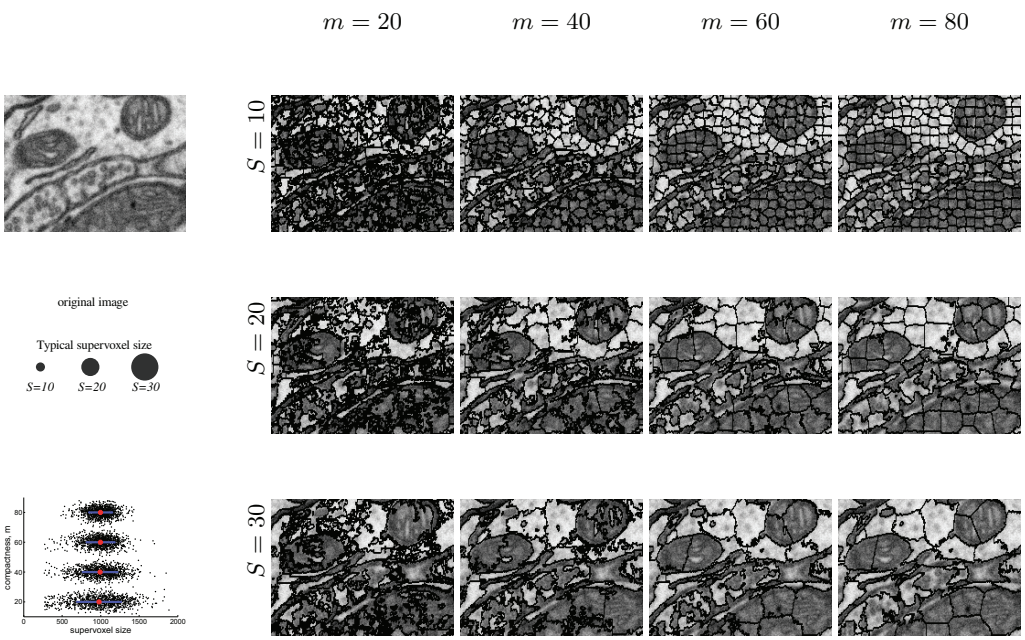


Figure 3.2: *Supervoxel size and compactness as a function of parameters m and S of Equation 3.1. (top left)* A cropped EM slice containing three mitochondria. *(middle left)* Typical supervoxels sizes for $S = 10$, $S = 20$, and $S = 30$. *(bottom left)* Standard deviation of supervoxel size as a function of varying m. *(right)* A matrix of supervoxel segmentations showing the effect of varying m and S. Increasing m produces more compact, regular supervoxels. Increasing S increases supervoxel size. Note that supervoxels are three-dimensional, yet the images above show only a two-dimensional slice of each supervoxel.

for moderately-sized stacks, standard minimization techniques [66, 68, 153] become intractable. By replacing the voxel-grid with a graph defined over supervoxels, we reduce the complexity by several orders of magnitude while sacrificing little in terms of segmentation accuracy.

To efficiently generate high-quality supervoxels, we extend our earlier superpixel algorithm, *simple linear iterative clustering* (SLIC) [5], to produce 3D supervoxels such as those depicted in Figure 3.1. The approach used in SLIC is closely related to k-means clustering, with two important distinctions. First, the number of distance calculations in the optimization is dramatically reduced by limiting the search space to a region proportional to the supervoxel size. Second, a novel distance measure combines intensity and spatial proximity, while simultaneously providing control over the size and compactness of the supervoxels.

The supervoxel clustering procedure is summarized in the table marked Algorithm 2. Initial cluster centers are chosen by sampling the image stack at regular intervals of length S in all three dimensions. The number of supervoxels k and the number of voxels in the volume N determines the length, $S = \sqrt{N/k}$. Next, the centers are moved to the nearest gradient local minimum. The algorithm then assigns each voxel to the nearest cluster center, recomputes the centers, and iterates. After n iterations, the final cluster members define the supervoxels.

SLIC is many times faster than standard k-means clustering thanks to a distance function measuring the spatial and intensity similarities of voxels within a limited $2S \times 2S \times 2S$ region

$$\delta_{ik} = \sqrt{\frac{(I_k - I_i)^2}{m^2} + \frac{(u_k - u_i)^2 + (v_k - v_i)^2 + (z_k - z_i)^2}{S^2}}, \tag{3.1}$$

where I is image intensity; u_i, v_i, and z_i are the spatial coordinates of voxel i; u_k, v_k, and z_k are those of cluster center k. Normalizing the spatial proximity and intensity terms by S and m[1] allows the distance measure to combine these quantities which have very different ranges. Simply applying a Euclidean distance without normalization would result in clustering biased towards spatial proximity. Supervoxel compactness is regulated by m. As seen in Figure 3.2, higher m values produce more compact

[1] S and m are the average expected spatial and intensity distances within a supervoxel, respectively. m can be adjusted to control compactness.

supervoxels while lower m values produce less compact ones that more tightly fit the image boundaries.

To ensure that the total number of distance calculations remains constant in N, irrespective of k, the distance calculations are limited to a $2S \times 2S \times 2S$ volume around the cluster centers. This makes the complexity $O(N)$, whereas a conventional k-means implementation would be of complexity of $O(kN)$ where N is the number of voxels.

A post-processing step enforces connectivity because the clustering procedure does not guarantee that supervoxels will be fully connected. Orphan voxels are assigned to the most similar nearby supervoxels using a flood-fill algorithm. We refer the interested reader to [4] for further details.

We found SLIC to be particularly well adapted to EM segmentation as it delivers high quality supervoxels efficiently, provides size and compactness control, and can operate on large volumes. Besides SLIC, only a few algorithms are designed to generate supervoxels. In [142], supervoxels are obtained by stitching together overlapping patches followed by optimizing an energy function using a graph cuts approach. However, this approach performs worse than SLIC in terms of segmentation quality using standard measures [4], consumes too much memory, and it is 20 times slower with a worst case complexity is $O(N^2)$. A second alternative is to apply the watershed algorithm [144] to generate supervoxels, as used in [7, 84]. However, the size and quality of the watershed supervoxels are unreliable. Finally, other popular superpixel methods could potentially be extended to 3D, including Quickshift [140], Turbopixels [81], and the method of [28]. However, these methods all produce lower quality segmentations than SLIC in 2D [4], and are orders of magnitude slower: 13, 164 and 5 times slower, respectively. They also require much more memory. These comparisons are documented in [4].

3.3.2 Feature Vector Extraction

After extracting supervoxels, the next step of the algorithm is to extract feature vectors that capture local shape and texture information. For each supervoxel i, we extract a feature vector $\mathbf{f}_i$ combining Ray descriptors and intensity histograms, written as

$$\mathbf{f}_i = [\mathbf{f}_i^{\mathrm{Ray}\top},\ \mathbf{f}_i^{\mathrm{Hist}\top}]^\top\ , \tag{3.2}$$

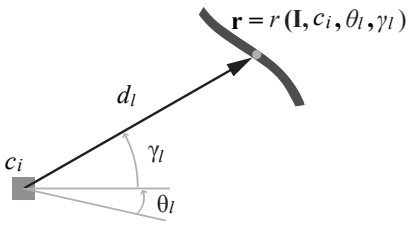


Figure 3.3: *Ray feature function* $r(\mathbf{I}, c_i, \theta_l, \gamma_l)$. All components of the Ray descriptor depend on this basic function. For a given location c_i, it returns the location of the closest boundary point $\mathbf{r}$ in direction l defined by angles (θ_l, γ_l). d_l is the corresponding distance from c_i to the boundary.

where $\mathbf{f}_i^{\text{Ray}}$ represents a Ray descriptor and $\mathbf{f}_i^{\text{Hist}}$ represents an intensity histogram. For simplicity, we omit the i subscript in the remainder of the section.

3.3.2.1 Ray Descriptors

Rays are a class of image features introduced in [124] that capture non-local shape information around a given point. We extend Ray features to 3D in this work, and propose a method for bundling a set of Ray features into a rotationally invariant descriptor. Ray features are attractive because they provide a description of the local shape relative to a given location. This formulation fits naturally into a graph partitioning framework because Rays can provide a description of the local shape for locations corresponding to every node in the graph. Descriptors commonly used for shape retrieval that rely on skeletonization or contours, including distance sets [40] and Lipschitz embeddings [46], do not have this property.

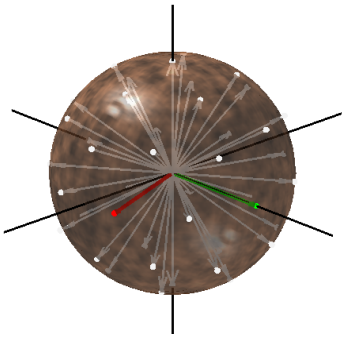
A Ray feature is computed by casting an imaginary ray in an arbitrary direction (θ_l, γ_l) from a point c, and measuring an image property at a distant point

$$\mathbf{r} = r(\mathbf{I}, c_i, \theta_l, \gamma_l) \tag{3.3}$$

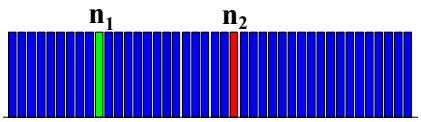
where the ray encounters an edge (depicted in Figure 3.3). In our implementation, edges are found by applying a 3D extension of the Canny edge detection algorithm [50].

For supervoxel i, we construct a *Ray descriptor* by concatenating a set of $3L$ Ray features emanating from the supervoxel center c_i, where L is a fixed set of orientations. The L orientations are uniformly spaced over a geodesic sphere, as depicted in Figure 3.4, and defined by polar angles $\mathbf{\Theta} = \{\theta_1, \ldots, \theta_L\}$ and $\mathbf{\Gamma} = \{\gamma_1, \ldots, \gamma_L\}$. The Ray

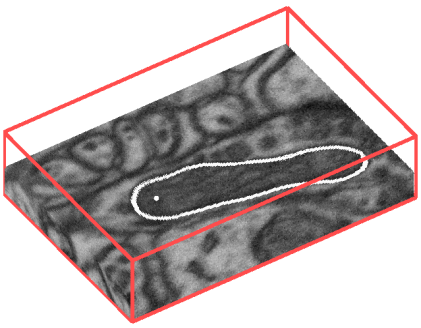
(a) L Rays cast on a geodesic sphere.



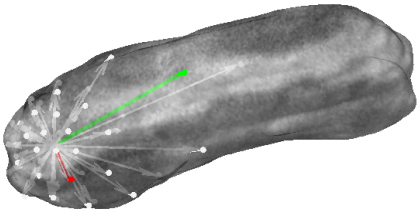
(b) The f_{ndist} descriptor ordered according to the canonical orientation defined by $\mathbf{n_1}$ and $\mathbf{n_2}$.



(c) A cropped EM image stack containing a mitochondrion. Edges appear in white.



(d) L Rays cast from c_i in the mitochondrion to the closest surface boundary. Principle axes $\mathbf{e_1}$ and $\mathbf{e_2}$ appear in green and red.



(e) The f_{ndist} descriptor re-ordered into the canonical orientation defined by $\mathbf{e_1}$ and $\mathbf{e_2}$.

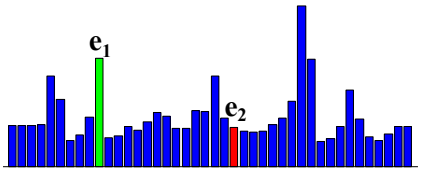


Figure 3.4: *Rotation invariant 3D Ray descriptor. (a)-(b)* depict the Ray descriptor cast from the center of a unit sphere. The two axes defining the orientation of the descriptor $\mathbf{n_1}$ and $\mathbf{n_2}$ are shown in green and red, respectively. *(c)* shows a cropped volume containing a mitochondria with boundaries highlighted in white. The white point corresponds to the location of the Ray descriptor in *(d)-(e)*. $\mathbf{e_1}$ and $\mathbf{e_2}$ are used to estimate the orientation of the descriptor and are aligned to the canonical orientation.

descriptor for supervoxel i in an image stack $\mathbf{I}$ at orientation (θ_l, γ_l) is written

$$\mathbf{f}^{\text{Ray}}(\mathbf{I}, c_i, \theta_l, \gamma_l) = [f_{\text{ndist}}, f_{\text{norm}}, f_{\text{ori}}]^\top, \tag{3.4}$$

where individual Ray features are given by

$$\begin{aligned} f_{\text{ndist}}(\mathbf{I}, c_i, \theta_l, \gamma_l) &= \frac{\|r(\mathbf{I}, c_i, \theta_l, \gamma_l) - c_i\|}{D}, \\ f_{\text{norm}}(\mathbf{I}, c_i, \theta_l, \gamma_l) &= \|\nabla\mathbf{I}(r(\mathbf{I}, c_i, \theta_l, \gamma_l))\|, \\ f_{\text{ori}}(\mathbf{I}, c_i, \theta_l, \gamma_l) &= \frac{\nabla\mathbf{I}(r(\mathbf{I}, c_i, \theta_l, \gamma_l))}{\|\nabla\mathbf{I}(r(\mathbf{I}, c_i, \theta_l, \gamma_l))\|} \cdot \frac{\mathbf{r} - c_i}{\|\mathbf{r} - c_i\|}, \end{aligned} \tag{3.5}$$

and $\nabla\mathbf{I}$ is the gradient of the image stack.

In other words, each descriptor $\mathbf{f}^{\text{Ray}}$ contains three Ray features that measure image characteristics at the nearest edge point $\mathbf{r}$ given by Equation 3.3. The features in Equation 3.5 are

- f_{ndist}, the most basic feature, simply encodes the distance from c_i to the closest edge $d_l = \|r(\mathbf{I}, c_i, \theta_l, \gamma_l) - c_i\|$. It is made scale-invariant by normalizing by D, the mean distance over all L directions,
- f_{norm}, the gradient norm at $\mathbf{r}$,
- f_{ori}, the orientation of the gradient at $\mathbf{r}$ computed as the dot product of the unit Ray vector and a unit vector in the direction of the local gradient at $\mathbf{r}$.

The final step is to align the descriptor to a canonical orientation, making it rotation invariant. It is important that the descriptor is the same no matter the orientation of the mitochondria, otherwise the learning step would have difficulty finding a good decision boundary. In Figure 3.4(a), two perpendicular axes $\mathbf{n_1}$ and $\mathbf{n_2}$ define a canonical frame of reference for the descriptor. These axes are assigned specific locations in the feature vector shown in Figure 3.4(b), and all other elements are ordered according to their angular offsets from $\mathbf{n_1}$ and $\mathbf{n_2}$. To achieve rotational invariance, we re-order the descriptor such that $\mathbf{n_1}$ and $\mathbf{n_2}$ align with an orientation estimate.

To obtain an orientation estimate, Principle Component Analysis (PCA) is applied to the set of Ray terminal points, yielding two orthogonal vectors e_1 and e_2 in the directions of maximal variance of the local shape. Because e_1 and e_2 do not necessarily correspond to any of the Ray vectors, we pick the two closest Ray vectors $\mathbf{e_1}$ and $\mathbf{e_2}$ to

be the principle axes, as shown in Figure 3.4(d). Finally, the extracted feature vector is re-ordered into the canonical orientation such that $\mathbf{e_1}$ and $\mathbf{e_2}$ correspond to $\mathbf{n_1}$ and $\mathbf{n_2}$, as shown in Figure 3.4(e). Note that the accuracy of the pose estimation depends on the number of Rays in the descriptor.

3.3.2.2 Histogram Features

Recall from Equation 3.2 that the feature vector $\mathbf{f}$ contains intensity histograms $\mathbf{f}^{\text{Hist}}$ extracted for a given supervoxel i and its neighborhood. It complements the Ray features by providing low level intensity and texture cues. We tried several types of local texture and intensity features, including local binary patterns [90] and DAISY [135], but found that a simple histogram computed from a supervoxel i and its set of neighboring supervoxels $\mathcal{N}$ yields the best results. $\mathbf{f}^{\text{Hist}}$ is a concatenation of two b-dimensional histograms. The first one is extracted from the central supervoxel i, and the second from all supervoxels belonging to the neighborhood $\mathcal{N}$ of i. We write

$$\mathbf{f}^{\text{Hist}}(\mathbf{I}, i) = \left[h(\mathbf{I}, i, b),\ \frac{1}{|\mathcal{N}|} \sum_{j \in \mathcal{N}_i} h(\mathbf{I}, j, b) \right]^\top , \tag{3.6}$$

where $h(\mathbf{I}, j, b)$ is a histogram extracted from $\mathbf{I}$ over the voxels contained in supervoxel j. Including the neighbors is necessary, because individual supervoxels are not very discriminative as their intensities are nearly uniform by design.

3.3.3 Graph Cuts with Learned Potentials

The final step of our approach is to segment mitochondria using a graph cuts approach where the unary and pairwise potentials of the energy function incorporate shape cues and learned boundary appearance.

3.3.3.1 Energy Function

Graph partitioning approaches minimize a global objective function defined on an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. In our work, nodes i correspond to supervoxels and edges connect neighboring supervoxels [2, 14, 18]. Our energy function takes the standard

form,

$$E(y|x,\lambda) = \sum_{i\in\mathcal{V}} \underbrace{\psi(y_i|x_i)}_{\text{unary term}} + \lambda \sum_{(i,j)\in\mathcal{E}} \underbrace{\phi(y_i,y_j|x_i,x_j)}_{\text{pairwise term}} \ , \tag{3.7}$$

where $\mathcal{E}$ is the set of edges and $y_i \in \{0,1\}$ is a class label assigned to i corresponding to the foreground and the background. The so-called *unary* term ψ encourages agreement between a node's label y_i and the local image evidence x_i. ϕ is known as the *pairwise* term, which promotes consistency between labels of neighboring nodes i and j. The weight λ controls the relative importance of the two terms.

We segment the image stack by finding a graph cut that minimizes the energy function of Equation 3.7. When the pairwise term is submodular[1], which is the case in our formulation, a global minima of the energy function can be found using the mincut-maxflow algorithm [39]. This results in an optimal labeling

$$\hat{y} = \arg\min_{y} E(y|x,\lambda). \tag{3.8}$$

However, following this standard approach does not mean that resulting segmentations are necessarily perfect, or even good. This is because, as is the case in most other works, the criterion being minimized fails to take shape information into account, even though it is crucial for effective segmentation. Another contributing factor is that the standard pairwise term fails to properly encode the likelihood that edges correspond to mitochondrial membranes, due to the noisy nature of EM data and presence of distracting membranes. In the following subsections, we propose machine learning based solutions to these shortcomings.

3.3.3.2 Learned Shape Cues in the Unary Term

We train a Support Vector Machine (SVM) classifier to predict the unary term in Equation 3.7 using the feature vector $\mathbf{f}$ defined in Section 3.3.2. Because $\mathbf{f}$ includes rotationally invariant shape cues in the form of the Ray descriptor, the SVM injects

[1] The submodularity condition requires *(1)* that the unary term $\psi(y_i|x_i)$ be positive. This is achieved by adding a constant to the energy without affecting the minimum. Submodularity also requires *(2)* that the pairwise term $\phi(y_i,y_j|\cdot)$ satisfies the following condition: $\phi(0,0|\cdot)+\phi(1,1|\cdot) \leq \phi(0,1|\cdot)+\phi(1,0|\cdot)$. Note that the minimum energy of binary submodular functions can be found in polynomial time [67].

important shape information into the unary term, which is taken to be

$$\psi(y_i|x_i) = \frac{1}{1 + P_\psi(y_i|x_i)}, \tag{3.9}$$

where $y_i = 0$ indicates background, $y_i = 1$ indicates foreground, and P_ψ represents the probability that i is within a mitochondria. Because the mitochondria have thick boundaries with specific gray-level statistics, the classifier is trained using manually annotated data with three labels $\{BG, BD, MI\}$, corresponding to background, boundary, and mitochondria instead of only background and mitochondria. Empirically, we found that introducing an explicit boundary class improved the classifiers' ability to recognize mitochondrial membranes from other membranes in the image stack. Thus, the SVM returns probabilities of being within a mitochondria $P(MI|x_i)$, within the boundary $P(BD|x_i)$, or outside $P(BG|x_i)$. Since the boundary label separates background regions from mitochondria regions, we write

$$P_\psi(y_i|x_i) = \begin{cases} P(BG|x_i) & \text{, if } y_i = 0 \text{ ,} \\ P(BD|x_i) + P(MI|x_i) & \text{, otherwise .} \end{cases} \tag{3.10}$$

A three-way one-vs-rest SVM classifier was used to estimate P_ψ, using a Radial Basis Function (RBF) kernel whose parameters were optimized through cross validation to minimize the estimated generalization error.

Only a few previous graph-partitioning methods have attempted to incorporate shape information into the energy function, having done so only for 2D images. They can be categorized as either template or fragment-based. The first category fits shape templates to the image in an alignment or detection step. Templates represent target objects as either contours [32] or silhouettes [1, 79, 100], which are learned or painstakingly constructed beforehand. Typically, a distance transform from the template is used to modulate the potential functions. The complexity of these types of approaches and the difficulty of simultaneously aligning multiple templates have restricted previous works to segment singular well-centered objects.

Fragment-based approaches match image patches extracted around a graph node to a predefined fragment code book in an attempt to encode shape information [3, 72]. However, for highly deformable objects such as mitochondria, an extremely large code book is necessary, making such an approach prohibitively expensive.

3.3.3.3 Learned Boundary Appearance in the Pairwise Term

Most graph-partitioning approaches define the pairwise term as a simple function which favors cutting edges at locations of abrupt color or intensity changes, such as the one proposed in [14]

$$\phi(y_i, y_j | x_i, x_j) = \begin{cases} \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) & \text{, if } y_i \neq y_j \\ 0 & \text{, otherwise,} \end{cases} \tag{3.11}$$

where the observation x_i is simply I_i, the intensity taken from node i, and σ is a constant. However, in EM imagery containing many distracting contours, this may backfire and result in erroneous cuts either along one of the many membranes found in the data or through a mitochondrial cristae.

We address this problem by learning from the data what types of image characteristics indicate a true object boundary and incorporating this information into the pairwise potential. The pairwise term ϕ is defined as

$$\phi(y_i, y_j | x_i, x_j) = \begin{cases} \frac{1}{1+P_\phi(y_i, y_j | x_i, x_j)} & \text{, if } y_i \neq y_j \text{ ,} \\ 0 & \text{, otherwise,} \end{cases} \tag{3.12}$$

where P_ϕ is the SVM output probability that i is within the mitochondria and i's neighbor j is outside. In our application, relevant boundaries are characterized by a very dark membrane separating bright cytoplasm on the exterior, and the dark textured interior of the mitochondria on the interior, as seen in Figure 1.5. We therefore train the second three-way SVM using concatenated feature vectors from neighboring supervoxels i and j

$$\mathbf{f}_{i,j} = [\mathbf{f}_i^\top,\ \mathbf{f}_j^\top]^\top, \tag{3.13}$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ are the feature vectors extracted from the individual supervoxels. The resulting classifier assigns probabilities to one of the three classes $y_{ij} = \{0, 1, 2\}$ where class 0 corresponds to BD-BG pairs, class 1 corresponds to BD-BD pairs, and class 2 corresponds to any other combination $**$-$**$ of ground truth labels

$$P_\phi(y_i, y_j | x_i, x_j) = \begin{cases} P(y_{ij} = 0 | x_i, x_j) & \text{, if } y_i \neq y_j, \\ P(y_{ij} = 1 | x_i, x_j) + & \\ P(y_{ij} = 2 | x_i, x_j) & \text{, otherwise.} \end{cases} \tag{3.14}$$

Very few other works use a more sophisticated pairwise potential than that of Equation 3.11. While some incremental extensions based on Laplacian zero-crossings,

gradient orientations, and local histograms exist [98], very few works go much further. A recent exception can be found in [2], where the authors define an interaction term that encodes geometric relations between multi-region objects. In [112], a set of boundary pixels extracted with an edge detector are pruned using a classifier such that only class-specific edges remain. These edges are attenuated in the pairwise term of the graph cuts segmentation.

3.4 Results

In this section, we first provide details related to the experimental setup and the FIB-SEM data. We then list the parameters we used and describe the learning procedure. We then present our mitochondria segmentation results, investigate some of the trade-offs of our approach, and finally compare our approach to a state-of-the-art method.

3.4.1 Experimental Setup

The data used in our experiments, shown in Figure 1.5, come from two different locations in the brain. The first image stack represents a $5 \times 5 \times 5$ μm section taken from the *CA1 hippocampus*, corresponding to a $1024 \times 1024 \times 1000$ volume which contains $N \approx 10^9$ total voxels. The resolution of each voxel is approximately $5 \times 5 \times 5$ nm. The second section measures approximately $9 \times 5 \times 2.5$ μm, and was taken from the *striatum*, a subcortical brain region. This image stack contains $1536 \times 872 \times 318$ voxels, with a $6 \times 6 \times 7.8$ nm resolution.

Because of the forbiddingly large amount of labor involved in generating an accurate ground truth for such large volumes, we annotated sub-volumes for training and testing purposes. The testing sub-volume for the CA1 hippocampus consists of the first 165 slices of the $1024 \times 1024 \times 1000$ image stack, as indicated by the dotted line in Figure 1.5. A separate image stack from another hippocampus sample containing 200 similarly sized slices was annotated for training our algorithm.

For the striatum, the $1536 \times 872 \times 318$ volume was fully annotated and split into a training and test set, as indicated in Figure 1.5.

Each of these sub-volumes had a size of $768 \times 872 \times 318$. The results provided in Figure 3.6 and Table 3.2 are computed on the test sub-volumes after training the classifiers on the training sub-volumes. The segmentations shown in the top row of

Figure 3.5 are over the entire $1024 \times 1024 \times 1000$ image stack for the hippocampus data including the test volume and unannotated data, while the striatum segmentations are shown only for the test sub-volume.

3.4.2 Parameters and Implementation Details

A summary of parameters used in our experiments is provided in Table 3.1. The sampling interval S for supervoxel centers introduced in Section 3.3.1 was chosen empirically. The resulting supervoxels contain approximately 1000 voxels on average. Supervoxels of this size typically fit within the membranes which helps to ensure that superpixels do not straddle boundaries. As discussed in Section 3.4.4.1, using supervoxels decreases the computational complexity by several orders of magnitude as compared to what would have been required to operate directly on voxels. A strength of the SLIC supervoxel generation scheme is that S value can be adapted if the image resolution were to be changed. The compactness factor m was chosen empirically and provides a good compromise between compactness and boundary adherence. The typical neighborhood size of a supervoxel is $|\mathcal{N}| \approx 8$ for the m and S values given in Table 3.1.

The ray descriptors $\mathbf{f}^{\text{Ray}}$ of Equation 3.4 are $3L = 126$ dimensional vectors, consisting of 3 Ray feature types and L orientations. We have found $L = 42$ to be a good trade-off between computational complexity and angular resolution for the rotational alignment discussed at the end of Section 3.3.2. Rays terminate when they encounter edges found in a 3D Canny edge map [50], whose parameters σ_G, σ_C, t_l, and t_u must be tuned to the data. Because the Canny edge detector can easily miss edges or add spurious ones, we increase robustness by shooting rays from 5% of the voxels within each supervoxel—50 in our case—for each direction and average the results. It is those averages that we use for classification.

All parameters of our algorithm were fixed for both data sets, except for parameters related to the 3D canny edge detector which was adjusted due to differences in contrast between the two data sets.

3.4.3 Experiments and Evaluation

We evaluate our segmentation in terms of the so-called *Jaccard index*, or *VOC score* [26] to measure segmentation quality when ground-truth data is available. It is computed

Table 3.1: Parameters and Settings

Parameter	Value(s)	Notes
S	10	Normalized spatial distance. Controls the number of voxels per supervoxel.
m	40	Normalized intensity distance. Controls supervoxel compactness.
n	5	Number of iterations required for supervoxel clustering to converge.
L	42	Number of Ray directions. Corresponds to vertices on a geodesic sphere.
ρ	≈ 50	Number of Ray features computed per supervoxel.
σ_G	9	Variance of Gaussian derivative filter used to compute gradient in f_{ori} and f_{norm}.
σ_C	(8,10)	Variance used in 3D Canny edge detection for (CA-1 Hippocampus, Striatum).
t_l	(8,14)	Lower threshold used in 3D Canny edge detection for (CA-1 Hippocampus, Striatum).
t_u	(16,27)	Upper threshold used in 3D Canny edge detection for (CA-1 Hippocampus, Striatum).
b	10	Number of histogram bins. $\mathbf{f}^{\text{Hist}}$ concatenates two b-bin histograms from i and i's neighborhood.

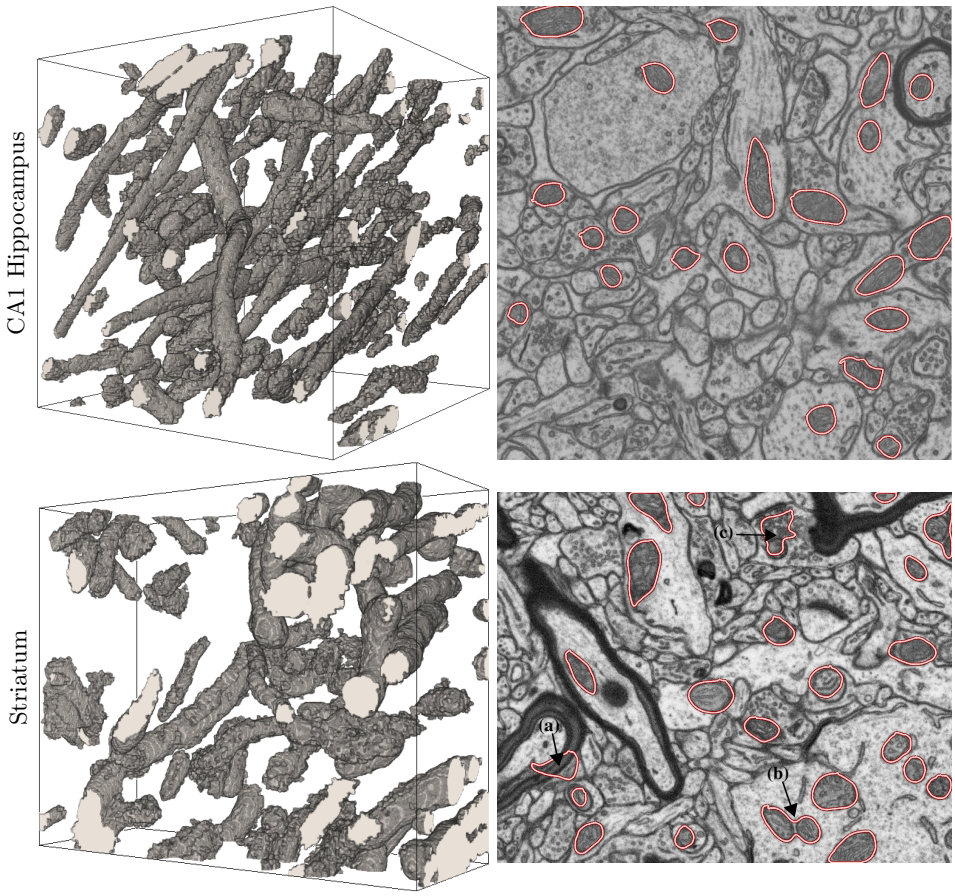


Figure 3.5: *Segmentation of mitochondria from FIB-SEM image stacks and 3D reconstructions.* We applied our approach to two FIB-SEM test stacks acquired from different brain regions. The left column shows the 3D reconstructions of extracted mitochondria. Renderings were produced using V3D [109]. The right column shows segmentation results on individual image slices taken from the image stack. Automatically segmented mitochondria are marked by red contours. Most mitochondria are correctly segmented, but some mistakes remain. Failure modes are indicated by black arrows. *(a)* Dendritic or axonal membranes in close proximity to a mitochondrion can confuse our algorithm, causing it to include part of the nearby membrane with the mitochondrion. *(b)* Occasionally, neighboring mitochondria are erroneously merged by the smoothness constraint in graph cuts when the space between the membranes is very small. *(c)* A cluster of vesicles is mistaken for a mitochondrion. The texture of vesicles can appear deceptively similar to that of mitochondria.

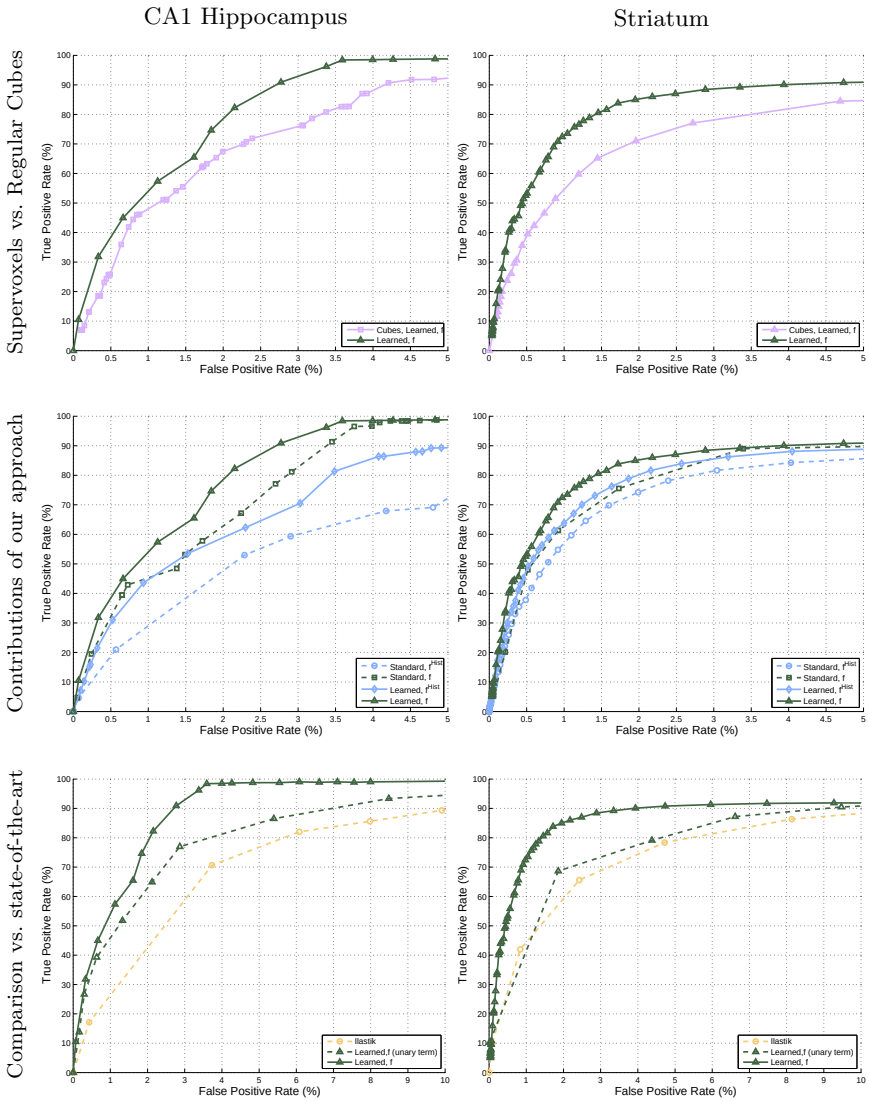


Figure 3.6: *Segmentation results. First row: Supervoxels vs. regular cubes.* We compare segmentation results obtained using SLIC supervoxels of size $S = 10$ to simple $10 \times 10 \times 10$ cubes. Supervoxels, which respect boundaries in the image stack, significantly outperform the cubes while similarly reducing computational complexity. *Second row: Contributions of our approach.* The dashed blue line labeled "Standard, $\mathbf{f}^{\text{Hist}}$" represents a baseline result obtained by using a unary term that only depends on the histogram features of Equation 3.6 and a contrast-based pairwise term given in Equation 3.11. Replacing this pairwise term by the learned one of Equation 3.12 results in the improved solid blue curve labeled "Learned, $\mathbf{f}^{\text{Hist}}$." An even larger improvement is obtained by introducing the Ray features of Equation 3.2, producing the green dashed curve labeled "Standard, $\mathbf{f}$." Finally, combining the learned pairwise term and the Ray features yield the high quality result denoted by the solid green curve labeled "Learned, $\mathbf{f}$". *Last row: Comparing our approach to Ilastik* [125]. We trained the publicly available Ilastik software on the same data we used to train our SVMs and evaluated the segmentations. The solid green curve was generated using our approach. Results obtained using Ilastik appear in as yellow dotted lines.

as

$$\text{VOC} = \frac{\text{True Pos}}{\text{True Pos} + \text{False Pos} + \text{False Neg}} , \tag{3.15}$$

which is the ratio of the areas of the intersection between what has been segmented and the ground truth, and of their union. As an alternative to the Jaccard index, we also considered using the Rand index [52] which attempts to penalize topological segmentation errors. However, since the Rand index does not account for all types of topological errors and the Jaccard index is the *de facto* standard in the Computer Vision community, we report our results using the latter.

Table 3.2 summarizes the segmentation results of our approach and several baseline methods for the hippocampus and striatum test sets. Adding the Rays to the feature vector $\mathbf{f} = [\mathbf{f}_i^{\text{Ray}^\top} \; \mathbf{f}_i^{\text{Hist}^\top}]^\top$ (*Standard* $\mathbf{f}$) is compared to histogram features alone $\mathbf{f} = \mathbf{f}^{\text{Hist}}$ (*Standard* $\mathbf{f}^{\text{Hist}}$). We also report results for learning the pairwise term of Equation 3.12 with the full feature vector (*Learned* $\mathbf{f}$). Finally, Table 3.2 also contains the results obtained using Ilastik [125], and results obtained by replacing the supervoxels with regularly space cubes (*Learned Cube* $\mathbf{f}$). The discussion in the next section provides further details for each method.

The VOC scores reported in Table 3.2 were computed by fixing the value of λ to a value determined through a cross-validation process on the training data. Typically, λ ranged from 0.07 to 0.13.

In the left of Figure 3.5, 3D reconstructions of mitochondria extracted from the test volumes using our approach are provided. In the right column of the same figure, segmentation results on individual image slices are shown where segmented mitochondria

Table 3.2: Segmentation Results measured by the VOC Score [26]

	Method				
	Ilastik	Standard $\mathbf{f}^{Hist}$	Learned Cube $\mathbf{f}$	Standard $\mathbf{f}$	Learned $\mathbf{f}$
Hippocampus	61%	63%	68%	81%	84%
Striatum	58%	60%	60%	70%	74%

are marked by red contours. The total training and processing time was 23 hours for the hippocampus data set and 7 hours for the striatum data set on a 8-core Intel Xeon CPU 2.4 GHz machine with 48 GB RAM.

3.4.4 Discussion

We now investigate several aspects of our approach in further detail. We will show the computational advantages of SLIC supervoxels, the benefits of using Ray descriptors, and the performance gained from learning the pairwise term. We also compare our approach against the state of the art, and discuss failure modes of our approach.

These discussions refer to results appearing in the ROC-like curves appearing in Figure 3.6. The ROC-like curves provided in Figure 3.6 explore points within the operating regimes of the various method we discuss. To generate the curves in the first two columns and the plain curves in the last row, we vary the value of λ, thus changing the influence of the unary and pairwise terms in the energy function of our approach. This results in variations in the true positive rate (TPR) and false positive rate (FPR) of the segmentation, albeit in a non-linear fashion. These curves were generated by jointly labeling supervoxels using information from their neighbors through graph cuts, thus, strictly speaking, they are not ROCs. However, they still provide valuable insight into how consistently our algorithm performs over a range of false positive rates. The dotted lines in the last row of Figure 3.6 are true ROC curves and were obtained by varying a classification or decision threshold for independent elements (supervoxels in our case). Because Ilastik includes neither smoothing nor regularization, we plot results obtained by thresholding the unary term of Equation 3.9 in our approach for a more fair comparison. The dotted curves essentially compare Ilastik's local texture features to our shape and texture features. Note that thresholding the unary term does not perform as well as our full approach but still better than Ilastik, indicating that the features we use are better adapted to the task at hand.

3.4.4.1 Computational Advantage of SLIC Supervoxels

The major bottleneck in our approach is in applying graph-cuts, which has a worst case complexity of $O(|\mathcal{E}|\ |\mathcal{V}|^2)$, where $|\mathcal{E}|$ is the number of edges and $|\mathcal{V}|$ is the number of vertices [15]. Using supervoxels instead of voxels reduces $|\mathcal{V}|$ by several orders of magnitude (a factor of 1000 given the parameters described in 3.4.2), and therefore significantly

speeds up the processing. It is also important to note that memory limitations make it impossible to process a graph of the size required by EM data sets such as ours on a conventional computer. The graph-cuts implementation of [15] requires $40\mathcal{V}+32\mathcal{E}$ bytes to store the graph on a 64-bit machine, which translates to a 227GB memory footprint (for 6-connectivity) or a 852GB memory footprint (for 26-connectivity) for the graph required by the *CA1 hippocampus* volume. Using supervoxels with our parameters reduces the memory consumption to a more manageable size of 296MB.

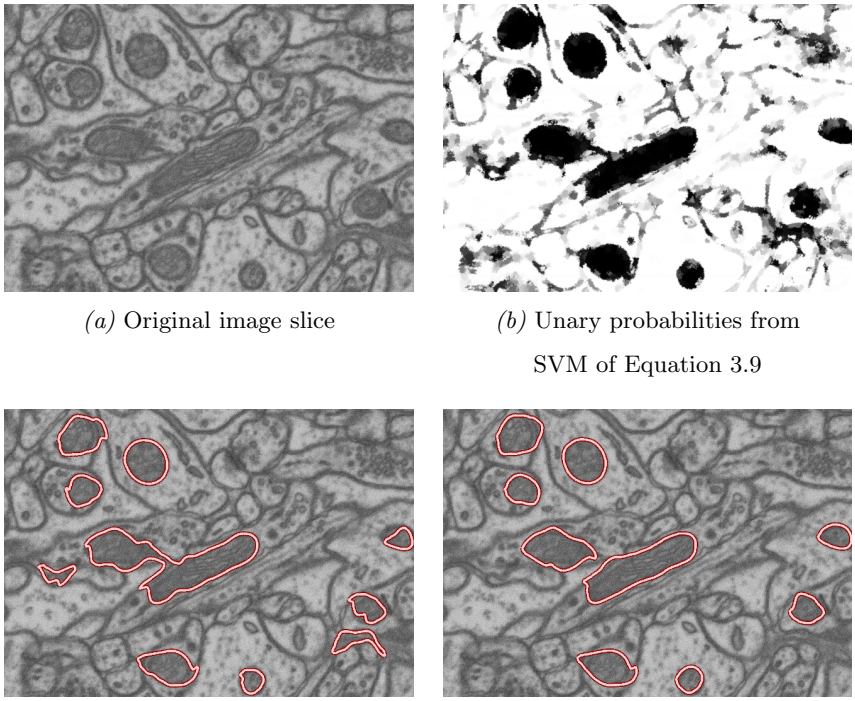
As an alternative to supervoxels, one might consider downsampling the data to reduce processing time and memory consumption. However, doing so reduces the quality of the segmentation. This is because supervoxels adhere to local image boundaries, whereas downsampling does not. To demonstrate this effect, we compare segmentations obtained using our method with SLIC supervoxels to segmentations obtained by replacing the supervoxels with regularly spaced $10 \times 10 \times 10$ cubes, which have roughly the same size but ignore boundaries. The results appear in Figure 3.6(a) and Figure 3.6(b). Results using our method with SLIC supervoxels are denoted Learned,f while the down-sampled results are labeled Cubes, Learned f.

It is clear that downsampling produces significantly worse segmentations than using similarly sized SLIC supervoxels. Consequently, downsampling reduces the VOC score by 14 to 16%, as shown in Table 3.2.

3.4.4.2 Benefits of Ray Descriptors

The Ray descriptor $\mathbf{f}^{\text{Ray}}$ in the feature vector of Equation 3.2 captures important information about the shape of mitochondria. Without it, the feature vector contains only local information provided by the intensity histograms $\mathbf{f}^{\text{Hist}}$. To demonstrate the importance of including shape information, we compare our method using the full feature vector $\mathbf{f} = [\mathbf{f}_i^{\text{Ray}\top} \ \mathbf{f}_i^{\text{Hist}\top}]^\top$ to our method using only histogram features $\mathbf{f} = \mathbf{f}^{\text{Hist}}$.

The results appear in Figure 3.6(c) and Figure 3.6(d). Blue lines denote the results obtained using $\mathbf{f} = \mathbf{f}^{\text{Hist}}$, while green lines incorporate the Ray features $\mathbf{f} = [\mathbf{f}^{\text{Ray}\top} \ \mathbf{f}^{\text{Hist}\top}]^\top$. Dashed lines and solid lines correspond to a standard or learned pairwise term, which are discussed in the next section. Rays significantly improve the segmentation performance. Without them, the VOC score drops by 18% (see Table 3.2).



(a) Original image slice

(b) Unary probabilities from SVM of Equation 3.9

(c) Thresholded unary probabilities

(d) Full approach ("Learned, **f**")

Figure 3.7: *Thresholding unary SVM predictions vs. our learned pairwise approach. (a)* Original image slice. *(b)* Unary mitochondria probability from SVM of Equation 3.9 (dark pixels indicate probable mitochondria). *(c)* Segmentation results obtained by directly thresholding *(b)*. *(d)* Results obtained with our full approach using graph cuts with a learned the pairwise term ("Learned, **f**"). The TPR was set to 85% in (c) and (d).

Looking at Figure 3.7, we can see the discriminative power of the combined feature vector. In Figure 3.7(b) the mitochondria probabilities output by the SVM of Equation 3.9 are shown. Directly thresholding these probabilities already results in reasonably good segmentations (Figure 3.7(c)).

3.4.4.3 Learning the Pairwise Term

Further improvement to segmentation performance is gained by learning the pairwise term of Equation 3.12. Results obtained using the standard pairwise potential of [14], which uses a gradient based approach of the form given in Equation 3.11, is shown in Figure 3.6(c) and Figure 3.6(d) as dashed lines. Replacing this pairwise potential with

one that learns which types of image characteristics indicate a true object boundary (Equation 3.12) results in a significant increase in performance, as indicated by the solid lines.

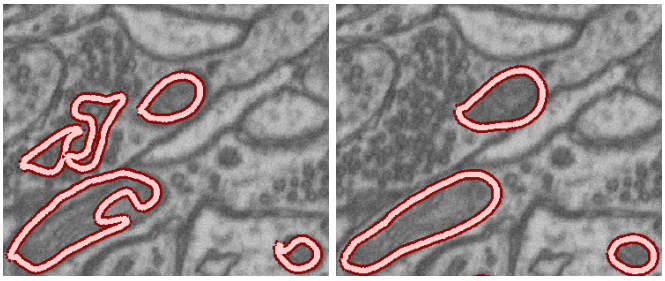
This corresponds to an increase in the VOC score by approximately 4%. In Figure 3.7(d), segmentation results using the learned pairwise term with graph cuts significantly improves the segmentation produced by the unary term in Figure 3.7(c). For the purpose of this experiment, we set $\sigma = \frac{1}{2E[\hat{I}_i - \hat{I}_j]^2}$ in Equation 3.11, where $\hat{I}_i$ is the average intensity within supervoxel i and $E[.]$ denotes the expectation over supervoxels.

3.4.4.4 Comparing against a state-of-the-art method

The Interactive Learning and Segmentation Tool Kit (Ilastik) is a software package for image classification and segmentation [125]. It allows for interactive labeling of an arbitrary number of classes in data sets of various dimensionality. Similar to the work of [101] which also segments mitochondria, Ilastik uses texture cues as well as color and edge orientation in a machine learning framework to perform segmentation. Ilastik's Random Forest classifier can provide real-time feedback of the current classifier predictions, allowing it to perform interactive or fully automatic classification and segmentation.

We provided Ilastik with the same training data used to train our approach, and compare its output to ours in Figure 3.6(e) and Figure 3.6(f). In addition to comparing Ilastik to our full approach, we also plot results obtained by simply thresholding probabilities of Equation 3.9 that define the unary term in the energy function. We do this to provide a more fair comparison of our features against those of Ilastik, which does not include a smoothing or regularization step.

While Ilastik achieves a reasonable segmentation, our approach consistently outperforms it, even when using only the unary term. As shown in Table 3.2, our full approach outperforms Ilastik by a margin of 23% on the hippocampus data and 16% on the striatum, as measured by the VOC score. Example segmentations comparing our method to Ilastik are provided in Figure 3.8. Ilastik mistakenly labels vesicles as mitochondria and has trouble with other various membranes and synapses. Without the global shape information provided by the Ray features such mistakes are difficult to avoid.



(a) Ilastik *(b)* Our approach

Figure 3.8: *Visual comparison of our results vs. Ilastik.* (a) The voxels of a particular slice that are labeled as being within mitochondria by Ilastik are marked by a red contour. These include a number of voxels that belong to vesicles instead of mitochondria. (b) These mistakes disappear when using our approach.

3.4.4.5 Failure modes

Qualitatively our segmentation results are very promising. Note that the 84% VOC score achieved by our algorithm is outstanding in terms of results reported in the VOC challenge [26]. However, this number should be taken with a grain of salt, as the VOC Challenge contains 21 categories of objects, while we only deal with 2 – the mitochondria and the background. Despite the promising results of our approach, there is still room for improvement. Examples of three failure modes are indicated by arrows in Figure 3.5. Dendritic or axonal membranes in close proximity to mitochondria can confuse our algorithm, causing it to include part of the nearby membrane with the mitochondria. Occasionally, neighboring mitochondria are erroneously merged as a result of smoothness enforced by graph cuts when the space between the membranes is very small. Finally, clusters of vesicles are mistaken for mitochondria because texture of vesicles can appear deceptively similar to that of mitochondria.

The shallow depth of the training data in the z-direction could account for some of these failure modes, as very few mitochondria were fully contained withing the training volumes. Increasing the amount of training data or enhancing the learning procedure using a bootstrapping approach could potentially reduce these errors. Furthermore, it would be relatively simple to exploit the fact that graph-cut minimization allows for efficient user interaction [14]. This means that, given an adequate interface, remaining errors could be quickly corrected by the user.

3.5 Conclusion

While the EM image stacks used in this work contain over a billion voxels, they span volumes smaller than $10 \times 10 \times 10$ μm, which represents less than a billionth of the volume of the entire mouse brain. If it is ever to be mapped in its entirety, efficient automatic segmentation methods, such as the one we propose in the work, will be required.

Our fully automatic approach to segment mitochondria from FIB-SEM image stacks overcomes the limitations of standard graph-partitioning approaches by: operating on supervoxels instead of voxels for computational efficiency, by using 3D Ray descriptors to model shape in the unary term, and by using a learning approach to model the appearance of the boundary in the pairwise term. We have demonstrated the computational efficiency of using supervoxels, and experimentally shown the increases in segmentation quality attributed with using Ray descriptors and learning to model boundaries in the pairwise term. Our experiments have also demonstrated that our approach outperforms a state-of-the-art 3D segmentation method, and that our segmentation closely matches the performance of human annotators.

The method presented in this chapter relied on building both a strong unary and pairwise term and then combining them using cross-validation. While effective, the cross-validation method is sub-optimal and the two following chapters will show that the max-margin approach presented in Section 2.3.4 yield better empirical results. While the focus of this chapter is on the segmentation of mitochondria in FIB-SEM image stacks, the proposed technique should be applicable to other cellular structures in EM as well as in other forms of microscopy.

CHAPTER

FOUR

STRUCTURED IMAGE SEGMENTATION USING KERNELIZED FEATURES

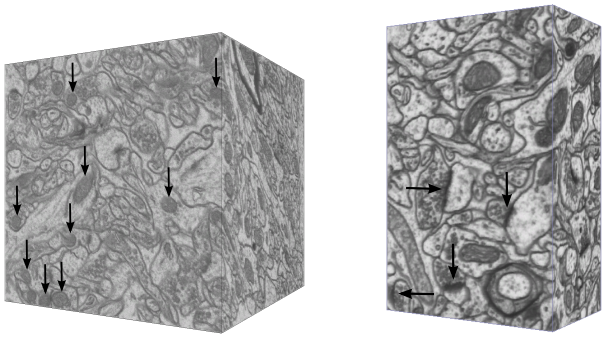
4.1 Introduction

In the previous chapter we have seen that CRFs are a class of powerful graphical models that greatly improve the performance for the segmentation of mitochondria in EM images. In the CRF framework, the solution is typically obtained by minimizing an energy function that is the sum of *unary* and *pairwise* terms [1]. The unary term encodes the likelihood that a particular label should be assigned to a pixel based on local image features[2]. The pairwise term encodes the tendency of neighboring pixels or superpixels to share the same label, thus enforcing spatial regularity.

In recent years, machine learning techniques have increasingly been used to derive these terms in favor of simpler traditional models. However, this is usually done separately for each term: the unary term is optimized for labeling individual pixels while the pairwise term is optimized for labeling pixel pairs. As a result, the two terms can often be incommensurate, and an *ad hoc* weighting step is required to balance their relative influences. For example, we have seen in Chapter 3 that cross-validation could be used to find this balance. However, this requires a separate validation set and still does not guarantee that the two terms are jointly optimized.

[1] Higher order terms are also possible in theory, though less commonly used in practice due to higher computational cost. The unary term is also referred to as the data term.

[2] Alternatively, groups of pixels or superpixels can be used in order to speed up computation.



(a) mitochondria *(b)* synapses

Figure 4.1: Two nearly isotropic stacks of neural tissue acquired using EM microscopy annotated for training and testing. *(a)* This stack contains 1000 images of 1024×1024 pixels, and was used for the task of segmenting mitochondria, indicated by arrows. *(b)* This stack contains 250 images of 655×429 pixels and was used to segment synaptic gaps, indicated by arrows.

Instead, the parameters of the unary and pairwise terms should be learned jointly to infer the optimal labeling. This can be done using the recent structured-SVM (SSVM) framework [136]. It involves learning the unary and pairwise terms jointly through an iterative cutting-plane scheme that provably minimizes an upper bound on the empirical loss and the model complexity. Currently, SSVMs can be of practical use only in conjunction with terms that are linear functions of the input features. This is because introducing non-linear terms would result in quadratic times for learning iterations, which quickly becomes unmanageable for regular-size 2D images and even more so for 3D data. Thus, it is usually not practical to use non-linear kernels in an SSVM framework, though they are often more powerful than their linear counterparts.

In this chapter, we present a method that overcomes this limitation through a two-step learning approach. We first use a regular non-linear SVM to create kernel-transformed feature vectors, each of which consists of kernel products between the input feature vector and a set of basis vectors that may not be orthogonal to each other. We then train a linear SSVM on the transformed features. This approach combines the power of non-linear kernels for individual pixel classification in the unary term with the regularizing effect of the pairwise term, while enforcing consistency between the two. This yields both improved segmentation performance and computational efficiency.

This line of research was primarily motivated by the need to more accurately segment synapses and mitochondria in electron microscopy (EM) stacks, such as those of Figure 4.1. In this kind of data, unlike in popular segmentation benchmarks such as MSRC [123], global features that predict whether or not a type of object appears anywhere in the image are not useful. This is because it is known that synapses and mitochondria always appear in EM stacks, whereas it is not known if a cow or bird will appear in a particular image from a standard benchmark dataset such as MSRC. Thus, for EM applications, the CRF must rely solely on unary evidence and spatial smoothness. We demonstrate on all three datasets that our approach indeed boosts the performance of the learned CRF. Furthermore, it outperforms the previous state-of-the-art in our target applications: synapse and mitochondria segmentation.

4.2 Related Work

For tasks such as segmentation, consistent labeling of highly-correlated neighboring pixels is of great importance. Structured prediction has emerged as a powerful tool to take into account such correlations. In this section, we first discuss current approaches to structured prediction and the computational complexity issues that restrict them from use in conjunction with non-linear kernels. We then consider kernel approximation techniques that can be used to address these issues. Finally, we discuss structural kernels which, like our approach, rely on kernel functions in a structured prediction framework. But unlike our approach, they do so over the entire output space, which carries certain disadvantages.

Structured Prediction Structured prediction methods such as conditional random fields (CRFs) [77] have been widely applied to problems with structured outputs. While traditional classifiers, such as decision trees and SVMs independently map each data instance to a single label, structured prediction methods take into account the statistical correlations between labels. This is critical for tasks such as image segmentation, where such correlations are strong between nearby pixels. Learning CRF models using large-margin methods has rapidly gained popularity in recent years. This is because they are more objective-driven and do not involve the daunting partition function that can render maximum-likelihood approaches intractable in CRFs with loopy graph structures.

Compared with earlier approaches including the max-margin Markov network [133], the structured support vector machine (SSVM) [136] is especially appealing, and has since been successfully applied to many computer vision tasks, such as in [87, 104, 131], among others. The SSVM's appeal is due, in part, to its ability to take into account a variety of *loss functions*.

Computational Complexity SSVMs, however, require the CRF energy function to be linear, which in turn places the same restriction on all the unary and spatial terms due to the additive nature of CRF energies. While, in principle, the linear function can be defined in some high dimensional or possibly infinite-dimensional space, reproducing this kernel Hilbert space through the use of non-linear kernels is often infeasible in practice given that the number of kernel evaluations grows quadratically with the model size and must be optimized in the dual space. Though SSVM learning techniques based on sampled cuts [6, 154] have alleviated this problem to some extent, they do have to sacrifice some performance for speed and, even with this trade-off, are still generally much slower than linear SSVMs [154]. Moreover, earlier implementations of these techniques were intended for use in conjunction with the cutting-plane method to speed up training of regular non-linear SVMs. This results in a *multivariate* output space in the SSVM formulation, analogous to a CRF without edges, which is considerably less useful for image segmentation.

Kernel Approximation Kernel Approximation is another way to improve SSVM training efficiency by seeking a lower, finite-dimensional representation of the kernel-induced feature map that lies in a higher or infinite dimensional space. This can be achieved by random sampling from the typically infinite-dimensional feature map [8, 113] whose analytical form can be obtained using Fourier analysis when the kernel is homogeneous or stationary [141]. While promising results have been shown for specific additive kernels [91, 141], it is less clear how this approach generalizes to non-additive kernels, such as the Gaussian RBF that is more difficult to approximate. Alternatively, the recently proposed *locally* linear SVM [76] can be used to simulate a non-linear decision boundary. However, this does not yield a *globally* linear function and, thus, does not fit into the SSVM framework. Moreover kernel approximation typically introduces

additional tuning parameters such as the number of samples, which often present a performance-speed trade-off for which well-defined tuning criteria are lacking.

Structural Kernels Finally, it is worth pointing out the difference between our approach and the recently proposed *structural kernels*, which also perform structured prediction using non-linear kernels, but in a very different setting. In [6, 11], the kernels are defined on the overall output space, that is, the entire CRF, to exploit image-level "structural" information such as shape and color. While this serves to bias local labels and is useful for segmenting large dominant objects from the background, it often requires training data that completely characterizes the possible object configurations, such as binary masks. Multiple objects, or objects whose pose are not represented in the trained model will cause the approach to fail. Our approach, on the other hand, uses regular kernels defined as products between a set of basis vectors and the feature vectors extracted from individual nodes (i.e. pixels or superpixels). This has the effect of making it more "local", and robust to such failures.

4.3 Learning a CRF with Kernelized Features

We begin by describing our CRF model for segmentation in Section 4.3.1. We then discuss how to learn its parameters using an SSVM framework in Section 4.3.2, with specific details on how to express the CRF model in the required linear form in Section 4.3.3. In Section 4.4, we introduce a technique to create "kernelized" features, enabling us to leverage the power of non-linear kernels while the SSVM remains linear. Figure 4.2 outlines our approach.

4.3.1 CRF for Segmentation

As a standard preprocessing step, we perform a preliminary over-segmentation of our input image into superpixels [1] using SLIC [5]. The CRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is thus defined so that each node $i \in \mathcal{V}$ corresponds to a superpixel and there is an edge $(i, j) \in \mathcal{E}$ between two nodes i and j if the corresponding superpixels are adjacent in the image. Let $Y = \{y_i\}$ for $i \in \mathcal{V}$ denote the labeling of the CRF which assigns a class label y_i to

[1] Or supervoxels in the case of volumetric data.

each node i. Its energy function can then be written as

$$E_{\mathbf{w}}(Y) = \sum_{i \in \mathcal{V}} D_i(y_i) + \sum_{(i,j) \in \mathcal{E}} V_{ij}(y_i, y_j), \tag{4.1}$$

where D_i is the unary term and V_{ij} is the pairwise term. Both D_i and V_{ij} depend on the observed data and the CRF parameters $\mathbf{w}$, in addition to the labeling Y. The energy is also commonly referred to as the "cost" in the literature and we will use the two terms interchangeably. The inferred optimal labeling is simply the one that minimizes it, that is, $Y^* = \arg\min_{Y \in \mathcal{Y}} E_{\mathbf{w}}(Y)$, where $\mathcal{Y}$ denotes the set of all possible labelings. While the exact minimization of the energy function is generally intractable on loopy CRFs, good approximate solutions can be found efficiently using techniques such as graph cuts [16] and belief propagation [99]. In our case, we use graph cuts when the energy function is submodular [67] and belief propagation otherwise.

4.3.2 Learning the CRF Using SSVM

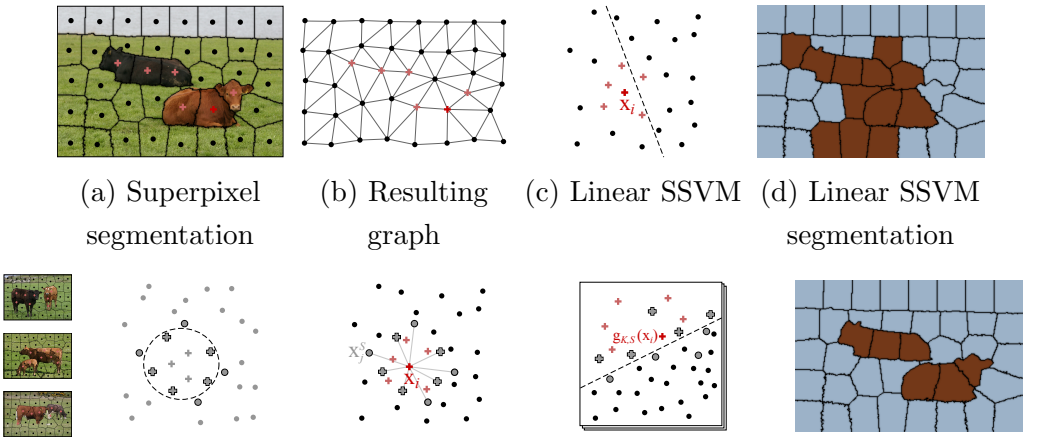
Structured SVM (SSVM) is a large-margin method for learning the parameters of models with structured outputs, such as the CRF model we use for segmentation. The SSVM uses the ground truth training data to learn the CRF parameters so that the inferred labeling of the CRF is "close" to that of the training data, defined as yielding a low *loss*. More specifically, given a set of N training examples with ground truth labelings $(Y^{(1)}, \ldots, Y^{(N)})$, the SSVM [1] optimizes a quadratic objective function of the parameters $\mathbf{w}$ subject to a set of linear soft margin constraints

$$\begin{aligned} &\min_{\mathbf{w}, \xi \geq \mathbf{0}} \frac{1}{2} ||\mathbf{w}||_2^2 + \frac{C}{N} \sum_{n=1}^{N} \xi_n \\ &\text{s.t. } \forall n, Y \in \mathcal{Y}_n \backslash Y^{(n)} : \delta E_{\mathbf{w}}(Y) \geq \Delta(Y^{(n)}, Y) - \xi_n \end{aligned} \tag{4.2}$$

where $\mathcal{Y}_n$ is the set of all possible labelings for example n, ξ_n are the slack variables, and $\delta E_{\mathbf{w}}(Y)$ is shorthand for $E_{\mathbf{w}}(Y) - E_{\mathbf{w}}(Y^{(n)})$. The constant C controls the trade-off between margin and training error, and the loss function Δ measures the closeness of a labeling Y to the ground truth $Y^{(n)}$.

A natural choice for Δ is the per-superpixel 0-1 loss $\Delta(Y^{(n)}, Y) = \sum_{i \in \mathcal{V}} \Delta(y_i^{(n)}, y_i)$ with $\Delta(y_i^{(n)}, y_i) = I(y_i \neq y_i^{(n)})$, which penalizes all errors equally. However, in image

[1] Here, we use the margin-rescaling variant of SSVM [136]



(a) Superpixel segmentation (b) Resulting graph (c) Linear SSVM (d) Linear SSVM segmentation

(e) Standard RBF SVM trained with individual superpixels (f) Kernel transform (g) "Kernelized" SSVM (h) "Kernelized" SSVM segmentation

Figure 4.2: Our approach. *(a)* A superpixel over-segmentation of an image where each superpixel center is marked (+/• denotes foreground/background). *(b)* The superpixel graph used to construct the CRF, where each node corresponds to a superpixel, and edges indicate adjacency in the image. *(c)* An illustration of the feature space. Each point represents a feature vector extracted from a superpixel. Because it is not linearly separable, the standard SSVM gives a poor segmentation result in *(d)*. *(e)* To address this, we train a non-structured kernel SVM on individual superpixels to obtain a set of support vectors S, indicated by outlined points. *(f)* Kernel-transformed features $\mathbf{g}_{K,S}(\mathbf{x}_i)$ are obtained for each feature vector x_i from the kernel products of x_i and S. *(g)* Data in the $|S|$-dimensional "kernelized" feature space is linearly separable, and can be used to train a linear SSVM. *(h)* The improved segmentation result.

segmentation, it is common for certain classes to occur much more frequently than others. To ensure good performance across all classes, we adopt a loss function that weighs errors for a given class inversely proportional to the frequency with which it appears in the training data

$$\Delta(y_i^{(n)}, y_i) = \begin{cases} \frac{1}{\text{frequency}(y_i^{(n)})} & \text{, if } y_i \neq y_i^{(n)} \\ 0 & \text{, otherwise.} \end{cases} \tag{4.3}$$

Since the total number of constraints grows exponentially with the CRF size, they cannot be exhaustively enumerated in most cases. The SSVM solves this by employing an iterative cutting-plane algorithm, which finds the most violated constraint for each example n

$$\hat{Y} = \underset{Y \in \mathcal{Y}_n}{\arg\min}\, E_{\mathbf{w}}(Y) - \Delta(Y^{(n)}, Y) \tag{4.4}$$

at every iteration and adds it to the working set of constraints. As with inferring the optimal labeling, finding the most violated constraint is intractable on loopy CRFs. However, the approximate most violated constraints can be found efficiently using the same kind of energy minimization techniques as in inference, and this approach has proven effective in practice [31, 131].

4.3.3 Linearizing the CRF Energy Function

Since the SSVM operates by solving a quadratic program (QP), all the constraints in Equation 4.2 must be linear [136]. This requires that the energy function $E_{\mathbf{w}}$ be expressible as an inner product between the parameter vector and a feature map. Since the energy is the sum of individual unary and pairwise terms, this implies that D_i and V_{ij} also must be expressible as

$$D_i(y_i) = \left\langle \mathbf{w}^D, \psi_i^D(y_i) \right\rangle \tag{4.5}$$

and

$$V_{ij}(y_i, y_j) = \left\langle \mathbf{w}^V, \psi_{ij}^V(y_i, y_j) \right\rangle, \tag{4.6}$$

where $\psi_i^D(y_i)$ and $\psi_{ij}^V(y_i, y_j)$ are feature maps dependent on both the observed data and the labels, and where

$$\mathbf{w} = ((\mathbf{w}^D)^T, (\mathbf{w}^V)^T)^T \tag{4.7}$$

is the vector of parameters that define the functions D_i and V_{ij} respectively.

If we let $\Psi^D(Y) = \sum_{i \in \mathcal{V}} \psi_i^D(y_i)$ and $\Psi^V(Y) = \sum_{(i,j) \in \mathcal{E}} \psi_{ij}^V(y_i, y_j)$, then $\Psi(Y) = (\Psi^D(Y)^T, \Psi^V(Y)^T)^T$. allowing the CRF energy to now be written linearly as

$$E_{\mathbf{w}}(Y) = \langle \mathbf{w}, \Psi(Y) \rangle . \tag{4.8}$$

Let $\mathbf{x}_i$ be a feature vector associated with node i extracted from the observed data. We can define the data feature map as

$$\psi_i^D(y_i) = (I(y_i = 1)\mathbf{x}_i^T, \ldots, I(y_i = K)\mathbf{x}_i^T)^T, \tag{4.9}$$

where K is the number of possible labels, i.e., $y_i \in \{1, \ldots, K\}$. If we write $\mathbf{w}^D = (\mathbf{w}_1^D, \ldots, \mathbf{w}_K^D)^T$, the unary term becomes the inner product

$$D_i(y_i) = \left\langle \mathbf{w}_{y_i}^D, \mathbf{x}_i \right\rangle , \tag{4.10}$$

which represents the energy of node i taking on label y_i. Similarly, if we define the pairwise feature map as

$$\psi_{ij}^V(y_i, y_j) = (I(y_i = a, y_j = b))_{(a,b) \in \{1,\ldots,K\}^2} \tag{4.11}$$

with the corresponding parameters $\mathbf{w}^V = (w_{ab})_{(a,b) \in \{1,\ldots,K\}^2}$, then the pairwise term

$$V_{ij}(y_i, y_j) = w_{y_i y_j} \tag{4.12}$$

reflects the transition cost between nodes i and j from label y_i to label y_j. Although the above definition depends only on the labels y_i and y_j, the pairwise term can, in fact, be made data-aware (as in [87, 123]). For instance, it can be made gradient-adaptive by including parameters for each discretized image gradient level. In a similar fashion, it can be made to consider geometric relationships such as "sky should appear above grass"[1].

4.4 Kernel-transformed Features

As shown in the previous section, standard SSVMs require an energy function that is linear in the parameters and features. Since unary terms based on non-linear SVMs

[1]We incorporate both geometric and gradient context in our model. They are omitted from the notation for brevity, as the extension follows naturally from above.

are often more powerful and produce better results [37, 87], this constitutes a major limitation.

It should be possible, in principle, to learn non-linear unary terms within the SSVM framework by implicitly defining $\mathbf{w}$ and $\mathbf{x}_i$ of Equation 4.10 in a high-dimensional space through kernels. In practice, however, these kernels are very high- or even infinite-dimensional, making such an approach computationally intractable.

Our approach aims at circumventing this problem. It starts from the observation that a non-linear binary (+1/-1 label) SVM classifier always takes the form

$$\text{Score}(\mathbf{x}) = \sum_j \alpha_j y_j^S K(\mathbf{x}_j^S, \mathbf{x}) \;, \tag{4.13}$$

where $\mathbf{x}_j^S \in S$ are the support vectors with corresponding labels y_j^S. Extended to multi-class labels ($y_i \in \{1, \ldots, K\}$) for a general unary term, it becomes

$$D_i(y_i) = -\sum_j \alpha_j c(y_j^S, y_i) K(\mathbf{x}_j^S, \mathbf{x}_i) \;, \tag{4.14}$$

where $c(y_j^S, y_i)$ is 1 if $y_i = y_j^S$ and -1 otherwise. Note that, although the function is non-linear in the input features $\mathbf{x}_i$, because of the non-linear kernel K, it is linear in the kernel products $K(\mathbf{x}_j^S, \mathbf{x}_i)$.

If we define $\mathbf{g}_{K,S}(\mathbf{x}_i)$ as the vector of kernel products

$$\mathbf{g}_{K,S}(\mathbf{x}_i) = (K(\mathbf{x}_1^S, \mathbf{x}_i), \ldots, K(\mathbf{x}_{|S|}^S, \mathbf{x}_i))^T, \tag{4.15}$$

and $\mathbf{w'}_{y_i}$ as their coefficients

$$\mathbf{w'}_{y_i} = (-\alpha_1 c(y_1^S, y_i), \cdots, -\alpha_{|S|} c(y_{|S|}^S, y_i))^T, \tag{4.16}$$

then the unary term can be re-expressed as

$$D_i(y_i) = \left\langle \mathbf{w'}_{y_i}, \mathbf{g}_{K,S}(\mathbf{x}_i) \right\rangle, \tag{4.17}$$

which is of the finite-dimensional linear form needed for learning within the SSVM framework, as discussed in the previous section.

This suggests a simple, 2-step learning approach to incorporate kernels into an SSVM, illustrated in Figure 4.2. First, we train a standard non-structured non-linear kernel SVM using feature vectors extracted from individual superpixels to obtain a

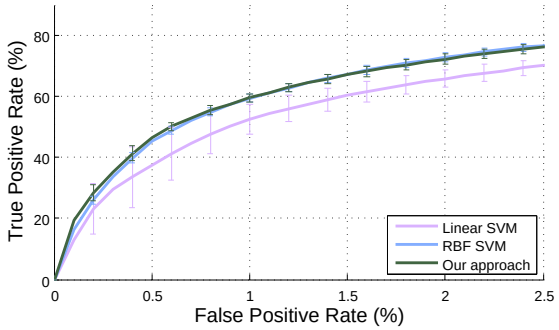


Figure 4.3: Comparison of a linear SVM, RBF-SVM, and a linear SVM trained on feature vectors kernelized using the support vectors of the RBF-SVM. For classification of individual superpixels (ignoring structure), training a linear SVM on kernelized feature vectors yields a similar performance to a standard RBF-SVM. Error bars indicate standard deviation over 10 experiments.

set of support vectors S. S are then used as *basis vectors* to create a set of *kernel-transformed* (or "kernelized") feature vectors $\mathbf{g}_{K,S}(\mathbf{x}_i)$, which are provided to train the linear SSVM.

Although our formulation is not equivalent to a non-linear kernel SSVM[1], nor can it be shown to approximate a non-linear SSVM as kernel approximation methods do [91, 113, 141], it does produce models with the same functional form as those learned using a kernel SSVM and, importantly, performs well in practice. To demonstrate the principle that a linear SVM trained using kernel-transformed features performs similarly to a non-linear SVM that uses the same kernel, we conducted a simple experiment. The goal was to classify individual superpixels, ignoring structure. We compare the performance of a standard linear SVM, an SVM trained with an RBF kernel, and a simplification of our approach in which feature vectors kernelized using the support vectors obtained from the RBF-SVM are used to train a standard linear SVM. The results appearing in Figure 4.3 support our intuition – so long as we transform the original feature vector using the right set of basis vectors, learning the new coefficients under a different objective function (i.e., as primal instead of dual variables) yields performance similar to a non-linear kernel SVM.

[1]The parameters $\mathbf{w}'$ now correspond to the primal variables of a linear SSVM instead of the dual variables of a non-linear SSVM.

4.5 Results

Our primary motivation for developing the technique presented here was to segment synaptic gaps and mitochondria from images acquired with an electron microscope, such as the ones depicted in Figure 4.1. For such data, it is known *a priori* that particular cellular structures will appear in the image. Consequently, the use of global features and/or priors which has proved to be essential when the presence of a particular category is uncertain, as is the case for segmentation benchmarks such as MSRC [123], is of no particular benefit. Because EM segmentation methods can not use such global features, they have to more heavily rely on smoothness terms.

Nevertheless, we demonstrate that our approach boosts performance of the learned CRF model on the MSRC dataset as well as the EM data. In subsection 4.5.2 we are able to match state-of-the-art results on the MSRC dataset by incorporating global features. For the EM data in subsections 4.5.3 and 4.5.4, our approach significantly outperforms the state-of-the-art.

4.5.1 Competing Methods

To highlight the relative importance of the various components of our approach, we compare against the following variations which we treat as baselines.

- **Linear SVM** – A standard linear SVM trained on the original feature vectors x_i. Each superpixel is classified independently (i.e., without CRF).
- **Linear SSVM/CRF** – As described in Section 4.3.2, a linear SSVM on the original features is used to learn the parameters of a CRF, which is used for segmentation.
- **RBF SVM** – A non-linear SVM trained on the original feature vectors using an RBF kernel. Each superpixel is classified independently.
- **RBF+SSVM/CRF** – Instead of being kernelized versions of the original feature vectors as in our approach, transformed feature vectors contain the per-class scores of an RBF SVM trained on the original features. A linear SSVM learns the parameters of a CRF using the transformed features.

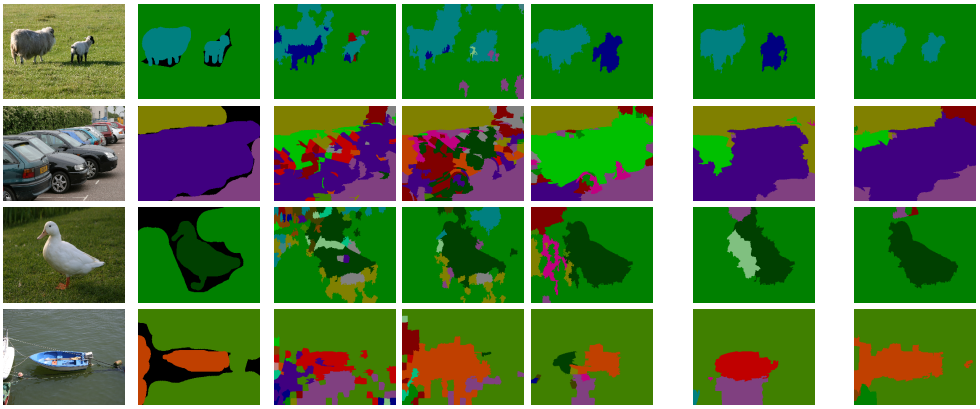


Figure 4.4: Example segmentations from the MSRC dataset.

- **Codebook SVM** – The original feature vectors are transformed using a "kernel codebook" approach inspired by [36]. It relies on k-means clustering to create a set of basis vectors (codewords) instead of discriminatively learned SVM support vectors. We used as many codewords as support vectors for all our experiments. The resulting features are classified independently with a linear SVM.

- **Codebook SSVM/CRF** – Features are constructed in the same manner as for Codebook SVM, but a linear SSVM is used to learn a CRF for structured prediction.

- **Kernelized SSVM/CRF** – Our method. A linear SSVM learns the parameters of a CRF for structured prediction, using kernel-transformed features $\mathbf{g}_{K,S}(\mathbf{x}_i)$, as described in Section 4.4.

In addition to these, we also compare to state-of-the-art methods, [37, 74, 87, 122] for MSRC and [88] for the EM dataset. All SSVM methods used Joachims' *SVM-struct* software [136] for training.

4.5.2 MSRC Dataset

The MSRC-21 dataset is a popular multi-class object segmentation benchmark dataset which contains 591 images with objects from 21 categories.

Linear SVM	**59**	79	78	60	63	70	78	61	46	61	76	57	64	59	58	63	70	52	55	68	31	68	62	building
RBF SVM	47	83	**80**	83	62	76	74	62	72	55	73	59	**86**	40	54	62	73	66	34	60	34	70	64	grass
Linear SSVM/CRF	53	81	74	76	72	73	85	63	59	60	81	54	77	62	68	72	76	66	**68**	**71**	6	71	65	tree
RBF+SSVM/CRF	54	**84**	70	75	69	80	86	**66**	69	**77**	87	46	74	**66**	65	69	65	77	58	68	18	71	68	cow
Kernelized SSVM/CRF	41	77	79	**87**	**91**	**86**	**92**	65	**86**	65	**89**	**61**	76	48	**77**	**91**	**77**	**82**	32	48	**39**	**73**	**70**	sheep
G-Kernelized SSVM/CRF	59	90	**92**	82	83	94	91	80	85	**88**	96	89	73	48	96	62	81	87	33	44	30	82	76	sky
Shotton et al. [122]	49	88	79	**97**	**97**	78	82	54	**87**	74	72	74	36	24	93	51	78	75	35	**66**	18	72	67	airplane
Ladicky et al. [74]	**80**	**96**	86	74	87	**99**	74	**87**	86	87	82	**97**	**95**	30	86	31	**95**	51	**69**	**66**	09	**86**	75	water
Gonfaus et al. [37]	60	78	77	91	68	88	87	76	73	77	**93**	**97**	73	**57**	95	**81**	76	81	46	56	46	77	75	face
Lucchi et al. [87]	50	83	87	81	84	90	**97**	72	75	79	90	95	79	52	**97**	**81**	80	**89**	51	64	**60**	79	**78**	car

Table 4.1: MSRC segmentation results. We follow the standard reporting procedure. For each category, the pixel-wise classification rate is provided. Global pixel-wise accuracy and average per-category scores provide measures for overall performance. The first part of the table shows baseline methods that ignore global information, while the second part compares our approach to state-of-the-art methods that consider global information. Bold entries are used for each part separately to indicate best performance.

Methodology We extract feature vectors from each image by first over-segmenting the image using SLIC superpixels [5]. We then extract SIFT descriptors and color histograms from image patches surrounding each superpixel centroid. We also include location information as in [74]. This information is converted to a bag-of-words descriptor using a nearest-neighbor search[1]. The resulting descriptor serves as the feature vector x_i used to train the various methods. Training and testing is done using the standard split of the dataset [123]. Note that the Codebook SVM and Codebook SSVM methods are not reported since x_i already contains a bag-of-words descriptor. In addition to the baseline methods described above, we compare state-of-the-art approaches [37, 74, 87, 122] to our approach incorporating global features:

- **G-Kernelized SSVM/CRF** – Our method as described in Section 4.4 using an augmented feature vector. In addition to SIFT descriptors and color histograms, the feature vector contains the global features described in [37].

Analysis Table 4.1 summarizes the segmentation performance of the various approaches and example segmentations appear in Figure 4.4. The top of Table 4.1 compares our method to the baselines introduced in Section 4.5.1, demonstrating that structured learning tends to yield higher segmentation performance than unstructured approaches. It is also clear that non-linear models outperform linear ones for both structured and instance-based learning. Because our approach combines structured prediction with the power of non-linear kernels, it yields superior performance over the baselines in nearly every category.

The bottom of Table 4.1 compares our approach to state-of-the-art methods. As observed in [87], global features are necessary to obtain state-of-the-art performance on datasets such as MSRC. Therefore, when we do not use them our performance is lower but by incorporating the features of [37] into our approach (G-Kernelized SSVM/CRF), we are able to match state-of-the-art performance and out-perform all other methods in several categories.

[1] A dictionary containing 1,000 words for SIFT features and 400 words for color histograms is constructed using k-means on extracted features.

Table 4.2: Segmentation results for the synaptic gap EM dataset. We measure segmentation performance using the *Jaccard index*, the ratio of the correctly segmented area to the union of the segmentation and the ground truth.

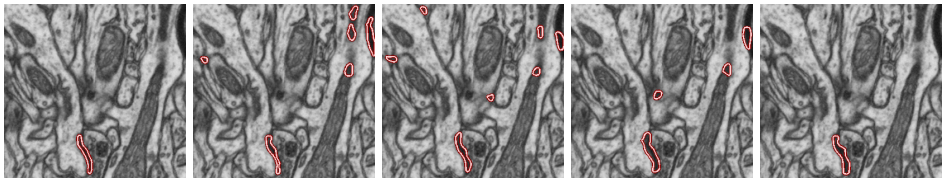
Linear SVM	Linear SSVM/CRF	RBF SVM	RBF+ SSVM/CRF	Codebook SVM	Codebook SSVM/CRF	**Kernelized SSVM/CRF**
58%	60%	61%	64%	60%	63%	**66%**

4.5.3 Synaptic Gap Dataset

Segmenting the synaptic gaps from EM images of neural tissue shown in Figure 4.1*(b)* is challenging due to the large amount of clutter including vesicles, mitochondria, and various cellular membranes that exhibit a variety of distracting shapes and textures. The dataset of Figure 4.1*(b)* contains 250 images of 655×429 pixels and a total of 24 synapses. Each pixel was labeled by an expert as either synaptic or non-synaptic. The dataset was then split into 2 parts for training and testing.

Methodology We begin by over-segmenting each image using SLIC superpixels [5]. Each image contained an average of 4000 superpixels and 11400 edges. At the location of each superpixel we extract a feature vector consisting of intensity histograms and steerable filter responses. The later is computed by convolving a patch extracted at the center of each superpixel with a set of steerable filters at 3 different scales ($\sigma = \{2, 5, 6\}$). An SVM with an RBF kernel trained on 40K randomly sampled superpixels provides the support vectors for the kernel transform in our approach.

Analysis A summary of the segmentation performance is provided in Table 4.2, and examples appear in Figure 4.5. As with the MSRC baselines, we observe a trend in which structured learning yields higher segmentation performance than unstructured approaches and non-linear models outperform linear models. Our approach, which combines structured learning with non-linear models outperforms all other methods.



Ground truth | Linear SVM | RBF SVM | Codebook SSVM/CRF | Kernelized SSVM/CRF

Figure 4.5: Segmentation results on the synaptic gap dataset. The kernelized SSVM correctly segments the synapse in this example, while the baseline methods include spurious segments.

4.5.4 Mitochondria Dataset

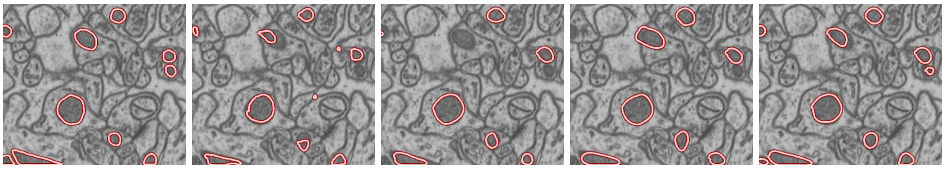
Here, we perform mitochondria segmentation in 3D using the large image stack presented in Chapter 3 and shown in Figure 4.1*(a)*. This greatly increases the scale of the problem since the image stacks are orders of magnitude larger than most 2D images.

Methodology We begin by over-segmenting the volume using SLIC supervoxels [5]. For each supervoxel, we extract a feature vector that captures local shape and texture information using Ray descriptors [88] and intensity histograms. Those feature vectors x_i are used to train each baseline methods, as well as our model. Due to the high cost of labeling such large volumes, our experiments are restricted to two subvolumes containing $1024 \times 768 \times 165$ voxels. The first subvolume, containing 42 mitochondria, was used to train the various methods; the second, containing 45 mitochondria, was used for testing. Each subvolume contains $\sim$13K supervoxels. The resulting graphs have $\sim$91K edges. An SVM with an RBF kernel trained on 4K randomly sampled supervoxels provides the support vectors for the kernel transform in our approach.

Analysis A summary of the segmentation performance is provided in Table 4.3. Example segmentations are provided in Figure 4.6. Once again, we observe the trend in which structured learning yields higher performance than unstructured approaches and non-linear models outperform linear models. Comparing to [88], we can see that our approach of jointly learning the data and pairwise parameters in an SSVM framework

Table 4.3: Segmentation results for the mitochondria EM dataset. We measure segmentation performance using the *Jaccard index*.

Linear SVM	Linear SSVM/CRF	RBF SVM	RBF+ SSVM/CRF	Codebook SVM	Codebook SSVM/CRF	Lucchi et al. [88]	**Kernelized SSVM/CRF**
73%	79%	75%	80%	75%	80%	80%	**84%**



Ground truth | Linear SVM | RBF SVM | Codebook SSVM/CRF | Kernelized SSVM/CRF

Figure 4.6: Segmentation results on the EM dataset. The kernelized SSVM correctly segments all mitochondria in this example, while other methods fail to detect some mitochondria, poorly delineate certain boundaries, or erroneously insert extra regions. Note that the data and segmentations are 3-dimensional – the images above correspond to slices through the test volume.

is superior to learning them independently and then using CRF energy minimization for inference.

4.5.5 Discussion

The expense of training is an important consideration for any classification-based approach. As previously noted, an SSVM can, in principle, be trained using a non-linear kernel. However, this is not feasible in practice as the number of kernel evaluations grows quadratically with the size of the graph. We keep the cost linear by applying a kernel-transform to the features. But to do so, we must first train a standard non-linear SVM to obtain the support vectors used in the kernel transform. The training time of this step could be a potential cause for concern, as it incurs the same quadratic cost. Fortunately, it can be kept in check using known techniques such as randomly sampling the data or iteratively mining for hard examples [29]. However, these techniques cannot

be applied to directly speed up the SSVM because they disregard the structure of the graph, which is essential for learning.

Note that while we only used Gaussian radial basis functions in our experiments, they are not required by design and one could easily substitute them with other kernels such as Polynomial or Hyperbolic tangent.

4.6 Conclusion

In this chapter, we introduced a technique to leverage the power of non-linear kernels in a structured prediction framework by applying a kernel transform to the feature vectors. Results on three different datasets demonstrate the advantages of our approach. Although this work focuses on segmentation, the concept should generalize to other structured prediction problems such as gene sequencing and natural language parsing.

Although the use of the max-margin method led to better empirical results compared to the method introduced in Chapter 3, one issue persists with the computation of the most violated constraint. For the kind of loopy graphs we considered, exact inference is intractable and the most violated constraints can only be approximated, voiding the optimality guarantees of the structured SVM's cutting plane algorithm. In the next chapter we will study a new training method based on stochastic gradient methods that alleviates this issue.

CHAPTER

FIVE

LEARNING FOR STRUCTURED PREDICTION USING STOCHASTIC DESCENT WITH WORKING SETS

5.1 Introduction

While the previous chapter focused on designing more discriminative features for the unary term of a CRF, we now turn our attention to learning the model parameters. We have seen that an as alternative to maximum likelihood and maximum *a posteriori* learning, the max-margin criteria enjoy the advantage of avoiding the need to estimate the computationally difficult partition function and being able to optimize for many different performance metrics. A particularly successful large-margin formulation is the structured support vector machine (SSVM) [136], where the learning objective is to minimize a regularized hinge loss due to the violation of a set of soft margin constraints. This can be solved iteratively using the SSVM cutting plane algorithm [136] or by solving the equivalent unconstrained optimization problem using subgradient based methods [96, 102, 116, 149]. Both approaches require finding at each iteration the *most violated constraint*, namely the labeling that maximizes the margin-sensitive hinge loss [136], which is necessary for obtaining a valid cutting plane or a true subgradient of the objective. Finding such constraints is, however, intractable in loopy graphical models, such as the MRFs and CRFs usually used in image segmentation. Although approximate maximizers can be obtained by approximate inference, such as belief propagation [99] and graph cuts [14], and used as substitutes, the approximation can sometimes be imprecise enough to have a major impact on learning: An unsatisfac-

tory constraint can cause the cutting plane algorithm to prematurely terminate if the new constraint does not have a higher hinge loss than all previous constraints; it can also induce erratic behavior of subgradient-based methods when the implied descent direction is too far away from any true subgradients. These phenomena therefore make the learning process more susceptible to failure.

In this chapter, we propose to use a working set of constraints to increase robustness of approximate subgradient descent based learning. The resulting algorithm is particularly suited for minimizing the margin-sensitive hinge loss in the SSVM formulation [136] when the most violated constraints and hence the resulting subgradients are not exact. We show that the proposed method is able to obtain better subgradient approximations by computing them with respect to the whole working set, as opposed to existing approaches where only the last constraint is considered. Therefore, we are able to obtain sufficiently reliable approximate subgradients even when those due to individual constraints are noisy. This further enables us to replace the most violated constraints with randomly sampled labelings, thus avoiding the need to perform inference at all during learning. The use of sampling leads to decreased learning time while still maintaining good levels of performance. We demonstrate the strength of our method on the task of learning CRF models for image segmentation.

The rest of this chapter is organized as follows. We discuss the prior work in Section 5.2 and provide the background on the large-margin framework and learning techniques based on subgradient descent in Section 5.4. Section 5.5 describes our working set based algorithm in detail and analyze its properties. We present the experimental results in Section 5.6 and conclude in Section 5.7.

5.2 Related work

Maximum margin learning of CRFs was first formulated in the max-margin Markov networks (M^3N) [133], whose objective is to minimize a margin-sensitive hinge loss between the ground-truth labeling and all other labelings for each training example. This is especially appealing for learning CRFs with loopy structures, due to its more objective-driven nature and its complete bypassing of the partition function that presents a major challenge to maximum likelihood based approaches. Nevertheless, the number of constraints in the resulting quadratic program (QP) is exponential in the size of the graph,

hence making it a highly non-trivial problem. In M^3N this is handled by rewriting the QP dual in terms of a polynomial number of marginal variables, which can then be solved by a coordinate descent method analogous to the sequential minimal optimization (SMO) [110]. However, solving such a QP is not tractable for loopy CRFs with high tree widths that are often needed in many computer vision tasks and even solving it approximately can become overwhelmingly expensive on large graphs.

Structured SVMs (SSVM) [136] optimize the same kind of objective as M^3N, while allowing for a more general class of loss functions. It employs a cutting plane algorithm to iteratively solve a series of increasingly larger QPs, which makes learning more scalable. However, the cutting plane algorithm requires the computation of the most violated constraints, namely the labeling maximizing the hinge loss [136]. This involves performing the *loss augmented inference* [132], which makes it intractable on loopy CRFs. Though approximate constraints can be used [31], they make the cutting plane algorithm susceptible to premature termination and can lead to catastrophic failure. Another alternative proposed in [131] is to perform the optimization over a much smaller set of labelings for which exact optimization can be performed with graph cuts. An important limitation of these methods is in solving the QP, which can become slow as the set of constraints grows larger after each iteration, especially when the dimensionality of the feature space is also high. In order to speed up the computation, [56] proposed a caching strategy where instead of performing the loss augmented inference at every iteration, the algorithm first tries to construct a sufficiently violated constraint from the cache. The use of a cache is similar in spirit to maintaining the working set presented in this chapter. However, the goal of [56] is to decrease the number of calls to the separation oracle while our method aims at increasing the robustness of approximate subgradient descent-based learning.

An alternative to solving the quadratic program deterministically is to employ stochastic gradient or subgradient descent. This class of methods has been studied extensively for non-structured prediction problems [119]. In the context of structured prediction, learning can be achieved by finding a convex-concave saddle-point and solving it with a dual extra-gradient method [134]. In [116] max-margin learning is solved as an unconstrained optimization problem and subgradients are used to approximate the gradient in the resulting non-differentiable problem. This method trades optimality for a lower complexity, making it more suitable for large-scale problems. The approach

of [96] proposes a perceptron-like algorithm based on an update whose expectation is close to the gradient of the true expected loss. However, the soundness of these methods heavily depends on the assumption that a valid subgradient is obtained at each iteration. Hence they become much less reliable when the subgradients are noisy due to inexact inference, as is the case for loopy CRFs.

The recently proposed SampleRank [148] avoids performing inference altogether during learning; Instead it samples labelings at random using Markov chain Monte Carlo (MCMC). At each step, parameters are updated with respect to a pair of sampled labelings. Though achieving notable speed improvement, the method does not in fact optimize the actual hinge loss but rather a loose upper bound on it. Hence, unlike our method, it solves a problem that is substantially different from the original max-margin formulation.

5.3 Formulation

5.4 Max-margin Learning of CRFs

Conditional random fields (CRF) [77] are graphical models used to encode relationships between a set of input and output variables. Given its parameters $\mathbf{w}$, a CRF predicts the labeling Y for a given input X by maximizing some score function $S_{\mathbf{w}} : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, i.e.,

$$\hat{Y} = \arg\max_{Y \in \mathcal{Y}} S_{\mathbf{w}}(Y) = \arg\max_{Y \in \mathcal{Y}} \mathbf{w}^T \Psi(X, Y) \tag{5.1}$$

The score is usually expressed as a linear function of $\mathbf{w}$ and can be written as $\mathbf{w}^T \Psi(X, Y)$, where the vector $\Psi(X, Y)$ is the *feature map* corresponding to the input X and the labeling Y. The fundamental properties of random fields imply that the feature map $\Psi(X, Y)$ and hence the score $S_{\mathbf{w}}$ decompose into sums over individual nodes and edges for any pairwise CRFs [12]. For a comprehensive review of CRF models, we refer readers to Chapter 2 or [77] and [129].

5.4.1 Discriminative Learning

Discriminative learning uses the labeled training data to learn the CRF parameters so that the inferred labeling of the CRF is "close" to that of the ground truth, defined as yielding a low *loss*. More specifically, given a set of N training examples $\mathcal{D} =$

$((X^1, Y^1), \ldots, (X^N, Y^N))$ where $X^i \in \mathcal{X}$ is an input example, such as the image or features associated to it, and $Y^i \in \mathcal{Y}$ is the associated labeling, the learning task consists in finding model parameters $\mathbf{w}$ that achieve low empirical loss subject to some regularization. In other words, we seek

$$\begin{aligned} \mathbf{w}^* &= \arg\min_{\mathbf{w}} \mathcal{L}(\mathcal{D}, \mathbf{w}) \\ &= \arg\min_{\mathbf{w}} \sum_{(X^n, Y^n) \in \mathcal{D}} l(X^n, Y^n, \mathbf{w}) + R(\mathbf{w}), \end{aligned} \tag{5.2}$$

where l is the surrogate loss function and $R(\mathbf{w})$ is the regularizer (typically the L2 norm). The most common choice of l is the hinge loss, as used in [133, 136], which will be described later on in this section. Note that the definition of the surrogate loss l depends on the score function $S_{\mathbf{w}}$, since the goal of learning is to make the maximizer of $S_{\mathbf{w}}$ a desirable output for the given input.

5.4.2 Max-margin Formulation

The max-margin approach is a particular instance of discriminative learning, where parameter learning is formulated as a quadratic program (QP) with soft margin constraints [136]:

$$\begin{aligned} &\min_{\mathbf{w}, \mathbf{m}\xi \geq \mathbf{0}} \frac{1}{2} ||\mathbf{w}||^2 + C \sum_{n=1}^{N} \xi_n \\ &\text{s.t. } \forall n : S_{\mathbf{w}}(Y^n) \geq \max_{Y \in \mathcal{Y}_n} (S_{\mathbf{w}}(Y) + \Delta(Y^n, Y)) - \xi_n, \end{aligned} \tag{5.3}$$

where $\mathcal{Y}_n$ is the set of all possible labelings for example n, the constant C controls the trade-off between margin and training error, and the task loss Δ measures the closeness of any inferred labeling Y to the ground truth labeling Y^n.

The QP can be converted to an unconstrained optimization problem by incorporating the soft constraints directly into the objective function, yielding:

$$\begin{aligned} &\min_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \\ &\min_{\mathbf{w}} \frac{1}{2C} ||\mathbf{w}||^2 + \sum_{n=1}^{N} [S_{\mathbf{w}}(Y^*) + \Delta(Y^n, Y^*) - S_{\mathbf{w}}(Y^n)]_+, \end{aligned} \tag{5.4}$$

where

$$Y^* = \arg\max_{Y \in \mathcal{Y}_n} (S_{\mathbf{w}}(Y) + \Delta(Y^n, Y)). \tag{5.5}$$

It is easy to see that Equation 5.4 has the same form as Equation 5.2 where the hinge loss is used as the surrogate loss l, i.e.,

$$l(Y^n, Y^*, \mathbf{w}) = [S_{\mathbf{w}}(Y^*) + \Delta(Y^n, Y^*) - S_{\mathbf{w}}(Y^n)]_+ \tag{5.6}$$

For most existing approaches, a key challenge to solving Equation 5.4 effectively is the loss-augmented inference, i.e., finding Y^*.

5.4.3 Stochastic Subgradient Descent

The objective function of Equation 5.4 can be minimized via stochastic subgradient descent, similar to [96, 116]. This class of methods iteratively computes and steps in the opposite direction of a subgradient vector with respect to a example X^n chosen by picking an index $n \in \{1 \dots N\}$ uniformly at random. We then replace the objective in Equation 5.4 with an approximation based on the training example (X^n, Y^n), yielding:

$$f(Y^n, Y^*, \mathbf{w}) = l(Y^n, Y^*, \mathbf{w}) + \frac{1}{2C} ||\mathbf{w}||^2 . \tag{5.7}$$

A subgradient of the convex function $f : \mathcal{W} \to \mathbb{R}$ at $\mathbf{w}$ is defined as a vector $\mathbf{g}$, such that

$$\forall \mathbf{w}' \in \mathcal{W}, \mathbf{g}^T(\mathbf{w}' - \mathbf{w}) \leq f(\mathbf{w}') - f(\mathbf{w}). \tag{5.8}$$

The set of all subgradients at $\mathbf{w}$ is called the *subdifferential* at $\mathbf{w}$ and is denoted $\partial f(\mathbf{w})$. The subdifferential is always a non-empty convex compact set.

A valid subgradient $g(Y^n, Y^*, \mathbf{w})$ with respect to the parameter $\mathbf{w}$ can always be computed as the gradient of $f(Y^n, Y^*, \mathbf{w})$ at Y^*. Hence for the hinge loss, it can be computed as:

$$\frac{\partial f(Y^n, Y^*, \mathbf{w})}{\partial \mathbf{w}} = \psi(Y^*) - \psi(Y^n) + \frac{\mathbf{w}}{C}. \tag{5.9}$$

This results in a simple algorithm that iteratively computes and steps in the direction of the negative subgradient. In order to guarantee convergence, the step size $\eta^{(t)}$ has to satisfy the following conditions :

$$\lim_{T \to +\infty} \sum_{t=1}^{T} \eta^{(t)} = \infty \quad \text{and} \lim_{T \to +\infty} \quad \sum_{t=1}^{T} (\eta^{(t)})^2 < \infty. \tag{5.10}$$

For loopy CRFs, however, true subgradients of the hinge loss cannot always be obtained due to the intractability of loss-augmented inference. This can lead to erratic behavior due to noisy subgradient estimates and loss of performance.

5.5 Estimating Subgradient Using Working Sets

Our algorithm aims at better estimating an approximate subgradient of $f(Y^n, Y, \mathbf{w})$ by using working sets of constraints, denoted $\mathcal{A}^n$, for learning loopy CRFs where exact inference is intractable. The algorithm we propose is outlined in Algorithm 3. It first solves the loss-augmented inference to find a constraint Y^* and add it to the working set $\mathcal{A}^n$. It then steps in the opposite direction of the approximate subgradient computed as an average over the set of violated constraints belonging to $\mathcal{A}^n$.

Algorithm 3

INPUTS :
  $\mathcal{D}$: Training set of N examples.
  β : Learning rate parameter.
  $\mathbf{w}^{(1)}$: Arbitrary initial values, e.g., 0.
OUTPUT : $\mathbf{w}^{(T+1)}$
Initialized $\mathcal{A}^n \leftarrow \emptyset$ for each $n = 1 \dots N$
for $t = 1 \dots T$ **do**
  Pick some example (X^n, Y^n) from $\mathcal{D}$
  $Y^* = \arg\max_{Y \in \mathcal{Y}_n}(S_{\mathbf{w}}(Y) + \Delta(Y^n, Y))$
  $\mathcal{A}^n \leftarrow \mathcal{A}^n \cup \{Y^*\}$
  $\mathcal{A}^{n'} \leftarrow \{Y \in \mathcal{A}^n \mid l(Y, Y^n, \mathbf{w}^{(t)}) > 0\}$
  $\eta^{(t)} \leftarrow \frac{\beta}{t}$
  $\mathbf{g}^{(t)} \leftarrow \frac{1}{|\mathcal{A}^{n'}|} \sum_{Y \in \mathcal{A}^{n'}} \frac{\partial f(Y^n, Y, \mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta^{(t)} \mathbf{g}^{(t)}$
end for

Hence unlike dual averaging methods [102, 149] that aggregate over all previous subgradients, our algorithm only considers the subset of active, namely violated, constraints when computing the parameter updates. Therefore all subgradients are computed with respect to the parameters at the *current* iteration, as opposed to using their historical values. This produces more meaningful descent directions, as evidenced by the results in Section 5.6.

We now analyze the convergence properties of the algorithm presented in Algorithm 3. Although finding true subgradients as defined in Equation 5.8 cannot be guaranteed for loopy CRFs, interesting results can still be obtained even if one can

only find an approximate ϵ-subgradient $\mathbf{g}$, as defined in [121]:

$$\forall \mathbf{w}' : \mathbf{g}^T(\mathbf{w} - \mathbf{w}') \geq f(\mathbf{w}) - f(\mathbf{w}') - \epsilon \tag{5.11}$$

The convergence properties of ϵ-subgradient descent methods were studied in [116, 117, 121]. The "regret" (i.e., loss) of the parameter vector $\mathbf{w}$ can be bounded as follows (the re-derivation of the proof using our notation is given in the supplementary material):

$$\mathbb{E}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|_2^2 \leq \frac{G^2}{\lambda^2 t} + \frac{\epsilon}{\lambda}, \tag{5.12}$$

where G is a constant satisfying the condition $||\mathbf{g}||^2 \leq G^2$ and $\lambda = \frac{1}{C}$.

Given that the choice of the step size satisfies Equation 5.10, we can see that the first term on the right side of Equation 5.12 goes to 0 so stochastic ϵ-subgradient descent converges to a certain distance ϵ to the optimum. The key to improving convergence is thus to obtain more accurate ϵ-subgradients, and we show below how this is achieved through the use of working sets.

Let $\mathbf{g}_1, \ldots, \mathbf{g}_m \in \mathbb{R}^d$ be the approximate subgradients with respect to example (X^n, Y^n) of $\mathcal{L}$ for labelings in the working set $\mathcal{A}^n$ that still violates the margin constraint at a given iteration. Assume that each $\mathbf{g}_i \in \mathbb{R}^d$ comes from some distribution with mean $\mathbf{m}\mu_i \in \partial\mathcal{L}(\mathbf{w})$ and bounded variance.

Let $\mathbf{m}\delta_i = \mathbf{g}_i - \mathbf{m}\mu_i$ be the difference between approximate ϵ-subgradient $\mathbf{g}_i$ and true ϵ-subgradient $\mathbf{m}\mu_i$, and assume that all $\mathbf{m}\delta_i$ are independent of one another. Note that, by definition, each $\mathbf{m}\delta_i$ has zero expectation and hence their average $\mathbf{m}\bar{\delta} = \frac{1}{m}\sum \mathbf{m}\delta_i = \frac{1}{m}\sum \mathbf{g}_i - \frac{1}{m}\sum \mathbf{m}\mu_i$.

Therefore, using Hoeffding's inequality [47] and the union bound, we can show that the average error $\mathbf{m}\bar{\delta}$ concentrates around its expectation, i.e., 0 in this case, as the number of violated constraints in the working set m increases:

$$\Pr\left(\left|\left|\mathbf{m}\bar{\delta}\right|\right| \geq r\right) \leq 2d \exp\left(\frac{-mr^2}{2G^2}\right), \tag{5.13}$$

The convexity of the subdifferential $\partial\mathcal{L}(\mathbf{w})$ implies that $\mathbf{m}\bar{\mu} = \frac{1}{m}\sum_i \mathbf{m}\mu_i \in \partial\mathcal{L}(\mathbf{w})$. Therefore the probability of $\mathbf{g}^{(t)} \triangleq \frac{1}{m}\sum \mathbf{g}_i$ being more than a distance r away from any true subgradient is bounded by Equation 5.13 as well.

Algorithm 3 solves the loss-augmented inference to generate new constraints, which can be expensive to compute. The analysis presented in Section 5.5 suggests that

Algorithm 4

INPUTS :
  $\mathcal{D}$: Training set of N examples.
  $\mathcal{Q}$: MCMC walker.
  β : Learning rate parameter.
  $\mathbf{w}^{(1)}$: Arbitrary initial values, e.g., 0.
OUTPUT : $\mathbf{w}^{(T+1)}$
Intialized $\mathcal{A}^n \leftarrow \emptyset$ for each $n = 1 \dots N$
for $t = 1 \dots T$ **do**
  Pick some example (X^n, Y^n) from $\mathcal{D}$
  Sample Y^* according to $\mathcal{Q}(w^{(t)}, Y^n)$
  $\mathcal{A}^n \leftarrow \mathcal{A} \cup \{Y^*\}$
  $\mathcal{A}^{n'} \leftarrow \{Y \in \mathcal{A}^n \mid l(Y, Y^n, \mathbf{w}^{(t)}) > 0\}$
  $\eta^{(t)} \leftarrow \frac{\beta}{t}$
  $\mathbf{z} \leftarrow \mathbf{w}^{(t)}$
  for $Y \in \mathcal{A}^{n'}$ **do**
    $\mathbf{g}^{(t)} \leftarrow \frac{1}{|\mathcal{A}^{n'}|} \frac{\partial f(Y^n, Y, \mathbf{w}^{(t)})}{\partial \mathbf{w}^{(t)}}$
    $\mathbf{z} \leftarrow \mathbf{z} - \eta^{(t)} \mathbf{g}^{(t)}$ * atomic update *
  end for
  $\mathbf{w}^{(t+1)} \leftarrow \mathbf{z}$
end for

it is possible to use a sampling method instead of the loss-augmented inference to obtain new constraints, and under similar assumptions the average subgradient $\bar{g}$ still converges to a valid subgradient. Based on this observation, we propose an adaptation of Algorithm 3 that uses sampling instead of solving the loss-augmented inference. This adapation described in Algorithm 4 generates new constraints using an MCMC walker denoted $\mathcal{Q}$ similar to the one described in [148]. We also replace the standard update of Algorithm 3 by a sequence of atomic updates that has been shown to improve the speed of convergence [148]. Concerning the practicality, we would like to point out that the working set does not lead to a significant increase in memory as we only need to store the feature maps rather than the whole labellings.

5.6 Experimental Results

We apply our algorithm to image segmentation. We first briefly describe how a CRF model is used for this task and then present our experimental results on two distinct datasets to demonstrate the effectiveness of our method.

5.6.1 CRF for Image Segmentation

As a standard preprocessing step, we perform a preliminary over-segmentation of our input image into superpixels [1] using SLIC [5]. The CRF $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is thus defined so that each node $i \in \mathcal{V}$ corresponds to a superpixel and there is an edge $(i, j) \in \mathcal{E}$ between two nodes i and j if the corresponding superpixels are adjacent in the image. Let $Y = \{y_i\}$ for $i \in \mathcal{V}$ denote the labeling of the CRF which assigns a class label y_i to each node i.

The score function associated with the CRF can then be written as

$$S_{\mathbf{w}}(Y) = \sum_{i \in \mathcal{V}} D_i(y_i) + \sum_{(i,j) \in \mathcal{E}} V_{ij}(y_i, y_j), \tag{5.14}$$

where D_i is the unary data term and V_{ij} is the pairwise spatial term. Both D_i and V_{ij} are linear in the CRF parameters $\mathbf{w}$ and also depend on the observed data X in addition to the labeling Y. For inference, we use graph cuts when the corresponding energy function (i.e., negated score) is submodular [67] and belief propagation otherwise.

[1] Or supervoxels in the case of volumetric data.

A natural choice for the task loss Δ (Equation 5.3) is the per-superpixel 0-1 loss $\Delta(Y^n, Y) = \sum_{i \in \mathcal{V}} \mathcal{I}(y_i \neq y_i^n)$, which penalizes all errors equally. However, in image segmentation, it is common for certain classes to occur much more frequently than others. To ensure good performance across all classes, we adopt a loss function that weighs errors for a given class inversely proportional to the frequency with which it appears in the training data.

5.6.2 Methods

In the following, we will compare our learning methods (referred as **Working sets + inference** and **Working sets + sampling**) with the following baselines. We also experimented with averaging all past subgradients [102, 149], which did not produce meaningful results for our task.

- **Linear SVM** – A linear SVM classifying each sample independently (i.e., without CRF).
- **SSVM** – The cutting plane algorithm described in [136].
- **SampleRank** – The method described in [148].
- **SGD + inference** – solve the loss-augmented inference using graph-cuts or belief-propagation. This algorithm is the SGD (subgradient descent) formulation of [116].
- **SGD + sampling** – Instead of performing inference, use MCMC to sample constraints from a distribution targeting the loss-augmented score. This is equivalent to the method named "SampleRank SVM" described in [148].

In all cases, we used a decreasing step size rule of $\frac{1}{\sqrt{t}}$ and used cross-validation on the training set to determine the regularization constant. The results reported for the sampling method and SampleRank were averaged over 5 runs.

5.6.3 MSRC Dataset

The MSRC dataset is a popular multi-class object segmentation dataset containing 591 images with objects from 21 categories. Training and testing are done using the standard split of the dataset [123], and we used the annotated images provided by [92]. We

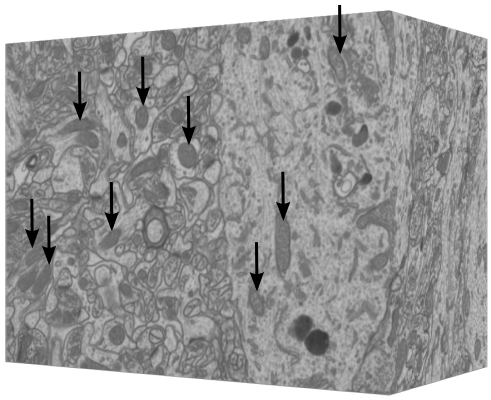


Figure 5.1: A nearly isotropic stack of neural tissue acquired using EM microscopy annotated for training and testing. This stack contains 1065 images of 2048×1536 pixels, and was used for the task of segmenting mitochondria, indicated by arrows.

Figure 5.2: Example segmentations from the MSRC dataset. Best viewed in color.

Table 5.1: MSRC segmentation results. We follow the standard reporting procedure. For each category, the pixel-wise classification rate is provided. Global pixel-wise accuracy and average per-category scores provide measures for overall performance. Bold entries are used for each part separately to indicate best performance. Note that all the methods in the top part of the table were optimized for the average score, for which our method achieves the best results. We also include the results of two state-of-the-art algorithms as reported in [152].

	building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Average**	Global
Linear SVM	63	88	77	90	85	91	89	59	86	88	94	85	80	53	93	85	79	65	50	82	0	**74.8**	79.6
SSVM [136]	69	91	82	80	82	92	78	68	85	79	85	90	77	58	95	82	87	74	44	80	44	**77.1**	82.3
SampleRank	66	91	80	84	80	92	76	70	82	65	89	91	85	54	93	80	88	69	48	89	40	**76.7**	82.2
SGD + sampling	67	93	80	82	75	93	64	67	84	74	81	93	82	43	93	75	88	68	52	73	48	**75.4**	82.1
SGD + inference [116]	69	91	84	87	82	92	80	64	86	82	85	83	76	61	95	82	88	73	43	74	35	**76.7**	82.2
Working set + sampling	71	90	87	91	84	92	83	74	82	84	85	87	66	46	94	84	88	79	49	74	37	**77.2**	83.3
Working set + inference	67	89	85	93	79	93	84	75	79	87	89	92	71	46	96	79	86	76	64	77	50	**78.9**	83.7
Ladický et al. [75]	73	93	82	81	91	98	81	83	88	74	85	97	79	38	96	61	90	69	48	67	18	**75.8**	85.0
Yao et al. [152]	67	92	80	82	89	97	86	83	86	79	94	96	85	35	98	70	86	78	55	62	23	**77.4**	84.4

Table 5.2: Segmentation performance measured with the Jaccard index for the mitochondria EM dataset. We report results for two different set of features (see text for full description). Note that the original features were already kernelized with a RBF-SVM in [88].

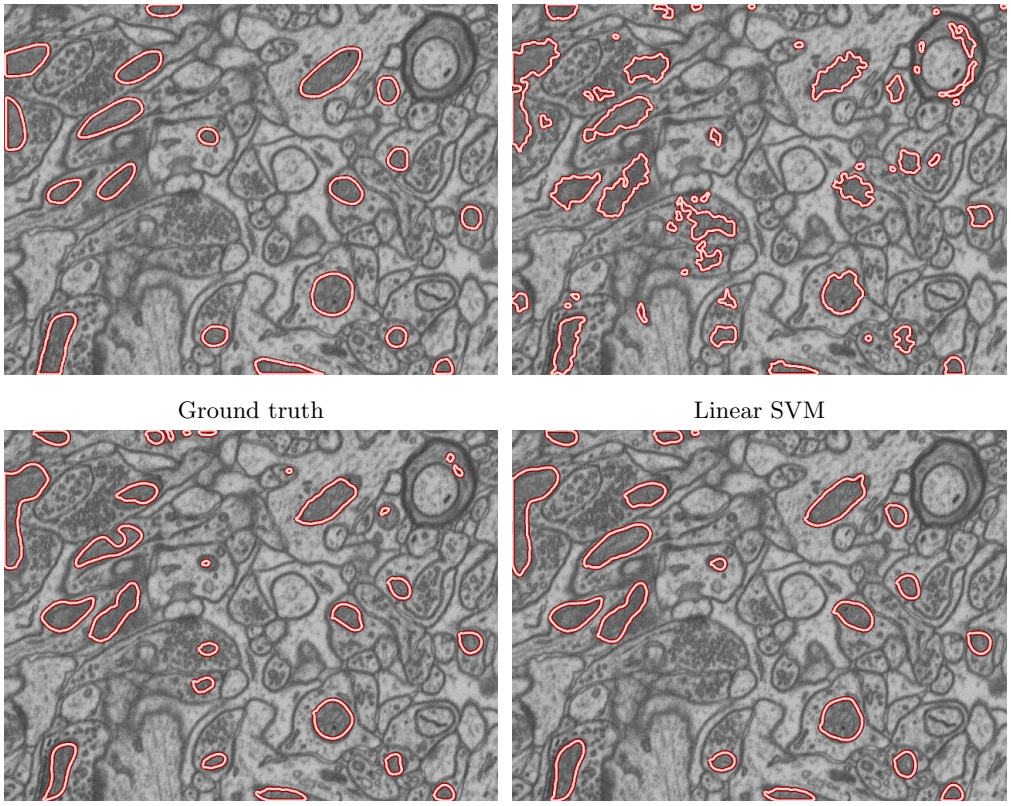
Features	SVM	Lucchi [88]	SSVM [136]	SampleRank [148]	SGD + sampling	SGD + inference [116]	Working set + sampling	Working set + inference
Original	73.0%	80.0%	80.5%	81.2%	77.5%	79.9%	83.0%	**84.5%**
Kernelized	75.4%	-	83.5%	82.9%	80.1%	81.5%	84.4%	**86.7%**

extract feature vectors by first over-segmenting images using SLIC superpixels [5]. We then extract SIFT descriptors and color histograms from image patches surrounding each superpixel centroid. We then create a bag-of-words descriptor by first generating a dictionary containing 1,000 words for SIFT features, and 400 words for color histograms are constructed using k-means on extracted features. We also include location information as in [74] and the unary potentials of [69]. The resulting feature vector x_i is used to train the various methods. Similarly to [123], the pairwise term we used was made gradient-adaptive by including parameters for each discretized image gradient level. In a similar fashion, it also considers geometric relationships such as "sky appears above grass".

Table 5.1 summarizes the segmentation performance of the various approaches and example segmentations appear in Figure 5.2. The quantitative results show that the working set of constraints improves the average score regardless of whether inference or sampling was used during learning. The results obtained by the sampling approach are close to those from using inference, but with a significantly lower running time as shown in Table 5.3. In addition to the baseline methods described above, we compare our approach to state-of-the-art approaches [75, 152]. We achieve the best results in terms of the average score for which we optimize our algorithm.

5.6.4 Electron Microscopy Dataset

Here, we perform mitochondria segmentation in 3D using the large image stack from Figure 5.1. This electron miscropy dataset is publicly available at `http://cvlab.epfl.ch/data/em`. Performance is measured by the *Jaccard index* commonly used for



Ground truth | Linear SVM

SGD + inference [116] | Working set + inference

Figure 5.3: Segmentation results on the EM dataset.

image segmentation [26]. The Jaccard index is the ratio of the areas of the intersection between what has been segmented and the ground truth, and of their union. It is written as:

$$\text{Jaccard index} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive} + \text{False Negative}} .$$

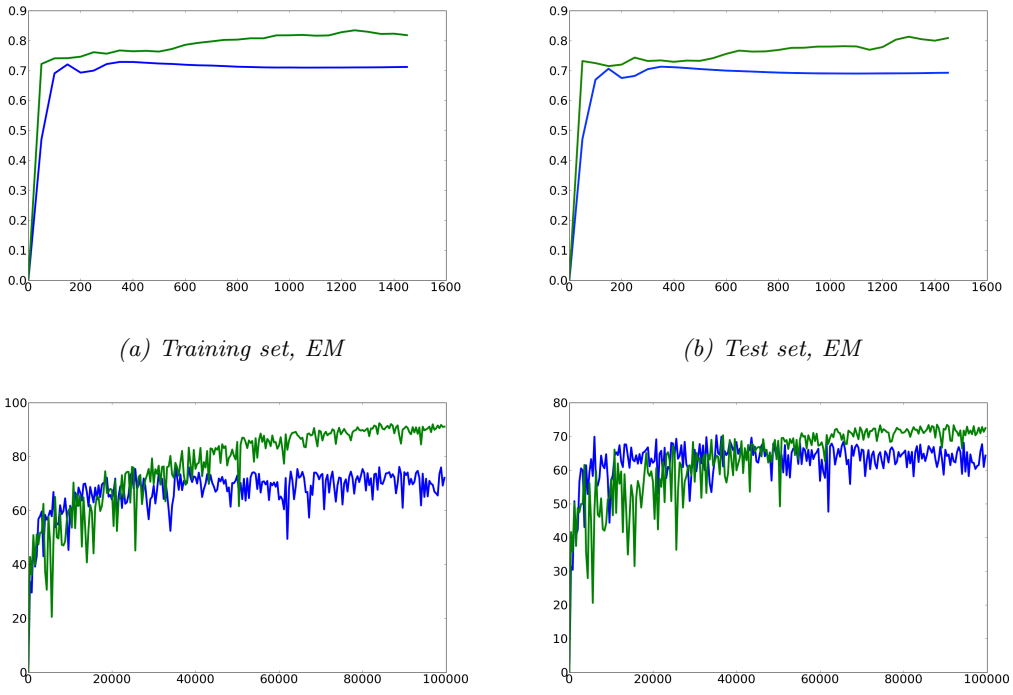
The segmentation process begins by over-segmenting the volume using SLIC supervoxels [5]. For each supervoxel, we extract a feature vector that captures local shape and texture information using Ray descriptors [88] and intensity histograms. These feature vectors are used to train each baseline method, as well as our model. In a second set of experiments, we also transform the features using the kernel method of Chapter 4. The original feature vectors are 120-dimensional and are thus mapped to a higher dimensional space. The details are described in Chapter 4. Due to the high cost of labeling such large volumes, our experiments are restricted to the two same subvolumes containing $1024 \times 768 \times 165$ voxels presented in Chapter 3 and 4. The first subvolume was used to train the various methods while the second one was used for testing. Each subvolume contains $\sim$13K supervoxels. The resulting graphs have $\sim$91K edges. Example segmentations are shown in Figure 5.3 and quantitative results are provided in Table 5.2. The increased reliability due to the use of working sets leads to higher scores for both the inference and sampling methods. The inference version of our algorithm outperforms the previous state-of-the-art [88].

5.6.5 Time analysis

We conducted a time analysis of the standard subgradient method of [116] against the 2 versions of the algorithm introduced in this chapter. As shown in Table 5.3, the sampling method is much faster than solving the loss-augmented inference to find the most violated constraint. We can see that the computational overhead due to the working set is of the order of 5% for the sampling method and less than 10% when solving the loss-augmenting inference to find the most-violated constraint. The evolution of the training scores as a function of the number of iterations is shown on Figure 5.4 for the EM and MSRC datasets. The curves clearly show that the working set of constraints leads to a much higher score on both the training and test sets.

Table 5.3: Running time for the EM and MSRC datasets for $T = 1000$ iterations. The computational overhead reported in the brackets is the increase in time resulting from the working set. On both datasets, our method achieves better results at the price of a very slight overhead.

	EM	**MSRC**
SampleRank [148]	2524s	80s
SGD + Sampling	2481s	72s
Working set + Sampling	2619s (+5.5%)	76s (+5.2%)
SGD + inference [116]	5315s	546s
Working set + inference	5842s (+9.9%)	583s (+6.8%)



(a) Training set, EM *(b) Test set, EM*

(c) Training set, MSRC *(d) Test set, MSRC*

Figure 5.4: Evolution of the training and test scores (Jaccard index for EM and average score for MSRC) as a function of the number of iterations t. We report results for the sampling method with and without working set in green and blue respectively.

5.7 Conclusion

We have presented a working set based approximate subgradient descent method for learning graphical models for structured prediction. This is particularly appealing for learning large CRFs with loops, which are common in computer vision tasks, since under these circumstances the use working sets of constraints produces better subgradient estimates and higher-quality solutions. We applied our method to the task of image segmentation, where the results show that it compares favorably against existing algorithms in terms of segmentation accuracy. We also experimentally demonstrated that sampling can replace the more expensive inference step without much performance loss, leading to significantly lower learning time.

Our method makes no assumption that is particular to computer vision and thus is readily applicable to other structured prediction problems. However, our line of research was primarily motivated by the need to more accurately segment synapses and mitochondria in electron microscopy stacks. In the next chapter, we will study how the method proposed in this chapter can be used for real biological studies.

CHAPTER

SIX

ENHANCED TECHNIQUES FOR THE SEGMENTATION OF MITOCHONDRIA

6.1 Introduction

Studying the geometric properties of sub-cellular structures in electron microscopy images is critical for determining the nature of numerous cellular processes. As mentioned in Chapter 3, focused ion beam scanning electron microscopy (FIB-SEM) has become a powerful technology for creating 3D representation of biological specimens at a very high resolution. This technology can already generate a tremendous amount of data, which is likely to grow far beyond its capacity to be segmented manually. In order to perform a reliable analysis of such a big amount of data, a fully automatic method has to provide very accurate results. Although the method proposed in Chapter 5 got us one step closer to a fully automatic solution it does nevertheless make mistakes and still require manual editing before performing a statistical analysis. Figure 6.1 illustrates one of the main problems that impedes a quantitative analysis. Nearby or touching mitochondria are segmented as a single entity preventing from inferring statistics for each individual entity.

In this chapter, we first propose an adaptation of the multi-layer segmentation model proposed in [2] that can encode geometric interactions between the boundary and other regions such as interior/exterior. The use of these additional constraints help the model perform better with regard to the segmentation of mitochondria on the two datasets presented in Chapter 3. The second innovation introduced in this chapter

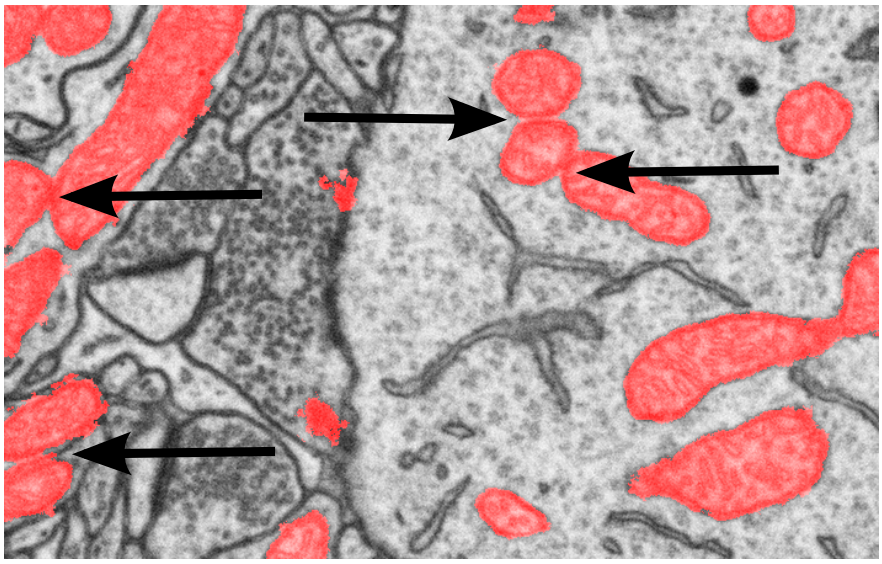


Figure 6.1: The red overlay is the segmentation produced by the algorithm presented in Chapter 5. The black arrows indicate nearby cells that are merged by the segmentation algorithm.

is an interactive splitting procedure that allows users to quickly separate touching mitochondria. We conclude this chapter with a discussion concerning the use of our method for a statistical analysis.

6.2 Encoding geometric interactions

In this section we compare the **SGD + inference** method presented in Chapter 5 with the following two methods:

- **Multi-layer** - An adaptation of the multi-layer model proposed in [2] that can encode geometric interactions between a boundary and other regions such as interior/exterior. As shown in Figure 6.2, the method of [2] extends the multi-surface segmentation method of Li et al. [82] but their graph construction handles topological constraints better while still requiring a single graph cut to extract a segmentation that satisfies the constraints. We used the learning method presented in Chapter 5 to learn the parameters of the multi-layer model proposed in [2].

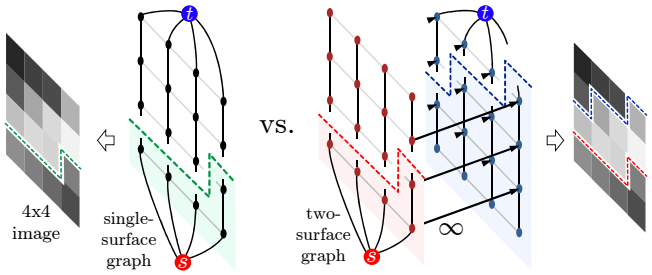


Figure 6.2: LEFT: *s-t* min cut construction corresponding to [82]; any cut must separate the top row from the bottom row. RIGHT: Basic idea from [82]. Each column separates top from bottom at two distinct locations, one forced to be strictly above the other. Figure courtesy of Andrew Delong.

- **Projection** - A variant of the **SGD + inference** method where we project the weights at each iteration of the subgradient descent algorithm to make sure they satisfy the sumodularity conditions presented in Section 2.2.6. This projection allows us to use graph cut to solve the loss-augmented inference, which is guaranteed to find the global optimum for binary labels. This method is similar to the one presented in [131] in which the submodularity condition is satisfied by adding a constraint to the QP formulation of Structured SVM as given in Equation 2.64.

6.3 Results

We present our experimental results for the image segmentation task described in Section 5.6 on the two datasets presented in the Section 3.4.1 of Chapter 3, and shown in Figure 1.5.

The experimental setup is similar to the one described in Section 5.6.4. The segmentation process begins by over-segmenting the volume using SLIC supervoxels [5]. For each supervoxel, we extract a feature vector that captures local shape and texture information using Ray descriptors [88], intensity histograms and the following features computed at five different scales: gradient magnitude, laplacian of gaussian and eigenvalues of the hessian matrix, eigenvalues of the structure tensor. These feature vectors are used to train each baseline method, as well as our model. In a second set of experiments, we also transform the features using the kernel method of Chapter 4. The

Table 6.1: Segmentation performance measured with the Jaccard index (see Section 5.6.4) for the Hippocampus EM dataset.

	SGD + inference	Projection	Multi-layer
Original features	85.2%	84.7%	**90.6%**
Kernelized features	90.2%	91.8%	**92.6%**

Table 6.2: Segmentation performance measured with the Jaccard index (see Section 5.6.4) for the Striatum EM dataset.

	SGD + inference	Projection	Multi-layer
Original features	84.4%	84.2%	**90.6%**
Kernelized features	90.4%	92.1%	**93.6%**

original feature vectors are 140-dimensional and are thus mapped to a higher dimensional space. The details are described in Chapter 4.

Visual results are shown in Figures 6.3 and 6.4 and quantitative results are given in Tables 6.1 and 6.2. The **projection** method suffers from a slight decrease of performance compared to **SGD + inference** for the original features but does better for the kernelized features. The **Multi-layer** method outperforms both the **SGD + inference** and **Projection** methods on both datasets. As seen in Figure 6.5, the **Multi-layer** method also returns predictions for the boundary label. This label is treated as a foreground label for the quantitative evaluation but can still be of interest for biologists interested in studying properties of cell boundaries.

The **Multi-layer** method performs fairly well (92.6% and 93.6% on the two datasets presented in Chapter 3) but some errors persist and have to be eliminated. Some of the remaining errors are shown in Figure 6.6 and include erroneously detecting packs of vesicles or missing mitochondria whose appearances are significantly different from the ones appearing in the training dataset. One of the major problems that impedes a rigorous analysis of the data is when nearby or touching mitochondria are erroneously grouped together. In the next section, we propose a method that requires minimal manual interaction to solve this issue.

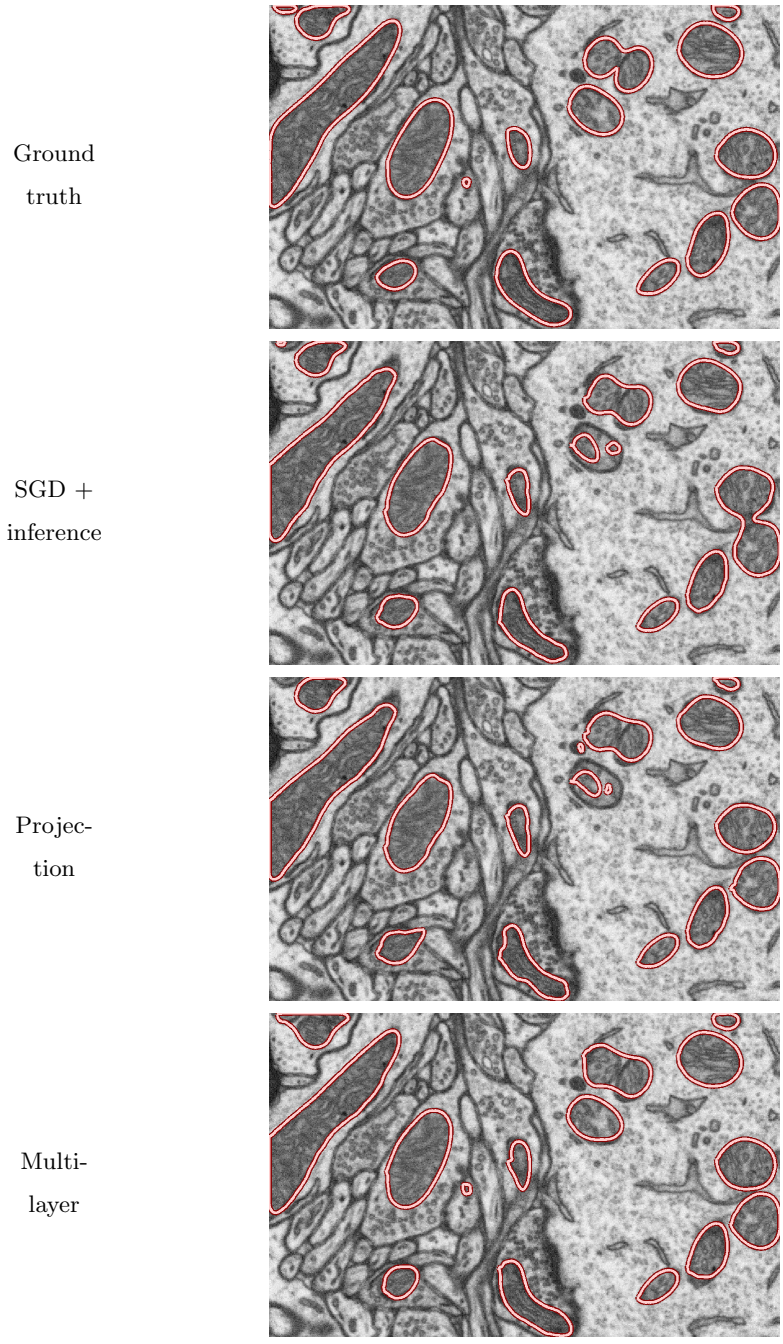


Figure 6.3: Example segmentations produced by different methods using non-kernelized features on the Striatum EM dataset. Best viewed in color.

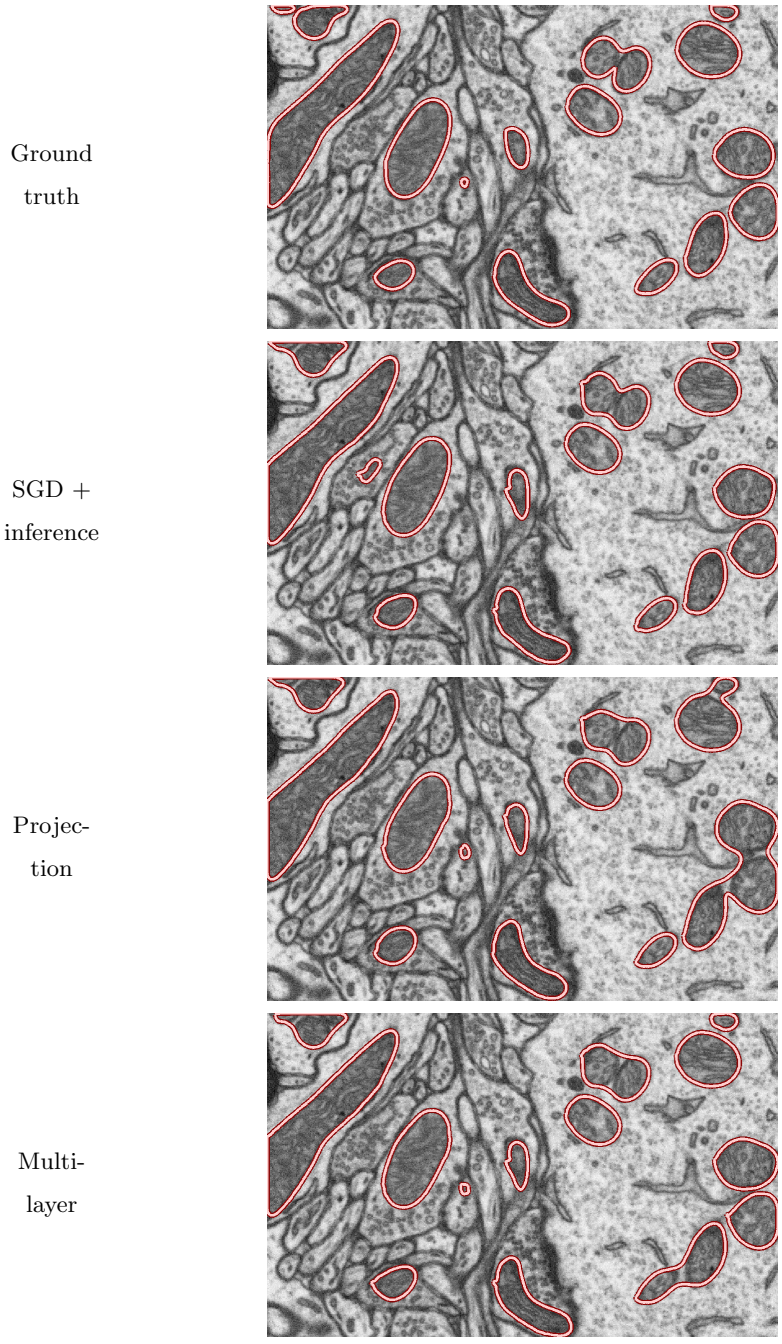


Figure 6.4: Example segmentations produced by different methods using kernelized features on the Striatum EM dataset. Best viewed in color.

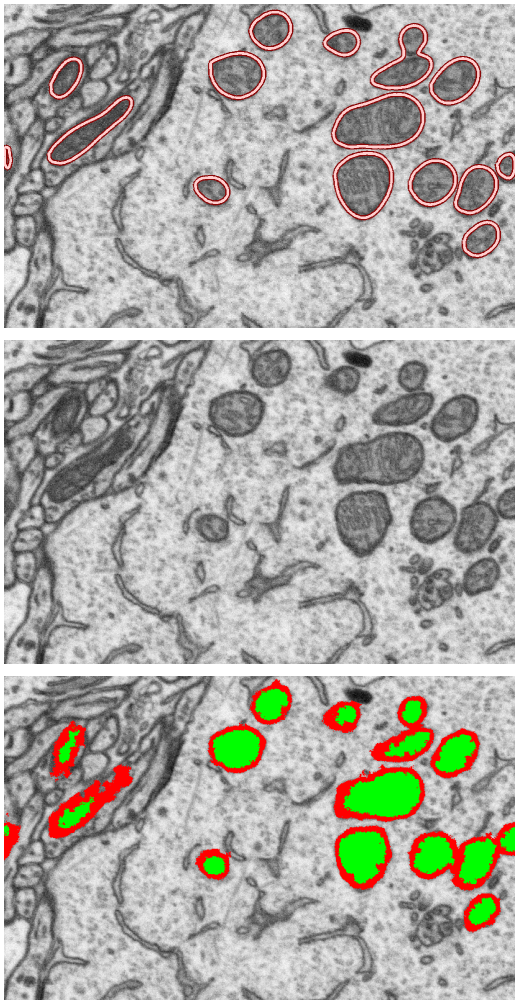


Figure 6.5: Example segmentations produced by the **Multi-layer** method on the Striatum EM dataset. Best viewed in color.

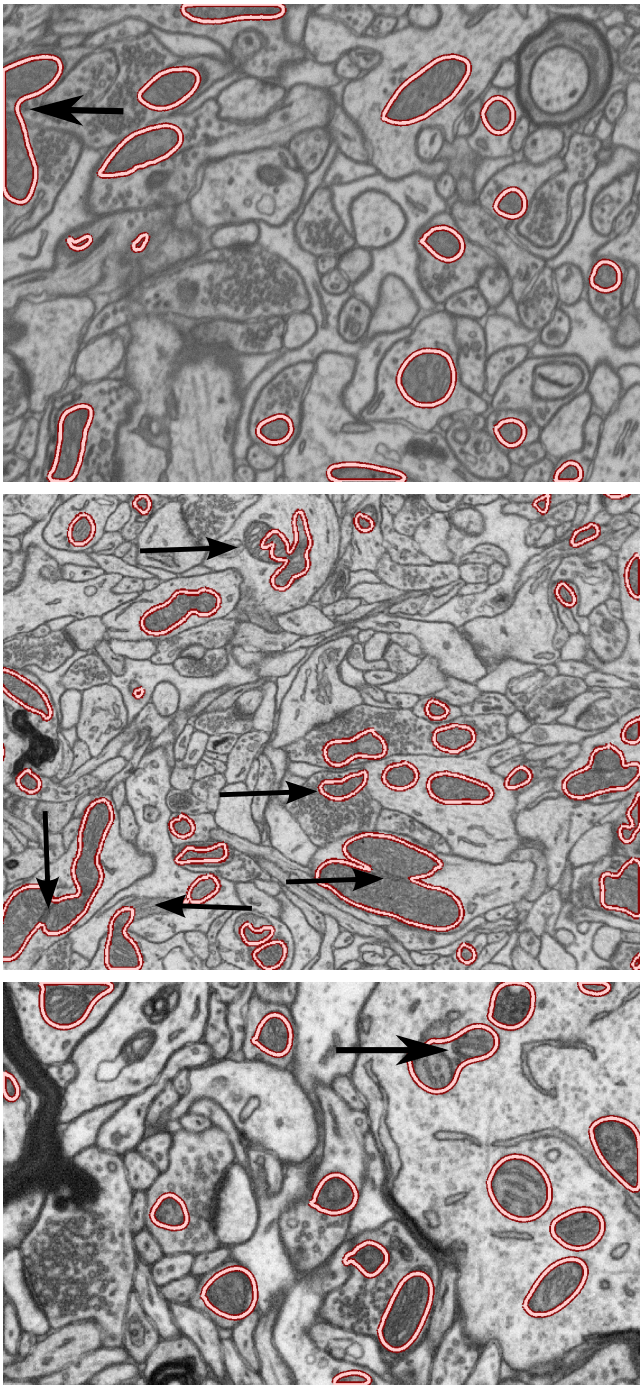


Figure 6.6: The black arrows indicate mistakes made by the **Multi-layer** method on several EM datasets. These errors include erroneously detecting packs of vesicles or missing mitochondria whose apperances are significantly different from the ones appearing in the training dataset. Another major problem that impedes a rigorous analysis of the data is when nearby or touching mitochondria are erroneously grouped together. This problem is discussed in Section 6.4. Best viewed in color.

Table 6.3: Unary weights for splitting procedure.

	Weight	For
$\psi(y_i = 0\|x_i)$	0	$i \in \mathcal{O}$
	∞	$i \notin \mathcal{O}$
$\psi(y_i = 1\|x_i)$	0	$i \in \mathcal{B}$
	∞	$i \notin \mathcal{B}$

6.4 Semi-automated splitting of merged regions

The identification of individual cells is crucial in many cytological applications, in which the expected result is a population count. Although automatic methods such as [24], [23] have been proposed to separate erroneously grouped objects, this process must be performed accurately in order to guarantee correctness of the final analysis results thus justifying the use of a semi-automatic approach where the knowledge of a human expert can be integrated. We used an algorithm similar to the interactive graph-cuts technique proposed by [14]. The user marks certain pixels as "object" or "background" which are used as hard constraints for the segmentation. In the following, we assume $\mathcal{O}$ and $\mathcal{B}$ are the subsets of pixels marked as "object" and "background" seeds. Additional soft constraints are defined to respect the boundaries of the objects to be segmented. A minimum cost cut generates a segmentation that is optimal given the aforementioned constraints.

The energy function is similar to the standard form used in Chapter 3 and is defined as:

$$E(y|x,\lambda) = \sum_{i \in \mathcal{V}} \underbrace{\psi(y_i|x_i)}_{\text{unary term}} + \lambda \sum_{(i,j) \in \mathcal{E}} \underbrace{\phi(y_i, y_j|x_i, x_j)}_{\text{pairwise term}} \ , \tag{6.1}$$

where $\mathcal{V}$ is the set of nodes corresponding to voxels, $\mathcal{E}$ is the set of edges and $y_i \in \{0, 1\}$ is a class label assigned to i corresponding to the object and the background classes. The so-called *unary* term ψ defined in Table 6.3 encourages agreement between a node's label y_i and the hard constraints provided by the pixels marked by the user.

The pairwise term $\phi(y_i, y_j|x_i, x_j)$ is defined in terms of the image gradient between the pixels i and j. As shown in Figure 6.7, this term is computed by filtering the

image with a derivative of Gaussian filter. Finally, the weight λ controls the relative importance of the two terms.

A new volume of rodent tissue was analyzed using the **Multi-layer** model presented in Section 6.2. A volume of size $385 \times 1000 \times 1000$ voxels ($3.8 \times 5 \times 5\ \mu m^3$) was fully annotated and used for training. A different volume of size $385 \times 754 \times 508$ voxels ($3.8 \times 3.8 \times 2.5\ \mu m^3$) was first segmented using the **Multi-layer** model and then post-processed by splitting merged mitochondria and manually cleaning the results. The volume and surface area of the mitochondria present in the axons and dendrites [1] were then computed and are reported in Figures 6.8 and 6.9. A statistically significant difference in the geometric properties of the mitochondria coming from the two structures is observed, and it is seen that the volume and surface area are much smaller for the mitochondria in the axons, which agrees with prior neuroscientific knowledge on this subject.

6.5 Conclusion

We have shown that coupling the learning method presented in Chapter 5 with the multi-layer model of [2] leads to better results on two different EM datasets. The quantitative results obtained after post-processing are of sufficiently high quality to perform quantitative analyses. The neuroscientists who conducted the analysis presented in Chapter 6.4 (Dr. Bohumil Maco and Dr. Graham Knott) are currently processing the data and further analysis on several datasets will be conducted in the near future.

[1] Axons and dendrites are two types of protoplasmic protrusions that extrude from the cell body of a neuron. Axons typically conduct electrical impulses away from the neuron's cell body while dendrites conduct the impulses received from other neural cells to the cell body.

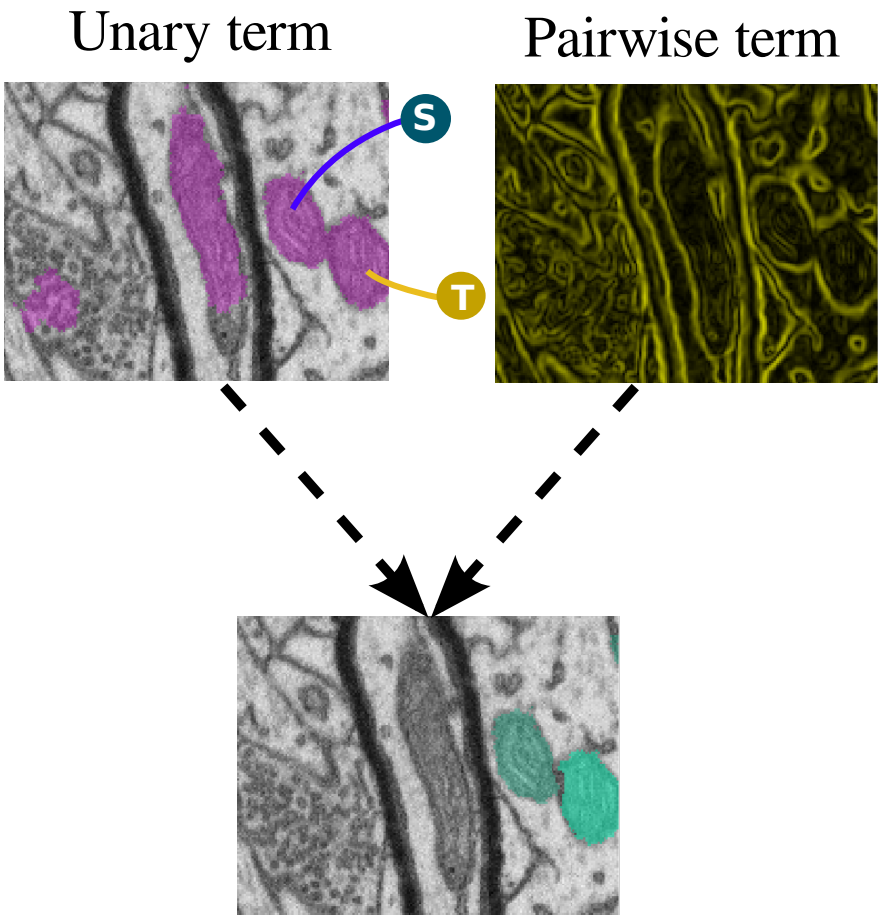


Figure 6.7: Splitting procedure: *Top-left* Seeds are manually provided by an expert user and used to compute the weights for the unary term. *Top-right* Edge weights are computed by using a derivative Gaussian filter on the original image. *Bottom* The result of the splitting procedure correctly separated the two mitochondria overlaid with different shades of green.

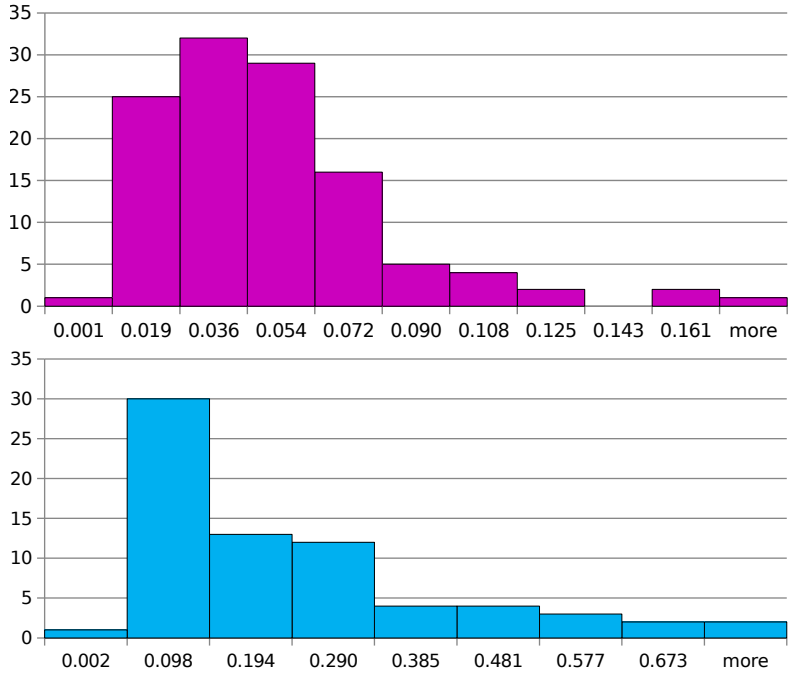


Figure 6.8: Volume in μm^3 for mitochondria in the axons (top) and dendrites (bottom).

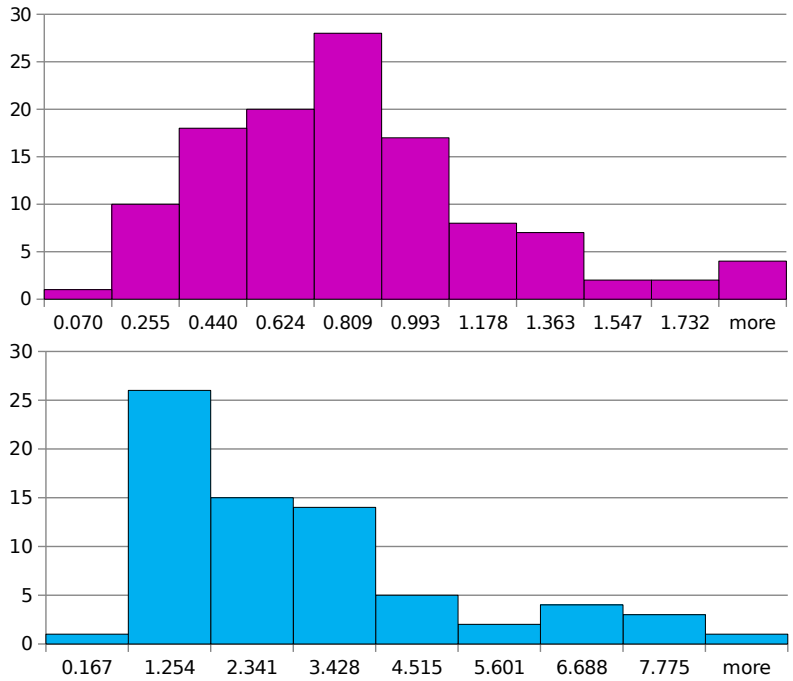


Figure 6.9: Surface area in μm^2 for mitochondria in the axons (top) and dendrites (bottom). We note that the two distributions are statistically different.

CHAPTER

SEVEN

CONCLUSION

We started this thesis with a description of the challenges posed by the image segmentation problem. We then described some of the dominant mathematical models to address these issues. The different methods we presented in this thesis are graph-partitioning-based approaches that rely on machine learning to learn optimal parameters. Chapter 2 reviewed notions related to graph-partitioning based approaches, such as MRFs and CRFs, and their application to image segmentation. In Chapter 3, we proposed a new algorithm to cluster groups of similar voxels into regularly spaced supervoxels. The use of supervoxels reduces the computational and memory costs by several orders of magnitude without sacrificing much accuracy because supervoxels naturally respect image boundaries. We also focused our attention on extracting discriminative features that can exploit the shape of the objects being segmented. In order to exploit the interactions between random variables associated to superpixels or supervoxels, we used a CRF model where the pairwise term was based on the prediction of a classifier trained to recognize which pairs of supervoxels are most likely to straddle a relevant boundary. A cross-validation method was used to estimate the parameters of the CRF model. We showed the effectiveness of our model for the segmentation of mitochondria on a dataset of electron microscopy images.

In Chapter 4, we investigated the maximum-margin approach to learn the parameters of a CRF model. This approach was limited to linear kernels, since more powerful non-linear kernels cause the learning to become prohibitively expensive. We introduced an approach to "kernelize" the features so that a linear structured SVM can leverage

the power of non-linear kernels without incurring the high computational cost. The resulting approach outperformed the first approach used in Chapter 3.

In Chapter 5, we proposed a working set-based approximate subgradient descent algorithm to minimize the margin-sensitive hinge loss arising from the soft constraints in max-margin learning frameworks, such as the structured SVM. We focused on the setting of general graphical models, such as loopy MRFs and CRFs commonly used in image segmentation, where exact inference is intractable and the most violated constraints can only be approximated, voiding the optimality guarantees of the structured SVM's cutting plane algorithm as well as reducing the robustness of existing subgradient based methods. We showed that the proposed method obtains better approximate subgradients through the use of working sets, leading to improved convergence properties and increased reliability. Furthermore, our method allowed new constraints to be randomly sampled, instead of computed using the more expensive approximate inference techniques such as belief propagation and graph cuts, which can be used to reduce learning time at only a small cost of performance. We demonstrated the strength of our method empirically on the segmentation of several electron microscopy datasets as well as the popular MSRC data set and showed state-of-the-art results.

In Chapter 6, we studied a new method combining the learning method presented in Chapter 5 with a multi-layer model encoding geometric interactions between the boundary and other regions such as interior/exterior. We showed that this new combination outperformed the previous methods presented in this thesis. More importantly, this method was successfully used to study statistical properties of mitochondria in an EM dataset and further analysis is currently underway.

Limitations and future work

The graph partitioning approaches discussed in Chapter 3, 4 and 5 rely on a pre-processing step to extract image regions called superpixels/supervoxels. The rest of the algorithm is then prone to errors made by this pre-processing step. A first attempt to address this issue was proposed by [51], where a joint image segmentation and labeling model was proposed in a probabilistic setting. However, it is still unknown if a similar model would perform well with the maximum-margin setting adopted in Chapter 4 and 5.

We have seen that the maximum-margin framework on which we relied in Chapter 4 to estimate the model parameters assumes that the loss-augmented prediction problem can be solved exactly. In such cases, the theoretical properties are well understood. The cutting-plane algorithm guarantees polynomial time termination and correctness (i.e. it returns a solution within the desired accuracy) and an empirical risk bound can be derived [136]. As explained in Chapter 5 the loss-augmented prediction problem can not generally be solved exactly for the kind of loopy graphs encountered in many computer vision problems, violating the optimality guarantees. Although approximations exist (see Section 5.2), it has been shown in [71] that the use of approximate inference during learning can often lead to surprisingly poor parameter estimates. The impact of approximate inference on structured learning is still largely misunderstood and is likely to be an active area of research for the future.

The work presented in this thesis only relies on pairwise CRFs. As discussed in the introduction, higher order terms have been shown to improve results for the problem of multi-class object segmentation [64] but they also lead to a higher computational complexity. The use of high order terms is likely to worsen the approximation of the most violated constraint during learning but a theoretical or practical understanding of this issue has yet to be derived.

The training method of Chapter 5 is fully supervised (all the variables were observed) but in the real world some variables might be unobserved even during training. Labeling a full EM stack is time consuming so a method that can deal with a partial ground-truth is highly desirable. One possible solution proposed by [155] is to use a maximum-margin method with maximization over the latent variables.

We have largely focused on the segmentation of mitochondria and we showed that the most advanced method presented in this thesis was mature enough to be used in real biological studies. The segmentation of other structures in EM datasets is also very challenging. In Chapter 4, we applied the kernelized features to the segmentation of synapses. Since the publication of this work, the authors of [9] showed that complex features considering the global context around synapses outperformed existing methods when applied to the segmentation of synapses in EM datasets. Being able to estimate the density of neurotransmitters is another very challenging problem that has been addressed by several authors [30, 80]. Finally, the reconstruction of neuronal processes is an active area of research that has already led to impressive large-scale automatic

reconstructions [60]. A thorough understanding of the brain will require modeling all these different structures, which will necessitate the collaboration of many researchers across different fields. No doubt the years to come will be extremely challenging and will lead to many very interesting discoveries in neuroscience.

CHAPTER

EIGHT

APPENDIX

8.1 Quadratic complexity of non-linear Structured SVM

For a set of training instances $(X_i, Y_i)_{i=1}^N$, from a sample space $\mathcal{X}$ and label space $\mathcal{Y}$, the structured SVM minimizes the following primal regularized risk function:

$$\begin{aligned} &\min_{\mathbf{w},\xi\geq\mathbf{0}} \frac{1}{2}\,||\mathbf{w}||_2^2 + \frac{C}{N}\sum_{n=1}^{N}\xi_n \\ &\text{s.t. } \forall n, Y \in \mathcal{Y}_n \backslash Y^{(n)} : \delta S_{\mathbf{w}}(Y) \geq \Delta(Y^{(n)}, Y) - \xi_n \end{aligned} \tag{8.1}$$

where $\mathcal{Y}_n$ is the set of all possible labelings for example n, ξ_n are the slack variables, and $\delta S_{\mathbf{w}}(Y)$ is shorthand for $S_{\mathbf{w}}(Y^{(n)}) - S_{\mathbf{w}}(Y)$.

The dual problem associated to the primal is :

$$\begin{aligned} \max_{\alpha} & \sum_{k,\hat{Y_k}\in S_k} \alpha_{k,\hat{Y_k}}\Delta(Y_k,\hat{Y_k}) - \\ & \frac{1}{2}\sum_{k,\hat{Y_k}\in S_k}\sum_{l,\hat{Y_l}\in S_l}\alpha_{k,\hat{Y_k}}\alpha_{l,\hat{Y_l}} < \Psi_k(Y_k) - \Psi_k(\hat{Y_k}), \Psi_l(Y_l) - \Psi_l(\hat{Y_l}) > \\ \text{s.t.} \quad & \forall k, \hat{Y_k} \in S_k, \quad \alpha_{k,\hat{Y_k}} \geq 0, \quad \alpha_{k,\hat{Y_k}} \leq C, \end{aligned} \tag{8.2}$$

where $S_k = \{Y \backslash Y_k\}$ is the set of all possible labels except the ground-truth.

The dot product between feature maps is defined as:

$$< \Psi_k(Y_k), \Psi_l(Y_l) > = \sum_{v_k} \sum_{v_l} \mathbf{1}[Y_k(v_k) = Y_l(v_l)] K(f(v_k), f(v_l)) + \sum_{c_k} \sum_{c_l} \mathbf{1}[Y_k(c_k) = Y_l(c_l)] K(f^E(c_k), f^E(c_l)),$$

where $f^E(x)$ is the feature vector for edge c.

The quadratic sum in Equation 8.2 shows that the non-linear formulation of Structured SVM has quadratic complexity.

8.2 Supplementary: Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets

We analyze the convergence properties of Algorithm 3. Recall that our goal is to find the parameter vector $\mathbf{w}^*$ that minimizes the empirical objective function:

$$\mathcal{L}(\mathbf{w}) = \sum_{n=1}^{N} l(Y^n, Y^*, \mathbf{w}) + \frac{1}{2C} ||\mathbf{w}||^2 . \tag{8.3}$$

At each iteration, Algorithm 3 chooses a random training example (X^n, Y^n) by picking an index $n \in \{1 \dots N\}$ uniformly at random. We then replace the objective given by Equation 8.3 with an approximation based on the training example (X^n, Y^n), yielding:

$$f(Y^n, Y^*, \mathbf{w}) = l(Y^n, Y^*, \mathbf{w}) + \frac{1}{2C} ||\mathbf{w}||^2 . \tag{8.4}$$

We consider the case where $l : \mathcal{W} \to \mathbb{R}$ is a convex loss function so that $f(\mathbf{w})$ is a λ-strongly convex function where $\lambda = \frac{1}{C}$.

Recall that the definition of an ϵ-subgradient of $f(\mathbf{w})$ is:

$$\forall \mathbf{w}' \in \mathcal{W}, \mathbf{g}^T(\mathbf{w} - \mathbf{w}') \geq f(\mathbf{w}) - f(\mathbf{w}') - \epsilon. \tag{8.5}$$

In the following, we will assume that the magnitude of the ϵ-subgradients we compute is bounded by a constant G, i.e. $||g||_2^2 \leq G^2$.

Let $\mathbf{w}^*$ be the minimizer of $\mathcal{L}(\mathbf{w})$. The following relation then holds trivially for $\mathbf{w}^*$:

$$\mathbf{g}^T(\mathbf{w} - \mathbf{w}^*) \geq f(\mathbf{w}) - f(\mathbf{w}^*) - \epsilon. \tag{8.6}$$

8.2.1 Convergence properties of the t^{th} parameter vector

This proof for subgradients was derived in [114] and we extend it to approximate subgradients here. We first present some inequalities that will be used in the following proof.

By the strong convexity of $f(\mathbf{w})$, we have:

$$\langle g^{(t)}, \mathbf{w}^{(t)} - \mathbf{w}^* \rangle \geq f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*) + \frac{\lambda}{2} \|\mathbf{w}^{(t)} - \mathbf{w}^*\|_2^2 - \epsilon. \tag{8.7}$$

Because $\mathbf{w}^*$ minimizes $f(\mathbf{w})$, $g(\mathbf{w}^*)$ and we have:

$$f(\mathbf{w}^{(t)}) - f(\mathbf{w}^*) \geq \frac{\lambda}{2}\|\mathbf{w}^{(t)} - \mathbf{w}^*\|_2^2. \tag{8.8}$$

By combining Equation 8.7 and 8.8 we get:

$$\langle g^{(t)}, \mathbf{w}^{(t)} - \mathbf{w}^* \rangle \geq \lambda\|\mathbf{w}^{(t)} - \mathbf{w}^*\|_2^2 - \epsilon. \tag{8.9}$$

In the following, we first start by bounding $\|\mathbf{w}^{(1)} - \mathbf{w}^*\|$ and then derive a bound for $\mathbb{E}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|$.

Lemma 1. *The error of* $\mathbf{w}^{(1)}$ *is:*

$$\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 \leq \frac{G^2 + 2\epsilon\lambda}{\lambda^2}. \tag{8.10}$$

Proof. From Equation 8.7, we deduce:

$$\begin{aligned} \langle g^{(1)}, \mathbf{w}^{(1)} - \mathbf{w}^* \rangle &\geq f(\mathbf{w}^{(1)}) - f(\mathbf{w}^*) + \frac{\lambda}{2}\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 - \epsilon \\ &\geq \frac{\lambda}{2}\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 - \epsilon \\ &\geq \lambda\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 - \epsilon, \end{aligned} \tag{8.11}$$

where the last inequality follows from the fact that $f(\mathbf{w}^{(1)}) - f(\mathbf{w}^*) \geq 0$.

Using the Cauchy-Schwarz inequality ($|\langle X, Y\rangle| \leq \|X\|\|Y\|$), we get:

$$\begin{aligned} \|g^{(1)}\|_2^2 &\geq \frac{\left(\lambda\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 - \epsilon\right)^2}{\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2} \\ &= \lambda^2\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 - 2\epsilon\lambda + \frac{\epsilon^2}{\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2}, \end{aligned} \tag{8.12}$$

and from the assumption that $\|g^{(t)}\|^2 \leq G^2$, we have that:

$$G^2 \geq \lambda^2\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 - 2\epsilon\lambda + \frac{\epsilon^2}{\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2}. \tag{8.13}$$

We then derive the following bound for $\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2$:

$$\|\mathbf{w}^{(1)} - \mathbf{w}^*\|_2^2 \leq \max\left(\frac{G^2 + 2\epsilon\lambda}{\lambda^2}, \frac{\epsilon^2}{G^2 + 2\epsilon\lambda}\right). \tag{8.14}$$

$$\begin{aligned}\frac{G^2+2\epsilon\lambda}{\lambda^2}-\frac{\epsilon^2}{G^2+2\epsilon\lambda} &= \frac{(G^2+2\epsilon\lambda)(G^2+2\epsilon\lambda)-\epsilon^2\lambda^2}{\lambda^2(G^2+2\epsilon\lambda)}=\frac{(G^2+2\epsilon\lambda)^2-\epsilon^2\lambda^2}{\lambda^2(G^2+2\epsilon\lambda)} \\ &= \frac{(G^2+2\epsilon\lambda+\epsilon\lambda)(G^2+2\epsilon\lambda-\epsilon\lambda)}{\lambda^2(G^2+2\epsilon\lambda)} \quad (8.15) \\ &= \frac{(G^2+3\epsilon\lambda)(G^2+\epsilon\lambda)}{\lambda^2(G^2+2\epsilon\lambda)}\geq 0. \quad (8.16)\end{aligned}$$

Therefore, we see that:

$$\max\left(\frac{G^2+2\epsilon\lambda}{\lambda^2},\frac{\epsilon^2}{G^2+2\epsilon\lambda}\right)=\frac{G^2+2\epsilon\lambda}{\lambda^2}. \quad (8.17)$$

We get Equation 8.10 by combining Equation 8.14 and 8.17 .

□

Theorem 1. *The error of* $\mathbf{w}^{(t+1)}$ *is:*

$$\mathbb{E}\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|_2^2\leq\frac{G^2}{\lambda^2 t}+\frac{\epsilon}{\lambda}. \quad (8.18)$$

Proof.

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|_2^2 &= \mathbb{E}\|\mathbf{w}^{(t)}-\eta^{(t)}\mathbf{g}^{(t)}-\mathbf{w}^*\|_2^2 \\ &= \mathbb{E}\|\mathbf{w}^{(t)}-\mathbf{w}^*\|_2^2-2\eta^{(t)}\mathbb{E}(\langle\mathbf{g}^{(t)},(\mathbf{w}^{(t)}-\mathbf{w}^*)\rangle)+(\eta^{(t)})^2(\mathbb{E}\|\mathbf{g}^{(t)}\|_2^2) \\ &\leq \mathbb{E}\|\mathbf{w}^{(t)}-\mathbf{w}^*\|_2^2-2\eta^{(t)}(\lambda\mathbb{E}\|\mathbf{w}^{(t)}-\mathbf{w}^*\|_2^2-\epsilon)+(\eta^{(t)})^2G^2 \\ &= (1-2\eta^{(t)}\lambda)\mathbb{E}\|\mathbf{w}^{(t)}-\mathbf{w}^*\|_2^2+(\eta^{(t)})^2G^2+2\eta^{(t)}\epsilon \quad (8.19)\end{aligned}$$

By applying the inequality recursively:

$$\begin{aligned}\mathbb{E}\|\mathbf{w}^{(t+1)}-\mathbf{w}^*\|_2^2 &\leq (1-2\eta^{(t)}\lambda)\mathbb{E}\|\mathbf{w}^{(t)}-\mathbf{w}^*\|_2^2+(\eta^{(t)})^2G^2+2\eta^{(t)}\epsilon \\ &\leq (1-2\eta^{(t)}\lambda)((1-2\eta^{(t-1)}\lambda)\mathbb{E}\|\mathbf{w}^{(t-1)}-\mathbf{w}^*\|_2^2+(\eta^{(t-1)})^2G^2+2\eta^{(t-1)}\epsilon) \\ &+ (\eta^{(t)})^2G^2+2\eta^{(t)}\epsilon \\ &\leq \left(\prod_{i=2}^{t}(1-2\eta^{(i)}\lambda)\right)(\mathbb{E}\|\mathbf{w}^{(2)}-\mathbf{w}^*\|_2^2)+\sum_{i=2}^{t}\prod_{j=i+1}^{t}(1-2\eta^{(j)}\lambda)(\eta^{(i)})^2G^2+ \\ &\quad \sum_{i=2}^{t}\prod_{j=i+1}^{t}(1-2\eta^{(j)}\lambda)2\eta^{(i)}\epsilon. \quad (8.20)\end{aligned}$$

Plugging in $\eta^{(i)} = \frac{1}{\lambda i}$, we get:

$$\begin{aligned}
\mathbb{E}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|_2^2 &\leq \prod_{i=2}^{t}\left(1-\frac{2}{i}\right)(\mathbb{E}\|\mathbf{w}^{(2)} - \mathbf{w}^*\|_2^2) + \sum_{i=2}^{t}\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right)\left(\frac{1}{i}\right)^2\frac{G^2}{\lambda^2} \\
&+ \sum_{i=2}^{t}\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right)\frac{2\epsilon}{i\lambda} \\
&= \frac{G^2}{\lambda^2}\sum_{i=2}^{t}\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right)\left(\frac{1}{i}\right)^2 + \sum_{i=2}^{t}\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right)\frac{2\epsilon}{i\lambda} \qquad (8.21)
\end{aligned}$$

Rakhlin [114] showed that setting $\eta^{(i)} = \frac{1}{\lambda i}$ gives us a $O(1/t)$ rate. Indeed, we have:

$$\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right) = \prod_{j=i+1}^{t}\left(\frac{j-2}{j}\right) = \frac{(i-1)i}{(t-1)t}, \qquad (8.22)$$

and therefore

$$\sum_{i=2}^{t}\frac{1}{i^2}\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right) = \sum_{i=2}^{t}\frac{(i-1)}{i(t-1)t} \leq \frac{1}{t}, \qquad (8.23)$$

$$\sum_{i=2}^{t}\prod_{j=i+1}^{t}\left(1-\frac{2}{j}\right)\frac{2\epsilon}{i\lambda} = \sum_{i=2}^{t}\frac{2(i-1)i\epsilon}{i(t-1)t\lambda} = \frac{2\epsilon}{(t-1)t\lambda}\sum_{i=1}^{t-1} i = \frac{2\epsilon}{(t-1)t\lambda}\left(\frac{(t-1)t}{2}\right) = \frac{\epsilon}{\lambda} \qquad (8.24)$$

By combining Equation 8.21 with Equation 8.23 and Equation 8.24, we then get:

$$\mathbb{E}\|\mathbf{w}^{(t+1)} - \mathbf{w}^*\|_2^2 \leq \frac{G^2}{\lambda^2 t} + \frac{\epsilon}{\lambda}. \qquad (8.25)$$

We can deduce that the conditions of convergence are the same as the ones for subgradient descent (i.e. for $\epsilon = 0$) :

$$\begin{aligned}
\lim_{T\to+\infty}\sum_{i=1}^{T}\eta^{(i)} &\to \infty \\
\lim_{T\to+\infty}\sum_{i=1}^{T}(\eta^{(i)})^2 &< \infty \qquad (8.26)
\end{aligned}$$

As long as the choice of the step size satisfies Equation 8.26, we can see that the first term on the right side of Equation 8.25 goes to 0 so stochastic ϵ-subgradient descent will convergence to a distance ϵ away from the optimal value.

□

BIBLIOGRAPHY

[1] A. Ali, A. Farag, and A. El-Baz. Graph Cuts Framework for Kidney Segmentation with Prior Shape Constraints. In *MICCAI*, pages 384–92, 2007. 5, 54

[2] A. Delong and Y. Boykov. Globally Optimal Segmentation of Multi-Region Objects. In *ICCV*, pages 285–292, 2009. 7, 44, 52, 56, 107, 108, 116

[3] A. Levin and Y. Weiss. Learning to Combine Bottom-Up and Top-Down Segmentation. In *ECCV*, pages 581–94, 2006. 5, 54

[4] R. Achanta. *Finding Objects of Interest in Images Using Saliency and Superpixels*. PhD thesis, EPFL, 2010. 48

[5] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Suesstrunk. SLIC Superpixels Compared to State-Of-The-Art Superpixel Methods. *PAMI*, 34(11):2274–2281, 2012. 13, 47, 73, 83, 84, 85, 98, 102, 104, 109

[6] M. Aliaksei and A. Severyn. Fast Support Vector Machines for Structural Kernels. In *ECML*, 2011. 72, 73

[7] B. Andres, U. Koethe, M. Helmstaedter, W. Denk, and F. Hamprecht. Segmentation of Sbfsem Volume Data of Neural Tissue by Hierarchical Classification. In *DAGM*, pages 142–52, 2008. 43, 48

[8] M.-F. Balcan, A. Blum, and S. Vempala. Kernels as Features: On Kernels, Margins, and Low-Dimensional Mappings. *ML*, 65(1):79–94, 2006. 72

[9] C. Becker, K. Ali, G. Knott, and P. Fua. Learning Context Cues for Synapse Segmentation in EM Volumes. In *MICCAI*, September 2012. 121

[10] S. Belongie, J. Malik, and J. Puzicha. Shape Matching and Object Recognition Using Shape Contexts. *PAMI*, 24(24):509–522, April 2002. 4

[11] L. Bertelli, T. Yu, D. Vu, and B. Gokturk. Kernelized Structural SVM Learning for Supervised Object Segmentation. In *CVPR*, 2011. 73

[12] J. Besag. Spatial Interaction and the Statistical Analysis of Lattice Systems. *J. Royal Stat. Soc., B*, 36(2), 1974. 18, 37, 92

[13] J. Besag. Statistical Analysis of Non-Lattice Data. *The Statistician*, 24(3):179–195, 1975. 10

[14] Y. Boykov and M. Jolly. Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D Images. In *ICCV*, pages 105–12, 2001. 7, 44, 52, 55, 64, 66, 89, 115

[15] Y. Boykov and V. Kolmogorov. An Experimental Comparison of Min-Cut/max-Flow Algorithms for Energy Minimization in Vision. *PAMI*, 26(9):1124–1137, 2004. 62, 63

[16] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *PAMI*, 23(11), 2001. 74

[17] C. R. Brice and C. L. Fennema. Scene Analysis Using Regions. *Artificial Intelligence*, 1:205–226, 1970. 1, 3

[18] C. Rother, V. Kolmogorov, and A. Blake. "GrabCut" - Interactive Foreground Extraction Using Iterated Graph Cuts. In *SIGGRAPH*, pages 309–314, 2004. 7, 44, 52

[19] N. Campbell, B. Williamson, and R. Heyden. *Biology: Exploring Life*. Pearson Prentice Hall, 2006. 40

[20] S. Campello and L. Scorrano. Mitochondrial Shape Changes: Orchestrating Cell Pathophysiology. *EMBO Reports*, 11(9):678–84, 2010. 39

[21] J. Carreira and C. Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *CVPR*, pages 3241–48, 2010. 7, 44

[22] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image Segmentation Using Expectation-Maximization and Its Application to Image Querying. *PAMI*, 24(8):1026–1038, 2002. 4

[23] O. Daněk, P. Matula, C. O. de Solórzano, A. Muñoz-Barrutia, M. Maška, and M. Kozubek. Segmentation of touching cell nuclei using a two-stage graph cut model. In *Image Analysis*, pages 410–419, 2009. 115

[24] G. Díaz, F. Gonzalez, and E. Romero. Automatic clump splitting for cell quantification in microscopical images. In *Progress in Pattern Recognition, Image Analysis and Applications*, pages 763–772, 2007. 115

[25] U. Ernst, S. Mandon, N. Schinkel-Bielefeld, S. Neitzel, A. Kreiter, and K. Pawelzik. Optimality of human contour integration. *PLoS computational biology*, 8(5):e1002520, 2012. 5

[26] M. Everingham, C. W. L. Van Gool and, J. Winn, and A. Zisserman. The Pascal Visual Object Classes Challenge 2010 (VOC2010) Results. vii, 7, 11, 40, 44, 57, 61, 66, 104

[27] H. G. Feichtinger. *Gabor analysis and algorithms: Theory and applications*. Birkhauser Boston, 1998. 3

[28] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *IJCV*, 59(2):167–181, 2004. 7, 43, 48

[29] P. Felzenszwalb, D. Mcallester, and D. Ramanan. A Discriminatively Trained, Multiscale, Deformable Part Model. In *CVPR*, June 2008. 86

[30] L. Fiaschi, U. Koethe, R. Nair, and F. Hamprecht. Learning to count with regression forest and structured labels. In *ICPR*, pages 2685–2688, 2012. 121

[31] T. Finley and T. Joachims. Training Structural SVMs When Exact Inference is Intractable. In *ICML*, 2008. 76, 91

[32] D. Freedman and T. Zhang. Interactive Graph-Cut Based Segmentation with Shape Priors. In *CVPR*, pages 755–62, 2005. 5, 54

[33] W. Freeman and E. Adelson. The Design and Use of Steerable Filters. *PAMI*, 13:891–906, 1991. 3

[34] B. Fulkerson, A. Vedaldi, and S. Soatto. Class Segmentation and Object Localization with Superpixel Neighborhoods. In *ICCV*, 2009. 13

[35] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *PAMI*, 6(6):721–741, 1984. 21

[36] J. Gemert, J. Geusebroek, C. Veenman, and A. Smeulders. Kernel Codebooks for Scene Categorization. In *ECCV*, pages 696–709, 2008. 81

[37] J. Gonfaus, X. Boix, J. Weijer, A. Bagdanov, J. Serrat, and J. Gonzalez. Harmony Potentials for Joint Classification and Segmentation. In *CVPR*, pages 3280–87, 2010. 7, 8, 44, 78, 81, 82, 83

[38] S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-Class Segmentation with Relative Location Prior. *IJCV*, 80(3):300–316, 2008. 13

[39] D. Greig, B. Porteous, and A. Seheult. Exact Maximum a Posteriori Estimation for Binary Images. *Journal of the Royal Statistical Society*, 51:271–279, 1989. 53

[40] C. Grigorescu and N. Petkov. Distance Sets for Shape Filters and Shape Recognition. *PAMI*, 12(10):1274–1286, 2003. 49

[41] S. Gross, O. Russakovsky, C. Do, and S. Batzoglou. Training conditional random fields for maximum labelwise accuracy. In *NIPS*, page 529, 2007. 22

[42] T. Hazan and R. Urtasun. A Primal-Dual Message-Passing Algorithm for Approximated Large Scale Structured Prediction. In *NIPS*, 2010. 36

[43] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale Conditional Random Fields for Image Labeling. In *CVPR*, pages 695–702, 2004. 36

[44] R. F. Hess, W. Mcllhagga, and D. J. Field. Contour Integration in Strabismic Amblyopia: the Sufficiency of an Explanation Based on Positional Uncertainty. *Vision Research*, pages 3145–3161, 1997. 5

[45] G. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14:1771–1800, 2002. 36

[46] G. Hjaltason and H. Samet. Properties of Embedding Methods for Similarity Searching in Metric Spaces. *PAMI*, 25(5):530–49, 2003. 49

[47] W. Hoeffding. Probability Inequalities for Sums of Bounded Random Variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963. 96

[48] T. Hofmann, B. Schlkopf, and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008. 9

[49] A. Hyvrinen. Consistency of Pseudolikelihood Estimation of Fully Visible Boltzmann Machines. *Neural Comp*, 18:2283–2292, 2006. 36

[50] L. Ibanez, W. Schroeder, L. Ng, and J. Cates. *The Itk Software Guide*. 49, 57

[51] A. Ion, J. Carreira, and C. Sminchisescu. Probabilistic joint image segmentation and labeling. In *NIPS*, 2011. 120

[52] V. Jain, B. Bollmann, M. Richardson, D. Berger, M. Helmstaedter, K. Briggman, W. Denk, J. Mendenhall, W. Abraham, K. harris, N. Kasthuri, K. Hayworth, R. Schalek, J. Tapia, J. Lichtman, and H. Seung. Boundary Learning by Optimization with Topological Constraints. In *CVPR*, pages 2488–95, 2010. 42, 61

[53] V. Jain, J. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung. Supervised Learning of Image Restoration with Convolutional Networks. In *ICCV*, pages 1–8, 2007. 42

[54] V. Jain, H. S. Seung, and S. Turaga. Machines That Learn to Segment Images: A Crucial Technology for Connectomics. *Current Opinion in Neurobiology*, 20:1–14, 2010. 42

[55] J.M., Hammersley, and P. Clifford. Markov Field on Finite Graphs and Lattices. Technical report, 1971. 18

[56] T. Joachims, T. Finley, and C.-N. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009. 91

[57] E. Jurrus, A. Paiva, S. Watanabe, J. Anderson, R. Whitaker, B. Jones, R. Marc, and T. Tasdizen. Detection of Neuron Membranes in Electron Microscopy Images Using a Serial Neural Network Architecture. *MIA*, 14(6):770–783, 2010. 43

[58] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *IJCV*, 1(4):321–331, 1988. 6

[59] V. Kaynig, T. Fuchs, and J. Buhmann. Neuron Geometry Extraction by Perceptual Grouping in Sstem Images. In *CVPR*, pages 2902–09, 2010. 7, 11, 40, 43, 44

[60] V. Kaynig, A. Vazquez-Reina, S. Knowles-Barley, M. Roberts, T. Jones, N. Kasthuri, E. Miller, J. Lichtman, and H. Pfister. Large-Scale Automatic Reconstruction of Neuronal Processes from Electron Microscopy Images. Technical report, ArXiv, 2013. 122

[61] R. Kindermann and J. Snell. *Markov Random Fields and Their Applications*. AMS, 1980. 18

[62] A. Knott, G. Perkins, R. Schwarzenbacher, and E. Bossy-Wetzel. Mitochondrial Fragmentation in Neurodegeneration. *Nature Reviews. Neuroscience*, 9(7):505–18, 2008. 39

[63] G. Knott, H. Marchman, D. Wall, and B. Lich. Serial Section Scanning Electron Microscopy of Adult Brain Tissue Using Focused Ion Beam Milling. *Journal of Neuroscience*, 28(12):2959–64, 2008. 40

[64] P. Kohli, L. Ladicky, and P. Torr. Robust Higher Order Potentials for Enforcing Label Consistency. In *CVPR*, 2008. 7, 121

[65] V. Kolmogorov and Y. Boykov. What Metrics Can Be Approximated by Geo-Cuts, or Global Optimization of Length/area and Flux. In *ICCV*, pages 564–71, 2005. 7, 44

[66] V. Kolmogorov and C. Rother. Minimizing Nonsubmodular Functions with Graph Cuts-A Review. *PAMI*, 29(7):1274–1279, 2007. 47

[67] V. Kolmogorov and R. Zabih. What Energy Functions Can Be Minimized via Graph Cuts? *PAMI*, 26(2):147–159, 2004. 53, 74, 98

[68] N. Komodakis, G. Tziritas, and N. Paragios. Fast, Approximately Optimal Solutions for Single and Dynamic MRFs. In *CVPR*, pages 1–8, 2007. 47

[69] P. Krähenbühl and V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *NIPS*, 2011. 8, 102

[70] A. Kreshuk, C. N. Straehle, C. Sommer, U. Koethe, G. Knott, and F. Hamprecht. Automated Segmentation of Synapses in 3D Em Data. In *ISBI*, pages 220–223, 2011. 42

[71] A. Kulesza and F. Pereira. Structured learning with approximate inference. In *NIPS*, 2007. 121

[72] M. Kumar, P. Torr, and A. Zisserman. Obj Cut. In *CVPR*, pages 18–25, 2005. 5, 54

[73] R. Kumar, A. Vazquez-Reina, and H. Pfister. Radon-Like Features and Their Application to Connectomics. In *Workshop on MMBIA*, 2010. 42

[74] L. Ladicky, C. Russell, P. Kohli, and P. Torr. Associative Hierarchical CRFs for Object Class Image Segmentation. In *ICCV*, 2009. 81, 82, 83, 102

[75] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. Torr. What, Where and How Many? Combining Object Detectors and CRFs. In *ECCV*, pages 424–437, 2010. 101, 102

[76] L. Ladicky and P. Torr. Locally Linear Support Vector Machines. In *ICML*, 2011. 72

[77] J. Lafferty, A. Mccallum, and F. Pereira. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*, 2001. 18, 23, 71, 92

[78] D. Lee, K. Lee, W. Ho, and S. Lee. Target Cell-Specific Involvement of Presynaptic Mitochondria in Post-Tetanic Potentiation at Hippocampal Mossy Fiber Synapses. *The Journal of NeuroScience*, 27(50):13603–13, 2007. 39

[79] V. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *ECCV*, 2008. 54

[80] V. Lempitsky and A. Zisserman. Learning to Count Objects in Images. In *NIPS*, 2010. 121

[81] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. Turbopixels: Fast Superpixels Using Geometric Flows. *PAMI*, 31(12):2290–97, 2009. 48

[82] K. Li, X. Wu, D. Chen, and M. Sonka. Optimal surface segmentation in volumetric images-a graph-theoretic approach. *PAMI*, 28(1):119–134, 2006. 108, 109

[83] S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, 1995. 21

[84] Y. Li, J. Sun, and H.-Y. Shum. Video object cut and paste. *Transactions on Graphics*, 24(3):595–600, 2005. 48

[85] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy Snapping. *SIGGRAPH*, 23(3):303–308, 2004. 13

[86] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 20(2):91–110, 2004. 3

[87] A. Lucchi, Y. Li, X. Boix, K. Smith, and P. Fua. Are Spatial and Global Constraints Really Necessary for Segmentation? In *ICCV*, 2011. 4, 72, 77, 78, 81, 82, 83

[88] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua. Supervoxel-Based Segmentation of Mitochondria in EM Image Stacks with Learned Shape Features. *TMI*, 31(2):474–486, 2011. 81, 85, 86, 102, 104, 109

[89] A. Lucchi, K. Smith, A. Radhakrishna, V. Lepetit, and P. Fua. A Fully Automated Approach to Segmentation of Irregularly Shaped Cellular Stuctures in EM Images. In *MICCAI*, pages 463–71, September 2010. 42

[90] T. Mäenpää. *The Local Binary Pattern Approach to Texture Analysis- Extensions and Applications*. University of Oulu, Oulu Finland, 2003. 52

[91] S. Maji and A. Berg. Max-Margin Additive Classifiers for Detection. In *ICCV*, 2009. 72, 79

[92] T. Malisiewicz and A. Efros. Improving Spatial Support for Object via Multiple Segmentations. In *BMVC*, 2007. 99

[93] B. Manjunath, T. Simchony, and R. Chellappa. Stochastic and Deterministic Networks for Texture Segmentation. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(6):1039–1049, 1990. 22

[94] C. Mannella, M. Marko, and K. Buttle. Reconsidering Mitochondrial Structure: New Views of an Old Organelle. *Trends Biochem. Sci.*, 15:37–38, 1997. 40

[95] B. Marsh, D. Mastronarde, K. Buttle, K. Howell, and J. McIntosh. Organellar Relationships in the Golgi Region of the Pancreateic Beta Cell Line, Hit-T15, Visualized by High Resolution Electron Tomography. *PNAS*, 98:2399–2406, 2001. 11, 40

[96] D. Mcallester, T. Hazan, and J. Keshet. Direct Loss Minimization for Structured Prediction. In *NIPS*, 2010. 89, 92, 94

[97] G. Mori, X. Ren, A. Efros, and J. Malik. Recovering Human Body Configurations: Combining Segmentation and Recognition. In *CVPR*, 2004. 13

[98] E. Mortensen and W. Barrett. Interactive Segmentation with Intelligent Scissors. *Graphical Models and Image Processing*, 60:349–384, 1998. 56

[99] K. Murphy, Y. Weiss, and M. Jordan. Loopy Belief Propagation for Approximate Inference: An Empirical Study. In *UAI*, 1999. 74, 89

[100] N. Vu and B. Manjunath. Graph Cut Segmentation of Neuronal Structures from Transmission Electron Micrographs. In *ICIP*, pages 725–728, 2008. 5, 7, 11, 40, 44, 54

[101] R. Narasimha, H. Ouyang, A. Gray, S. McLaughlin, and S. Subramaniam. Automatic Joint Classification and Segmentation of Whole Cell 3D Images. *PR*, 42:1067–1079, 2009. 42, 65

[102] Y. Nesterov. Primal-Dual Subgradient Methods for Convex Problems. *Math. Program.*, 120(1):221–259, April 2009. 89, 95, 99

[103] H. Nguyen and Q. Ji. Shape-Driven Three-Dimensional Watersnake Segmentation of Biological Membranes in Electron Tomography. *TMI*, 27(5):616–628, 2008. 42

[104] S. Nowozin, P. Gehler, and C. Lampert. On Parameter Learning in CRF-Based Approaches to Object Class Image Segmentation. In *ECCV*, 2010. 72

[105] A. Oliva and A. Torralba. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *IJCV*, 42(3):145–175, 2001. 4

[106] S. Osher and N. Paragios. *Geometric Level Set Methods in Imaging, Vision, and Graphics*. Springer, 2003. 5, 6

[107] D. Padfield, J. Rittscher, N. Thomas, and B. Roysam. Spatio-Temporal Cell Cycle Phase Analysis Using Level Sets and Fast Marching Methods. *MIA*, 13(1):143–155, 2009. 6, 43

[108] T. Pavlidis. *Structural Pattern Recognition*. Springer-Verlag, 1977. 1

[109] H. Peng, Z. Ruan, F. Long, J. Simpson, and E. Myers. V3D Enables Real-Time 3d Visualization and Quantitative Analysis of Large-Scale Biological Image Data Sets. *Nature Biotechnology*, 28(4):348–353, 2010. 59

[110] J. Platt. *Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines*. MIT Press, 1998. 91

[111] A. Poole, R. Thomas, L. Andrews, H. Mcbride, A. Whitworth, and L. Pallanck. The Pink1/parkin Pathway Regulates Mitochondrial Morphology. *Proceedings of the National Academy of Sciences of the United States of America*, 105(5):1638–43, 2008. 39

[112] M. Prasad, A. Zisserman, A. Fitzgibbon, M. Kumar, and P. Torr. Learning Class-Specific Edges for Object Detection and Segmentation. In *ICVGIP*, pages 94–105, 2006. 56

[113] A. Rahimi and B. Recht. Random Features for Large-Scale Kernel Machines. In *NIPS*, 2007. 72, 79

[114] A. Rakhlin, O. Shamir, and K. Sridharan. Making gradient descent optimal for strongly convex stochastic optimization. Technical report, ArXiv, 2012. 125, 128

[115] W. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971. 42

[116] N. Ratliff, J. A. Bagnell, and M. Zinkevich. (Online) Subgradient Methods for Structured Prediction. In *AISTATS*, 2007. 89, 91, 94, 96, 99, 100, 101, 102, 103, 104, 105

[117] S. M. Robinson. Linear convergence of epsilon-subgradient descent methods for a class of convex functions. *Mathematical Programming*, 86:41–50, 1999. 96

[118] C. Russell, P. Kohli, and P. Torr. Exact and approximate inference in associative hierarchical networks using graph cuts. In *UAI*, 2012. 7

[119] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. Pegasos: Primal Estimated Sub-Gradient Solver for SVM. *Math. Program.*, 127(1):3–30, March 2011. 91

[120] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *PAMI*, 22(8):888–905, 2000. 4, 7, 43

[121] N. Shor, K. Kiwiel, and A. Ruszcayǹski. *Minimization Methods for Non-Differentiable Functions*. Springer-Verlag, 1985. 96

[122] J. Shotton, M. Johnson, and P. Cipolla. Semantic Texton Forests for Image Categorization and Segmentation. In *CVPR*, 2008. 81, 82, 83

[123] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context. *IJCV*, 81(1), January 2009. 7, 71, 77, 80, 83, 99, 102

[124] K. Smith, A. Carleton, and V. Lepetit. Fast Ray Features for Learning Irregular Shapes. In *ICCV*, pages 397–404, 2009. 42, 49

[125] C. Sommer, C. Straehle, U. Koethe, and F. Hamprecht. Interactive Learning and Segmentation Tool Kit. In *Systems Biology of Human Disease*, pages 230–33, 2010. 41, 42, 60, 61, 65

[126] C. Stretcha. Multi-View Stereo as an Inverse Inference Problem, 2007. 28

[127] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric Belief Propagation. In *CVPR*, 2003. 32

[128] J. Suri, S. Singh, S. Laxminarayan, X. Zeng, K. Liu, and L. Reden. Shape Recovery Algorithms Using Level Sets in 2D/3D Medical Imagery: A State-Of-The-Art Review. *TITB*, 6(1), 2002. 6

[129] C. Sutton and A. McCallum. Introduction to conditional random fields for relational learning. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2006. 92

[130] C. Sutton and A. McCallum. Piecewise Pseudolikelihood for Efficient Training of Conditional Random Fields. In *ICML*, 2007. 36

[131] M. Szummer, P. Kohli, and D. Hoiem. Learning CRFs Using Graph Cuts. In *ECCV*, 2008. 72, 76, 91, 109

[132] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning Structured Prediction Models: A Large Margin Approach. In *ICML*, 2005. 91

[133] B. Taskar, C. Guestrin, and D. Koller. Max-Margin Markov Networks. In *NIPS*, 2003. 37, 72, 90, 93

[134] B. Taskar, S. Lacoste-julien, and M. Jordan. Structured Prediction, Dual Extragradient and Bregman Projections. *JMLR*, 7:1627–1653, 2006. 91

[135] E. Tola, V. Lepetit, and P. Fua. Daisy: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *PAMI*, 32(5):815–830, 2010. 52

[136] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *ICML*, 2004. 10, 37, 38, 70, 72, 74, 76, 81, 89, 90, 91, 93, 99, 101, 102, 121

[137] S. Turaga, J. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. Seung. Convolutional Networks Can Learn to Generate Affinity Graphs for Image Segmentation. *Neural Computation*, 22:511–538, 2010. 42

[138] T. Tuytelaars and K. Mikolajczyk. Local Invariant Feature Detectors: A Survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, July 2008. 4

[139] A. Vazquez-Reina, M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister. Segmentation Fusion for Connectomics. In *ICCV*, 2011. 42

[140] A. Vedaldi and S. Soatto. Quick Shift and Kernel Methods for Mode Seeking. In *ECCV*, pages 705–18, 2008. 48

[141] A. Vedaldi and A. Zisserman. Efficient Additive Kernels via Explicit Feature Maps. *PAMI*, 2011. 72, 79

[142] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and Supervoxels in an Energy Optimization Framework. In *ECCV*, pages 211–224, 2010. 48

[143] K. Venkataraju, A. Paiva, E. Jurrus, and T. Tasdizen. Automatic Markup of Neural Cell Membranes Using Boosted Decision Stumps. In *IEEE Symposium on Biomedical Imaging: From Nano to Macro*, pages 1039–42, 2009. 43

[144] L. Vincent and P. Soille. Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations. *PAMI*, 13(6):583–598, 1991. 48

[145] S. V. N. Vishwanathan, N. Schraudolph, M. Schmidt, and K. Murphy. Accelerated Training of Conditional Random Fields with Stochastic Gradient Methods. In *ICML*, 2006. 36

[146] S. Vitaladevuni, Y. Mishchenko, A. Genkin, D. Chklovskii, and K. Harris. Mitochondria Detection in Electron Microscopy Images. In *Workshop on Microscopic Image Analysis with Applications in Biology*, 2008. 42

[147] M. Volkovs, H. Larochelle, and R. Zemel. Loss-Sensitive Training of Probabilistic Conditional Random Fields. *CORR*, abs/1107.1805, 2011. 36

[148] M. Wick, K. Rohanimanesh, K. Bellare, A. Culotta, and A. Mccallum. Samplerank: Training Factor Graphs with Atomic Gradients. In *ICML*, 2011. 36, 92, 98, 99, 102, 105

[149] L. Xiao. Dual averaging methods for regularized stochastic learning and online optimization. *JLMR*, 11:2543–2596, Dec. 2010. 89, 95, 99

[150] J. Yang, J. Wang, and H. Lu. People Detection by Boosting Features in Nonlinear Subspace. In *Advances in Multimedia Information Processing - PCM 2010*, 2010. 13

[151] M. Yang, K. Kpalma, and J. Ronsin. A Survey of Shape Feature Extraction Techniques. *PR*, 2008(November):43–90, 2008. 4

[152] J. Yao, S. Fidler, and R. Urtasun. Describing the Scene as a Whole: Joint Object Detection, Scene Classification and Semantic Segmentation. In *CVPR*, pages 702–709, 2012. 101, 102

[153] J. Yedidia, W. Freeman, and Y. Weiss. *Understanding Belief Propagation and Its Generalizations*, pages 239–269. Morgan Kaufmann, 2003. 30, 47

[154] C.-N. Yu and T. Joachims. Training Structural SVMs with Kernels Using Sampled Cuts. In *KDD*, 2008. 72

[155] C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, pages 1169–1176, 2009. 121

[156] C. Zitnick and S. Kang. Stereo for Image-Based Rendering Using Image Over-Segmentation. *IJCV*, 75(1):49–65, October 2007. 13

LUCCHI Aurélien

Avenue de Florissant, 22 · 1020, Renens, Switzerland · aurelien.lucchi@gmail.com · http ://cvlabwww.epfl.ch/∼lucchi/
Single · 28 years old · French Nationality · English (fluent) · Spanish (basic)

Work Experience

August 2008 - present
Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland
Research Assistant – PhD in Computer Science

- Advised by Professor Pascal Fua at EPFL, a top-ranked University in engineering and computer science.
- Thesis : *Learning Discriminative Features and Structured Models for Segmentation in Microscopy and Natural Images.*
- Detected and tracked pedestrians for an augmented reality project.

July 2012 - September 2012
Microsoft, Cambridge, UK
Research Intern

- Worked with Dr. Darko Zikic and Dr. Antonio Criminisi in the Machine Learning and Perception group at Microsoft Research Cambridge.
- Developed machine learning algorithms for the segmentation of brain tumors in MRI images.

June 2011 - September 2011
Google, New York City
Research Intern

- Worked with Dr. Jason Weston on a joint framework for image and word sense discrimination for Google Image Search.
- Developed machine learning algorithms for Google Image Search resulting in an ECCV (European Conference on Computer Vision) publication.

October 2007 - August 2008
Schlumberger Limited, Stonehouse, UK
Software Embedded Engineer

- Schlumberger's PowerDrive is a rotary steerable system for directional drilling.
- Developed software for an ARM-based platform with FPGAs and scientific instrumentation such as accelerometers and magnetometers. Programmed in C++ using a real-time operating system, ThreadX.

Summer 2007
6 months
WesternGeco - Schlumberger, Houston (TX), USA
Software Engineer

- Developed user paradigms for large scale seismic data visualization. Extended Volume Rendering algorithms using Open Inventor and GLSL.

Summer 2006
5 months
WesternGeco - Schlumberger, Gatwick, UK
Software Engineer

- Implemented a file transfer tool for large amounts of seismic data.

Summer 2005
4 months
Phoenix Interactive, Lyon, France
Video Game Programmer

- Assisted in developing *"Caméra café 2"*, a French video game released in 2005, working with the Torque Game Engine.

Teaching

Spring 2010-2012 — Teaching assistant for Master's level course, *Introduction to Computer Vision.*
Fall 2010 — Teaching assistant for Bachelor level course, *Algorithms.*

Education

2008 – 2013 — **PhD in Computer Science**
Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland

- Expected graduation date : August 2013.
- Courses : *Algorithms, Computer vision, Digital 3D Geometry.*

2006 – 2007 — **Exchange Student**
Fall semester — **Dublin City University, Dublin, Ireland**

- Courses : *Image Processing, Speech Processing, Cryptography, Computer graphics, Artificial Intelligence, Multimedia Information Retrieval, Astronomy.*

2004 – 2007 — **MSc in Computer Science**
INSA (National Institute of Applied Sciences), Lyon, France

- INSA is a top-ranked French engineering university.
- Courses : *General Training, Computer Hardware, System Software, Software Engineering, Information Systems, Mathematical tools, Modeling and Problem Solving, Industrial Informatics, Networks and Telecommunications.*

2002 – 2004 — **Bachelor in Computing**
University of Burgundy, France

- Highest class average.

2002 — **Secondary School Diploma in Science**
Dijon, France

- Graduated with distinction.

Computer Skills

Languages — C, C++, C#, Matlab, Python, SQL
Web — HTML, PHP
Design methods — UML, USDP (object oriented design)
Libraries — ITK, OpenCV

Patents

Efficient Scanning for EM Based Target Localization, 2012. Application No. 13/535,497, US Patent Pending.

Publications

1. Flash Scanning for Electron Microscopy. R. Sznitman, A. Lucchi, M. Cantoni, G. Knott and P. Fua, *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2013..*

2. Learning for Structured Prediction Using Approximate Subgradient Descent with Working Sets. A. Lucchi, Y. Li and P. Fua, *International Conference on Computer Vision and Pattern Recognition (CVPR), 2013.*

3. An Optimal Policy for Target Localization with Application to Electron Microscopy. R. Sznitman, A. Lucchi, P. Frazier, B. Jedynak and P. Fua, *International Conference on Machine Leaning (ICML), 2013.*

4. Structured Image Segmentation using Kernelized Features. A. Lucchi, Y. Li, K.Smith and P. Fua, *European Conference on Computer Vision (ECCV), 2012.*

5. Joint Image and Word Sense Discrimination For Image Retrieval. A. Lucchi and J. Weston, *European Conference on Computer Vision (ECCV), 2012.*

6. Efficient Scanning for EM Based Target Localization. R. Sznitman, A. Lucchi, N. Pjescic-Emedji, G. Knott and P. Fua, *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2012..*

7. Are Spatial and Global Constraints Really Necessary for Segmentation ?, A. Lucchi, Y. Li, X. Boix, K. Smith and P. Fua, *IEEE International Conference on Computer Vision (ICCV), 2011.*

8. Supervoxel-Based Segmentation of EM Image Stacks with Learned Shape Features, A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua, *IEEE Transactions on Medical Image Processing (TMI), 2011.*

9. SLIC Superpixels Compared to State-of-the-art Superpixel Methods, R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 2012.*

10. A Fully Automated Approach to Segmentation of Irregularly Shaped Cellular Structures in EM Images, A. Lucchi, K. Smith, R. Achanta, V. Lepetit, and P. Fua, *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI), 2010.*

11. An empirical evaluation of touch and tangible interfaces for tabletop displays, A. Lucchi, P. Jermann, G. Zufferey, and P. Dillenbourg. *International conference on tangible, embedded, and embodied interaction (TEI), 2010.*

12. TinkerSheets : Using Paper Forms to Control and Visualize Tangible Simulations, G. Zufferey, P. Jermann, A. Lucchi, and P. Dillenbourg. *International conference on tangible, embedded, and embodied interaction (TEI), 2009.*