THE UNIVERSITY OF NOTTINGHAM

Hierarchical Super-Regions and their Applications to Biological Volume Segmentation

by

Imanol Luengo

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Computer Vision Laboratory

School of Computer Science

December 2017

Declaration of Authorship

I, IMANOL LUENGO, declare that this thesis titled, 'Hierarchical Super-Regions and their Applications to Biological Volume Segmentation' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

THE UNIVERSITY OF NOTTINGHAM

Abstract

Computer Vision Laboratory
School of Computer Science

Doctor of Philosophy

by Imanol Luengo

Advances in Biological Imaging technology have made possible imaging of sub-cellular samples with an unprecedented resolution. By using Tomographic Reconstruction biological researchers can now obtain volumetric reconstructions for whole cells in near-native state using cryo-Soft X-ray Tomography or even smaller sub-cellular regions with cryo-Electron Tomography. These technologies allow for visualization, exploration and analysis of very exciting biological samples, however, it doesn't come without its challenges. Poor signal-to-noise ratio, low contrast and other sample preparation and reconstruction artifacts make these 3D datasets to be a great challenge for the image processing and computer vision community. Without previous available annotations due to the biological relevance of the datasets (which makes them not being publicly available) and the scarce previous research in the field, (semi-)automatic segmentation of these datasets tends to fail.

In order to bring state-of-the-art in computer vision closer to the biological community and overcome the difficulties previously mentioned, we are going to build towards a semi-automatic segmentation framework. To do so, we will first introduce superpixels, a group of adjacent pixels that share similar characteristics that reduce whole images to a few superpixels that still preserve important information of the image. Superpixels have been used in the recent literature to speed up object detection, tracking and scene parsing systems. The reduced representation of the image with a few regions allows for faster processing on the subsequent algorithms applied over them. Two novel superpixel algorithms will be presented, introducing with them what we call a Super-Region Hierarchy. A Super-Region Hierarchy is composed of similar regions agglomerated hierarchically. We will show that exploiting this hierarchy in both directions (bottom-up and top-down) helps improving the quality of the superpixels and generalizing them to images of large dimensionality.

Then, superpixels are going to be extended to 3D (named supervoxels), resulting in a variation of two new algorithms ready to be applied to large biological volumes. We will show that representing biological volumes with supervoxels helps not only to dramatically reduce the computational complexity of the analysis (as billions of voxels can be accurately represented with few thousand supervoxels), but also improve the accuracy of the analysis itself by reducing the local noisy neighbourhood of these datasets when grouping voxel features within supervoxels. These regions are only as poweful as the features that represent them, and thus, an in-depth discussion about biological features and grouping methods will lead the way to our first interactive segmentation model, by gathering contextual information from super-regions and hierarchical segmentation layers to allow for segmentation of large regions of the volume with few user input (in the form of annotations or scribbles).

Moving forward to improve the interactive segmentation model, a novel algorithm will be presented to extract the most representative (or relevant) sub-volumes from a 3D dataset, since the lack of training data is one of the deciding factors for automatic approaches to fail. We will show that by serving small sub-volumes to the user to be segmented and applying Active Learning to select the next best sub-volume, the number of user interactions needed to completely segment a 3D volume is dramatically reduced. A novel classifier based on Random Forests will be presented to better benefit from these regions of known shape.

To finish, SuRVoS will be introduced. A novel fully functional and publicly available workbench based on the work presented here. It is a software tool that comprises most of the ideas, problem formulations and algorithms into a single user interface. It allows a user to interactively segment arbitrary volumetric datasets in a very intuitive and easy to use manner. We have then covered all the topics from data representation to segmentation of biological volumes, and provide with a software tool that hopefully will help closing the gap between biological imaging and computer vision, allowing to generate annotations (or ground truth as it is known in computer vision) much quicker with the aim of gathering a large biological segmentation database to be used in future large-scale completely automatic projects.

Acknowledgements

I had a wonderful time at Nottingham over the last years and I'm very grateful to several people who stayed with me through the journey of this thesis, either in the form of collaboration and contributions or just through their friendship.

I would like to first thank Andrew P. French, Mark Basham and Tony Pridmore, who accepted me to work with them after a very brief skype interview when my English skills were still not very good and I had some troubles maintaining a conversation.

Andrew has helped me a lot through the thesis, starting with weekly meetings with very useful advices to guide through a good research direction. I've also met very good friends at the Computer Vision Laboratory, such as Amy Lowe, Xiaomeng Wang, Kike Sanchez-Lozano or Brais Martinez, which made the work office a very friendly space where it was a pleasure to work.

On the other hand, Mark Basham has helped me to understand the whole picture of the problem and helped me to identify problems and solutions for challenging data. This thesis couldn't have been done without the help of the B24 Beamline at Diamond, in particular Michele Darrow and Matthew Spink, which have not only been collaborators but also a good friends, and helped me through my journey into the biological world.

Although being far away, Ander Biguri has been of unmeasurable help these last years. We have been listening, supporting and advising each other and that made the bad days of the PhD fade away quickly.

Naroa, my parner, without whom I truly wouldn't have been able to finish this thesis. She has always been by my side. In the bad moments cheering me up and in the good ones enjoying the life as if it was the last day on it. She has been an incredible support these years, and I will make sure I'll pay this enormous debt to her whenever she most needs it.

Last, I would also like to thank my parents, who have supported me since I was a child, and they are the main reason that I'm here, doing what I am doing. They have always provided me with the confidence I needed to face new challenges, and they always have been proud of my decisions.

Contents

Declaration of Authorship i

Abstract ii

Acknowledgements v

List of Figures x

List of Tables xiii

1 Introduction and Motivation 1
- 1.1 Cryo-Soft X-ray Tomography (Cryo-SXT) . . . 6
  - 1.1.1 Cryo-SXT data and challenges . . . 8
- 1.2 Thesis Overview . . . 11
- 1.3 Contributions . . . 15

I Super-Regions: Towards Better Data Representation 18

2 Introduction to Image Segmentation and Superpixels 19
- 2.1 Interactive Segmentation . . . 22
- 2.2 Superpixels . . . 25
  - 2.2.1 Normalized Cuts (NC) . . . 27
  - 2.2.2 Efficient Graph-based Segmentation (FH) . . . 28
  - 2.2.3 Quickshift (QS) . . . 29
  - 2.2.4 Turbo Pixels (TP) . . . 30
  - 2.2.5 Simple Linear Iterative Clustering (SLIC) . . . 31
  - 2.2.6 Superpixels in an Energy Optimization Framework (EneOpt) . . . 32
  - 2.2.7 Entropy Rate Superpixels (ERS) . . . 34
  - 2.2.8 Superpixels Extracted via Energy-Driven Sampling (SEEDS) . . . 35
  - 2.2.9 Linear Spectral Clustering superpixels . . . 37
  - 2.2.10 Discussion on Superpixels . . . 38

3 Hierarchical Piecewise-Constant Superpixels 40
- 3.1 Super-Region Hierarchy . . . 41

3.2 From Graph Partitioning to Image Segmentation . . . 43
3.3 PCS: superpixels from Piece-wise Constant Super-Regions . . . 46
3.4 Hierarchical generalization . . . 50
3.5 Effect of parameters of HPCS . . . 52
3.5.1 Controlling the strength of the pairwise splitting term, λ . . . 52
3.5.2 Controlling the number of centroids to be sampled . . . 54
3.6 Experimental Comparison of HPCS with state of the art . . . 56
3.6.1 Qualitative Evaluation . . . 56
3.6.2 Quantitative Evaluation . . . 60
3.7 HPCS as post-processing for superpixel algorithms . . . 62
3.8 Conclusions . . . 63

4 SMURFS: Superpixels from Multi-scale Refinement of Super-Regions 64
4.1 Superpixels from a two-level Super-region Hierarchy . . . 65
4.1.1 Splitting large regions . . . 66
4.1.2 Merge similar regions . . . 68
4.1.3 Similarities and differences between splitting and merging . . . 74
4.2 SMURFS Algorithm . . . 75
4.2.1 Hierarchical Generalization . . . 77
4.2.2 Hierarchical Updates and Hierarchical Consistency . . . 77
4.3 The sampling function Γ: towards a robust deterministic solution . . . 80
4.4 Parameters of SMURFS . . . 84
4.4.1 Controlling the strength of the pairwise splitting term, λ . . . 85
4.4.2 Controlling the number of centroids per region . . . 86
4.5 Extensions of SMURFS . . . 88
4.5.1 Q-SMURFS: Quick SMURFS . . . 89
4.5.2 R-SMURFS: Regular SMURFS . . . 90
4.5.3 B-SMURFS: SMURFS with boundary prior . . . 91
4.5.4 C-SMURFS: SMURFS with Contextual prior information . . . 92
4.6 Experimental Comparison of Superpixel Algorithms . . . 93
4.6.1 Qualitative Evaluation . . . 93
4.6.2 Quantitative Evaluation . . . 98
4.7 Other applications and properties of Super-Regions . . . 99
4.7.1 Better reconstructions for better segmentations . . . 99
4.7.2 Super-Regions as post-processing for superpixel algorithms . . . 100
4.7.3 Interactive refinement of Super-Regions with SMURFS . . . 101
4.7.4 Super-Regions for biological volume over-segmentation . . . 103
4.8 Discussion . . . 108

II Applications of Super-Regions to Biological Volume Segmentation 110

5 Introduction to the Segmentation of Biological Volumes 111
5.1 Understanding Cryo-Soft X-ray Tomography data . . . 117
5.2 Decision Forests for Classification . . . 124
5.2.1 Decision Trees . . . 124
5.2.2 Offline Training of a Decision Tree . . . 125

5.2.3 Split functions . . . 125
5.2.4 Introducing randomness in the training . . . 126
5.2.5 Training Decision Trees for classification . . . 127
5.2.6 Class prediction with Decision Trees . . . 129
5.2.7 Ensemble models . . . 129
5.2.8 Other Ensemble Models . . . 130
5.2.9 Why to use (Extremely) Random Forests . . . 130
5.3 Decision Forests in Biomedical Imaging . . . 132
5.4 Discussion . . . 133

6 Super-Regions for Interactive Segmentation of Biological Volumes 135
6.1 Interactive segmentation with Hierarchical Segmentation Layers . . . 139
6.1.1 Supervoxel over-segmentation . . . 140
6.1.2 Robust Feature Extraction . . . 142
6.1.3 Interactive Classification with Contextual Segmentation Layers . . . 144
6.1.4 MRF-based Segmentation Refinement with prior Label Structure . . . 147
6.1.5 Quantitative evaluation . . . 148
6.1.6 Qualitative evaluation . . . 150
6.2 Representative Sub-Volumes for Interactive Volume Segmentation . . . 156
6.2.1 Segmentation framework . . . 158
6.2.2 Finding representative sub-volumes . . . 159
6.2.3 Grouping similar nearby sub-volumes . . . 160
6.2.4 Further refining window proposals . . . 161
6.2.5 Querying the next most informative sub-volume . . . 162
6.2.6 Experiments evaluation . . . 162
6.3 Extremely Contextual Forests . . . 166
6.3.1 Extremely Contextual Forests for Volume Segmentation . . . 168
6.3.2 Extremely Contextual Decision Trees . . . 169
6.3.3 Multi-scale generalization: from voxels to supervoxels . . . 173
6.3.4 Experiments and Results . . . 174
6.4 Discussion . . . 179

7 SuRVoS: Super-Region Volume Segmentation Workbench 180
7.1 Overview of SuRVoS Workbench . . . 183
7.2 Data Preparation . . . 185
7.3 Data Preprocessing and Feature enhancement . . . 185
7.4 Data Representation . . . 187
7.5 Model Training . . . 188
7.6 User Workflows . . . 191
7.7 Data Classification and Measurements . . . 192
7.8 Results . . . 193
7.8.1 PC-12 Cell . . . 193
7.8.2 Trypanosoma brucei cell . . . 195
7.8.3 Goosegrass' fruit body (micro-CT) . . . 196
7.8.4 Human Platelets (cryo-EM) . . . 196
7.8.5 Mammalian Cell-line (cryo-SXT) . . . 197
7.9 Discussion . . . 202

7.10 Conclusion 204

III Future Work **205**

8 Conclusions and Future Work **206**
8.1 Overview of the Main Contributions 207
8.2 Current State and Future Work 209

A Markov Random Field Models in Vision **211**

B Enhancing Features of Biological Volumes **214**

C Superpixels for Leaf Detection and Segmentation **217**

Bibliography **224**

List of Figures

1.1 Super-region Hierarchy . . . 5
1.2 X-Ray Tomography . . . 8
1.3 Cryo-Soft X-Ray Tomography samples . . . 9
1.4 Overview of the structure of this Thesis. . . . 11

2.1 Example of images to be segmented. . . . 20
2.2 Example of a simple image from the BSD500 dataset. . . . 20
2.3 Example result image from Level Set evolutions . . . 23
2.4 Example of image segmentation with user annotation as input using Graph Cut. Images adapted from (Boykov and Funka-Lea, 2006). . . . 24
2.5 Example of superpixel visualization strategies. . . . 26
2.6 Time-line of superpixel algorithms . . . 27
2.7 FH Superpixels Sample . . . 28
2.8 QuickShift Superpixels Sample . . . 30
2.9 Turbo Pixels Sample . . . 31
2.10 SLIC Superpixels Sample . . . 32
2.11 ERS Superpixels Sample . . . 34
2.12 SEEDS Superpixels Sample . . . 36
2.13 LSC Superpixels Sample . . . 37

3.1 Overview of a Super-region Hierarchy . . . 41
3.2 Super-region Hierarchy . . . 42
3.3 Overview of PCS . . . 49
3.4 Reconstruction of PCS . . . 49
3.5 HPCS as a post-processing of PCS and SLIC superpixels. . . . 52
3.6 Lambda in HPCS . . . 53
3.7 λ in HPCS . . . 54
3.8 Number of clusters in HPCS . . . 55
3.9 Number of clusters in HPCS . . . 56
3.10 Example of four images of BSD500 dataset. . . . 57
3.11 Comparison of segmentation algorithms . . . 58
3.12 Overview and comparison of PCS superpixels to state of the art with $M = 400$ superpixels. . . . 58
3.13 Overview and comparison of PCS superpixels to state of the art with $M = 400$ superpixels. . . . 59
3.14 Overview and comparison of PCS superpixels to state of the art with $M = 400$ superpixels. . . . 60
3.15 Quantitative evaluation of HPCS. . . . 61

3.16 Example of using HPCS as post-processing algorithm. 62

4.1 Graphical overview of SMURFS 65
4.2 Region splitting procedure with $K = 2$ cluster centres. 69
4.3 Example of a Region Adjacency Graph 70
4.6 Higher Level SMURFS: F1 77
4.7 Higher Level SMURFS: F2 78
4.8 Example of PCA-Part initialization for k-means. For the initial data (already coloured with real cluster membership for easier visualization), the axis with largest variance is the y axis, which gets splitted along its optimal split point ($y = 0$). Of the two newly created clusters, the one with largest variance is the top cluster along the x axis, of which optimal split point is $x = 0$. The centroids (marked as red dots) are then calculated by averaging data points within the clusters. 84
4.9 λ in SMURFS 85
4.10 λ in SMURFS 86
4.11 Number of clusters in SMURFS 87
4.12 Number of clusters in SMURFS 87
4.13 Quick SMURFS 90
4.14 Regular SMURFS 90
4.15 Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels. 94
4.16 Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels. 95
4.17 Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels (with boundary visualization). 96
4.18 Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels (with mean color visualization). 97
4.19 Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels. (a) enteOpt, (b) Turbopixels, (c) Superpixel Lattices, (d) SLIC, (e) SEEDS, (f) ERS, (g) LSC, (h) SMURFS. 98
4.20 Quantitative evaluation of SMURFS. 99
4.21 Image reconstruction using SMURFS. 100
4.22 Image reconstruction using SMURFS. 101
4.23 Interactive merging of super-regions. 102
4.24 Interactive splitting of super-regions. 102
4.25 Cryo-SXT denoising 104
4.26 SMURFS supervoxels applied to a 3D dataset. 105
4.27 Comparison of SMURFS supervoxels with SLIC supervoxels. 107

5.1 Broad overview of the resolution of different imaging techniques. Exceptions might apply using special techniques such as super-resolution light microscopy. 111
5.2 Example of cryo-SXT biological datasets, without any filter or preprocessing. 121
5.3 Example of cryo-EM biological datasets, without any filter or preprocessing. 122
5.4 Example of resin embedded FIB-SEM biological datasets, without any filter or preprocessing. 123

6.1 Hierarchical Segmentation Layers . . . 137
6.2 Region Adjacency Graph . . . 138
6.3 Overview of the interactive segmentation algorithm. . . . 144
6.4 Overview of the hierarchical classification. . . . 146
6.5 Creating the label cost matrix $\mathbf{C}$ from label structure. $0 < c_1 < c_2 \ll c_3$. . 148
6.6 Results of interactive segmentation with supervoxels. . . . 150
6.7 Results of interactive segmentation with supervoxels. . . . 151
6.8 Results of interactive segmentation with supervoxels. . . . 153
6.9 Results of interactive segmentation with supervoxels. . . . 154
6.10 Further cell segmentation with contextual information and MRF-based refinement. . . . 155
6.11 Overview of our proposed pipeline . . . 156
6.12 Overview of the window proposal method. For visualization purposes a 2D slice is shown, but every step is performed in 3D. . . . 159
6.13 Example representative sub-volumes extracted in an unsupervised fashion. 165
6.14 Underfitting vs overfitting . . . 167
6.15 Extremely Contextual Forest sampling . . . 170
6.16 Mitochondria segmentation with ECF . . . 176
6.17 Semantic segmentation with ECF . . . 177
6.18 Semantic segmentation with MS-ECF . . . 178

7.1 SuRVoS Workbench interface. . . . 181
7.2 Overview of the SuRVoS framework. . . . 184
7.3 Output of different denoising and textural features on a cryo-SXT dataset. 186
7.4 Super-region hierarchy example using a RoI of a cryo-SXT dataset . . . 188
7.5 Classification of a representative cryo-SXT volume . . . 191
7.6 Qualitative results for the PC-12 cell segmentation . . . 194
7.7 ualitative results for the Trypanosoma brucei cell. . . . 195
7.8 Processing of a microCT dataset using a model training segmentation strategy. . . . 199
7.9 Processing of a cryoET dataset using a super-region segmentation strategy. 200
7.10 Analysis of a cryoSXT dataset using the label splitter tool. . . . 201

B.1 Denoising features . . . 215
B.2 Local features . . . 215
B.3 Blob-like detectors . . . 216
B.4 Tubular structure detection . . . 216

C.1 Sample colour and grayscale images of ‘A1‘ dataset, along with its ground truth. . . . 217
C.2 Drawing of the proposed pipeline. . . . 219
C.3 Example results of our proposed pipeline for CVPPP. . . . 220
C.4 Example results of our proposed pipeline for CVPPP. . . . 221
C.5 Example results of our proposed pipeline for CVPPP. . . . 222
C.6 Results for the individual leaf segmentation challenge. . . . 223

List of Tables

3.1 Default parameters for HPCS 52

4.1 Disjoint-Set operations. 73
4.2 Default parameters for SMURFS 85
4.3 Default design choices for HPCS and SMURFS 88
4.4 Computational times for some superpixel algorithms. 99

6.1 Textural SigmaSet feature performance 149
6.2 Computational times for a $1000 \times 1000 \times 200$ volume. Note that the user can loop steps $3-5$ hierarchically. 150
6.3 Performance of our segmentation pipeline in the testing dataset 163
6.4 DICE coefficient of the simulated retrieval methods. Percentages indicate fractions of total training data. 165

1 Introduction and Motivation

Advances in technology and image processing make possible for biologists to image micro-organisms, cells and even molecules, in order to study their structure, properties and interactions. Special microscopes are used to be able to extract information from these very small specimens, which range from nanometres to a few micrometers in size. These microscopies operate within a large variety of imaging modalities (e.g. Electron Microscopy or X-rays) to generate 2D images or 3D volumes that capture different information at specific resolution from a biological specimen. With different properties, advantages and weaknesses in each of the imaging modalities, the resultant datasets (images or volumes) are analysed in order to find new interesting biological breakthroughs.

The amount of data generated by current biological imaging modalities is, however, overtaking our ability to efficiently analyse it. In order to analyse a biological volume, knowledge from a human expert is required in order to visually explore the dataset and manually annotate different areas and sections from it for later measurement. This manual segmentation labour is tedious, error prone and time consuming (Hecksel et al., 2016a), and yet, is a crucial step to fully analyse the scientific content within the data. Not having access to convenient tools to segment a volume results in many biological volumes remaining under-utilized. Hence, it comes as no surprise that there has been a lot of research focus in the recent years on trying to automate this biological volume segmentation task.

In the last decade, advances in Computer Vision made possible for computers to understand images and videos to an unprecedented level: from identifying objects or detecting and estimating human pose, to transferring artistic styles from one image to another or in-painting coherent details in damaged regions. These advances have been made by studying natural images, this is, real world images taken with a standard camera. Currently, efforts are being made to adapt these state-of-the-art techniques to the biological world, not without finding troubles in the process.

Computer Vision is a research field originated around the 1960s as an applied field born from the combination of mathematics and artificial intelligence. Since then, researchers have been trying to understand the human visual system by investigating mathematical models that simulate it, with the aim of automatically analysing and understanding images as we do. In the field, images are represented as numerical data in the form of a matrix I of size $width \times height$ pixels, adding an extra dimension for multi-channel images (such as a standard RGB image with 3 channels). A pixel has since then been the smallest representative unit in an image, representing the intensity (or colour) of a single point in the image. Image Segmentation, the process of partitioning an image into meaningful regions (by assigning to each pixel the higher level object to which they belong), has always been an important research area within the field, as it is thought to be a necessary intermediate step towards understanding what is happening in the scene. Early research in Image Segmentation was driven by simple threshold techniques, identifying objects with pixels within a range of intensities. As it didn't provide very satisfactory results, the next movement was focused on analysing the contour and edges in the image with the aim of delineating the different objects appearing on it. Initial approaches on contour and edge extraction were completely unsupervised and using only local information on the vicinity of each of the image pixels. These methods were later replaced by more robust techniques, using pixels to grow regions and contour models with the aim of reducing the noise of a local neighbourhood by enforcing *global* consistency to the segmentation results. These methods of image segmentation fall into the unsupervised category. Recent advances in this field have moved towards image over-segmentation, which include so called *Superpixels*. Superpixels are, by definition, groups of adjacent and similar pixels that belong to the same higher-level object. Superpixel algorithms perform unsupervised segmentation of an image to partition it in a predefined number of regions (i.e. partition an image into 400 superpixels). Superpixels do not represent whole objects but, at least theoretically, guarantee that all the pixels within a superpixel belong to the same object. This approach over-segments the image in more regions than objects in the image, but guarantees a successful partition, which is a solvable problem in comparison with image segmentation, which is a more difficult formulation. However, once these superpixels have been created, they can be used for various tasks, such as object detection, recognition or even image segmentation by identifying to which object they belong to and grouping them further.

Over the last two decades learning-based approaches (by combining image processing with the machine learning field) have demonstrated that algorithms that *learn to* do something (e.g. segment, track, detect, etc..) often outperform our ability to model similar approaches that do so. This is probably due to the difficulty of mathematically

modelling the world, which we don't completely understand yet, and our lack of ability to do it properly. Hand-crafted mathematical models have only demonstrated to give good results in very particular (and controlled) cases. Recently, Deep Learning algorithms have strengthened this premise by outperforming not only human-made unsupervised algorithms, but outperforming humans themselves on various tasks, such as object recognition in the ImageNet dataset. As of today, we (humans) have made computer algorithms that recognize objects in images better than we actually do. We have also made computers (AlphaGO (Silver et al., 2016)) that not only are able to play the most difficult board game in the world, GO, but are also able to beat the human world championship. Furthermore, the creators of AlphaGO didn't even know how to play the game GO properly. A machine learning program learned to play GO from recorded matches and was then asked to play against itself, with no knowledge of the rules of the game, for millions of times, until it itself figured out which actions would lead it to win the game. This suggests that we are better at creating intelligence that learns to model things, than we are at mathematically modelling them. Current research is, thus, focused on creating very complex machine learning models, and trying to understand their behaviour afterwards. This is a very exciting opportunity for us, as we can learn to create mathematical models that then will teach us back.

Biological datasets, specially large 3D volumes, present new and exciting challenges and opportunities to the Computer Vision field. For instance, biological datasets are usually much larger than images from standard Computer Vision databases. A sample biological volume is usually about $500 \times 500 \times 500$ voxels ($\sim$ 0.5 GB of RAM), going up to $2000 \times 2000 \times 2000$ ($\sim$ 30 GB of RAM), and is likely to dramatically increase due to advances in detector technologies. Standard Computer Vision methods can't deal with such amount of data due to limitations on computational complexity that easily become prohibitive when the 3rd dimension is added and memory limitations. Thus, more efficient methods are needed in order to solve an equivalent problem. Additionally, state-of-the-art computer vision algorithms require a large amount of training data (i.e. manually annotated data), which is not yet available in the biological sciences. Annotations to learn from are missing in most of the situations for various reasons.

1. Wide variety of biological samples.
2. Very different imaging modalities used to image them.
3. Different resolution at which the imaging techniques work.
4. Motivations for capturing the datasets.
5. Very tedious task of manually annotating large volumes, among others.

Thus, the current state of biological volume segmentation research is still mainly focused on region and contour based models. Learning-based strategies are slowly taking place into the field in the form of interactive segmentation, that is, by keeping the human-in-the-loop and using user input to learn weak models in a per-dataset basis. These weak models are incrementally improved with the amount of user input, and are then used to extrapolate the learned knowledge to the rest of the volume. The goal is then to fully segment a given volume with the minimum possible user interaction. Recent research is starting to include the aforementioned superpixels in their interactive segmentation formulation. When superpixels are extracted over the 3D volume, they are usually refereed as supervoxels, as a *voxel* is the denomination of a 3D pixel (or volumetric pixel). These supervoxels have shown to dramatically reduce the complexity of the subsequent tasks (in this case, volume segmentation), as the same dataset is represented by much fewer supervoxels (than voxels), and thus, reducing the computational complexity several orders of magnitude.

The properties of superpixels or supervoxels, however, are still not entirely well understood, with contradicting works in the literature recommending the usage of ones over the others. Their full potential in biological images hasn't been explored yet either, as only a few recent works benefit from their use. With the aim of giving insight to the biological volume segmentation field, to fill the gap in knowledge between theoretical and practical applications of superpixels and supervoxels and to provide practical methods to segment biological datasets by adapting state-of-the-art computer vision techniques, this Thesis will be separated in two main parts:

1. Super-Regions, a novel hierarchical supervoxel over-segmentation: Two novel superpixel algorithms (and their extension to 3D) will be introduced and compared to the state of the art. The novel hierarchical supervoxels better capture the boundary of biological volumes.

2. Super-regions for the interactive segmentation of biological volumes: a new framework will be proposed towards interactive biomedical volume segmentation.

Two novel superpixel/supervoxel algorithm will be proposed in the first part of the thesis. This new algorithm will be evaluated and compared to a variety of state-of-the-art superpixel algorithms and their properties will be studied in depth. Our new contribution is a generalizable algorithm capable of extracting what we call Super-Regions, an over-segmentation level-hierarchy where elements of each level are computed by grouping similar and adjacent elements of the previous layers, with the initial layer being the pixel/voxel grid (illustrated in Figure 1.1). These super-regions will then be used to segment biological volumes by incorporating human knowledge into an interactive

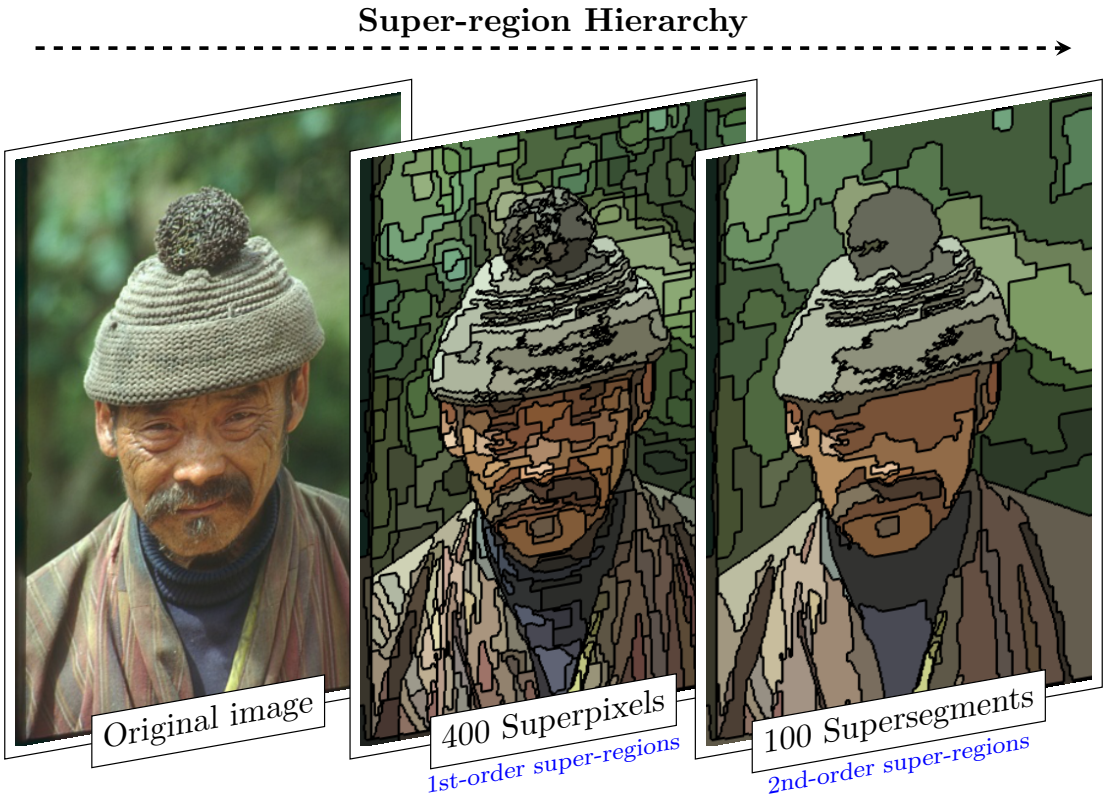


Figure 1.1: Super-region Hierarchy on a standard image of the BSD500 dataset. Pixels form the 0th-order super-region level. Superpixels are formed by grouping similar and adjacent pixels. Their inherent boundary adherence allow to represent high level objects of the image with much less representative units. Super-segments, similarly, are formed by grouping similar and adjacent superpixels together. This super-region hierarchy reduces the number of representative units needed to reconstruct an image.

framework which will incrementally learn to segment a given volume. By doing this, the human effort needed to segment a volume will be dramatically reduced from days or even weeks to a few hours. To demonstrate and validate our proposed segmentation pipelines a software workbench will be presented, which is already installed in a few research facilities and being used by several biologists around the world.

The following section will give a brief introduction to biomedical volumes, explaining how these datasets are usually obtained, with examples of a few imaging modalities and a particular focus on cryo-soft X-ray tomography (CryoSXT). The limitations of these imaging techniques will be briefly discussed, giving an initial outline of the challenges of the segmentation of these datasets.

1.1 Cryo-Soft X-ray Tomography (Cryo-SXT)

Standard Light microscopes, the most common and affordable ones, are unable to visualize structures less than the diffraction limit of light, which is related to the wavelength used, around $0.5\mu m$ for visible light. These microscopes are widely used, but lack the resolution necessary to study structural properties of micro-organisms, cells and sub-cellular elements. Electron microscopes (EM) have been developed in order to obtain higher resolution images of smaller structures, with the first prototype being dated back to 1937. These microscopes use electrons as illumination an source to reveal the structure and properties of objects that the light is not able to resolve, as the electron wavelength is much smaller than the visible light. There is a wide variety of electron microscopes, such a Transmission Electron microscope (TEM) or Scanning Electron microscope (SEM), with differences in how the biological sample is prepared and imaged, but, in general, electron microscopes lack the ability to image whole cells in their native-state as they require disruptive sample preparation prior to imaging.

Electron microscopes allow for the image of very small structures, however, electrons don't carry enough energy to penetrate whole cells, and thus, specific sample preparation techniques have to be used in order to image cells thicker than $750nm$. Scanning Electron microscope can scan the surface or topology of a given sample. In order to image whole cells, however, an iterative step of milling/slicing is necessary. There are two main procedures to slice a cell: Focus Ion Beam (FIB) and Serial Block Face (SBF), each one introducing different damage artifacts to the sample. In addition, the sample has to be prepared and stabilized prior to milling, and it can be done either at room temperature (e.g. by chemical fixation, resin embedding) or at cryo temperatures (e.g. plunge freezing). While the former is easier to perform and usually sufficient for most of the studies, the cells are fixed with external agents that fix some of the properties but don't affect others (e.g. some fixation techniques will fix membranes or organelles but not inner cell fluids or other structures), thus, imaging a cell in a fixed state, but not necessarily its real state. On the other hand, cryo-SEM is extremely challenging and can only be done in a small area of interest. Instant freeze of section of cells will preserve it as close as possible to its native state, but this technique is very new and still not widely used. Iterating imaging and milling operations will eventually image whole cells, but other challenges arise when trying to reconstruct the whole cell from separate parts.

The Transmission Electron microscope (TEM) can be used to image samples that are thinner than $750nm$. TEM works by throwing electrons towards the sample and capturing the absorption coefficient of electrons in a detector behind the sample. This

will create a 2D projection of the sample, and multiple projections are then combined through tomography subsection 1.1.1 to generate a 3D volume. As with SEM, it can be used to image at room temperature, or cryo temperatures for closer to native state imaging. Its main limitation is the thickness of the sample due to electron absorption, which makes it only suitable to image bacteria, yeast or edges of cells. However, milling strategies such as the previously mentioned FIB or SBF could be used to extract a thin sample and the reconstruct it in 3D.

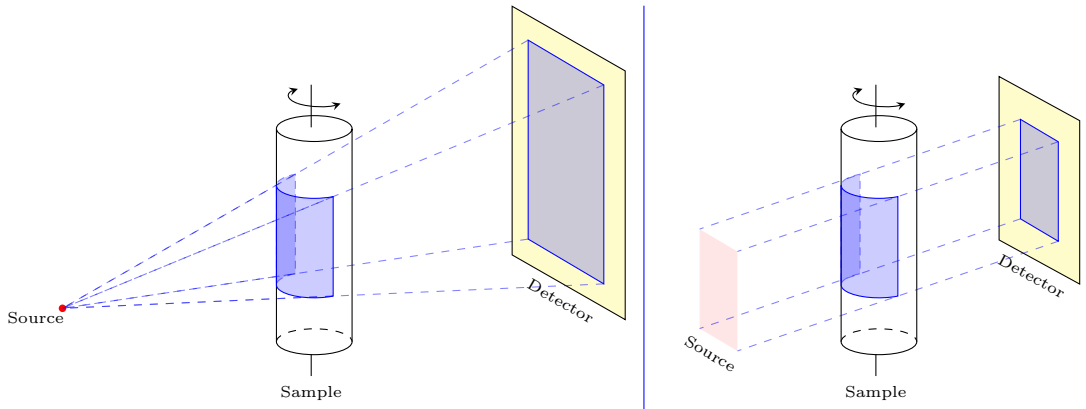
In the last years, a new technique has been gaining attention, named cryo-soft X-ray tomography (Cryo-SXT), that uses soft X-rays that are able to penetrate a whole specimen. These soft x-rays operate in the so called *water-window*, which energies vary from 280 to 530eV. At these energies the X-rays still have enough energy to penetrate up to $10\mu m$ of biological specimens, which is more than the average thickness of a cell. The water-rich components (e.g. as the ice formed when the cell is vitrified or the water inside the cell such us the cytoplasm) are nearly transparent to these X-rays, as they are not attenuated when passing through them, while carbon and nitrogen-rich elements attenuate X-rays that passes through them, providing natural-contrast to the images without need of external contrast-enhancing agents. Additionally, since the X-rays have higher penetration power, the samples to be imaged can be grown in near-living environments without disruptive milling/slicing strategies. Whole populations of cells are then put into the microscope and the X-rays are focused on a single cell, often selected by using supplemental techniques such as Fluorescent Microscopy. Samples are frozen prior to imaging and, during the imaging, they remain intact in a vitrified state. This some benefits, such as using of cryo immobilization, which instantly fixes the cell in a near native state, and at the same time protects from radiation damage from the X-rays.

The process of imaging a sample in a cryo-SXT can then be summarized as follows:

1. A biological sample, to be studied, is placed into the microscope.
2. Soft X-rays illuminate the sample.
3. A Detector gathers information about X-rays which pass through the sample, which allows the absorption coefficient to be calculated.
4. The sample is rotated, and steps 2-3 are repeated until the available field of view has been explored.

The detector creates a 2D projection of the sample being imaged by capturing information about the source that arrives to it. By rotating the sample different projections are acquired at different orientations, which are then used to reconstruct a 3D volume from

2D projections. Figure 1.2 provides an overview of a simplified projection acquisition with X-rays. It shows two different geometries depending on the source of X-ray chosen. On the left, it shows cone-beam X-rays, which are the most common ones and are available at (hard) X-ray machines at hospitals. On the right, it shows parallel X-rays used as source within the cryo-SXT. These X-rays travel in parallel to the sample, making its projection more accurate and the 3D reconstruction much simpler.



(a) Cone Beam (b) Parallel Beam

Figure 1.2: X-Ray tomography geometries for two most common X-Ray sources. Cone beam is used in many medical imaging applications (e.g. at a hospital when taking a X-ray of humans). Parallel beams are harder to obtain, but provide better quality projections easier to interpret and reconstruct into a 3D volume.

However, in order to generate a large quantity of the parallel soft X-rays used within Cryo-SXT, a particle accelerator is needed. Electrons are accelerated to near the speed of light and forced to travel through very powerful magnets in a circle-like structure (called a *storage-ring*). By making the electrons turn through magnetic fields energy is lost in the form of photons, which can be tuned to create soft X-rays. There are only five synchrotron based cryo-SXT microscopes in the world due to their energy and specific installation requirements.

1.1.1 Cryo-SXT data and challenges

The data generated from cryo-SXT microscopes is of high biological relevance, as it is can be used to study samples that cannot be easily imaged with other techniques. With the very interesting biological relevance and limited availability comes the lack of previous studies and freely available data from the computer vision domain. The resulting datasets are up to $2000 \times 2000 \times 2000$ voxels in size (with standard sizes of $1000 \times 1000 \times 500$) volumes of sub-cellular structures. The intra-cellular organelles are imaged in a 3D volumes that contain tens to hundreds of structures, are very tedious to manually annotate and very difficult to automatize due to the lack of previous studies

and previously annotated samples. Figure 1.3 show example central slices of 3D volumes of 5 different cryo-SXT datasets and the challenges on the segmentation of these datasets are presented below.

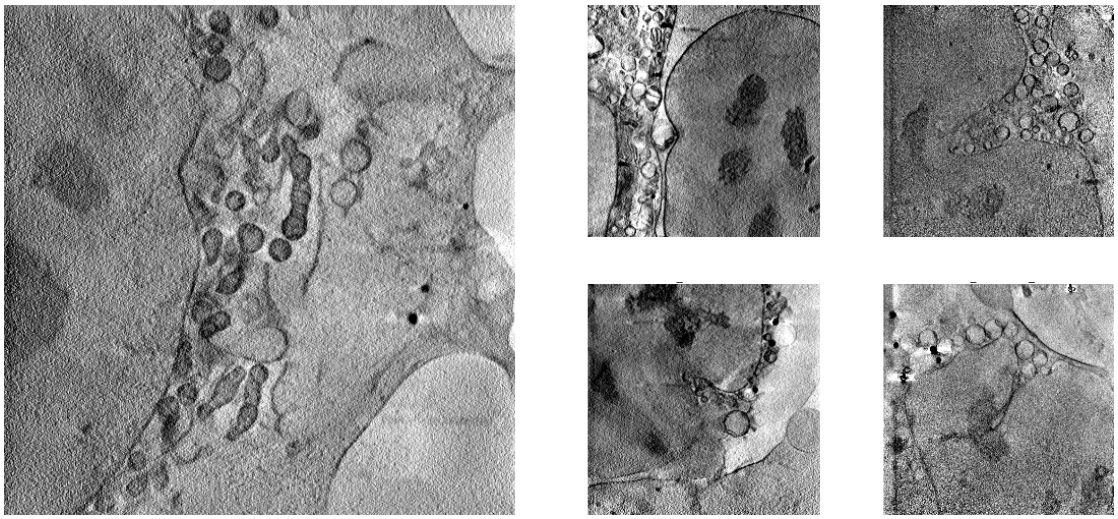


Figure 1.3: Central Slices of 5 different cryo-SXT volumes are shown. Although volumes are created with the same microscopy, organelles' shape and appearance have very high variance within the different cells.

Abscence of previous work

Very few previous works have been found into how to automate segmentation of cryo-SXT volumes due to the novel and very particular imaging modality. During this project state-of-the-art techniques from computer vision and medical imaging have been studied in order to offer a solution to these new challenging datasets.

Data dimensionality

Volumes are from $1000 \times 1000 \times 500$ (1.8 GB of RAM) to $2000 \times 2000 \times 2000$ voxels (36 GB of RAM). Standard computer vision techniques are not designed to work with such amount of data, which introduces many difficulties when trying to optimize global functions or apply functions to the whole volume, as in most of the cases the computational complexity of the solution increases at least linearly with the size of the data.

Dataset variability

This imaging technique is usually utilized to study different specimens. Volumes are extracted per specimen and similar specimens are rarely analysed. Thus, different volumes

represent different cells (or even cell types), with different kind of organelles which have very high variance in shape and appearance.

Signal to noise ratio and artefacts

Another challenge of these volumes is their signal to noise ratio being very poor, as low dosage is needed to void damaging the samples. A problem of the cryo-SXT microscopes is that due to the short working distance of X-ray optics, the sample can only be rotated from $-70°$ to $70°$ in intervals of $0.5°$, making a total of 241 projections used for the 3D reconstruction. Being unable to capture a full $180°$ view of the data introduces artefacts and noise in the volume. By adding the Poisson noise that characterize the X-rays and its reconstruction, volumes end up with a very low signal to noise ratio and several artefacts and damaged areas that diffitult the analysis of the underlying structures.

Lack of training data

Due to the relevance of the samples and the privacy issues when sharing them and the variability of the datasets, finding a large enough quantity of already annotated datasets to train state-of-the-art computer vision approaches is most of the times infeasible. Collected datasets often have very high biological relevance, preventing them from being publicly released as the study of their biological properties can take years from when they are collected.

Multi-class segmentation

Most of the segmentation works in the literature try to automatically or semi-automatically segment the salient object in the image. This is, it is assumed that there is an object of interest in the image and the goal is to detect and segment it. In this case, however, volumes are huge, not just 2D images, and there are tens to hundreds of organelles of various different types within the cell. Segmenting them all could be formulated as a multi-class segmentation problem, which shares some similarities with an active research field termed *scene-parsing* in computer vision, but approaches trying to solve scene-parsing use significant quantities of training data to learn to model scenes.

Towards the segmentation of cryo-SXT volumes

The final goal of this project has been to deliver a reusable system capable of segmenting all the individual structures of an image. For that, as there is limited training data

available, user interaction will be used as a source of knowledge to improve the segmentation methods. For more details about the nature of the cryo-SXT data, the imaging techniques and the cryo-SXT microscopy or the strengths and limitations of cryo-SXT we refer the reader to (Carzaniga et al., 2014; Duke et al., 2014b,a; Müller et al., 2012), including more details about the challenges of segmenting these volumes.

1.2 Thesis Overview

The content of the thesis will be separated in two main parts:

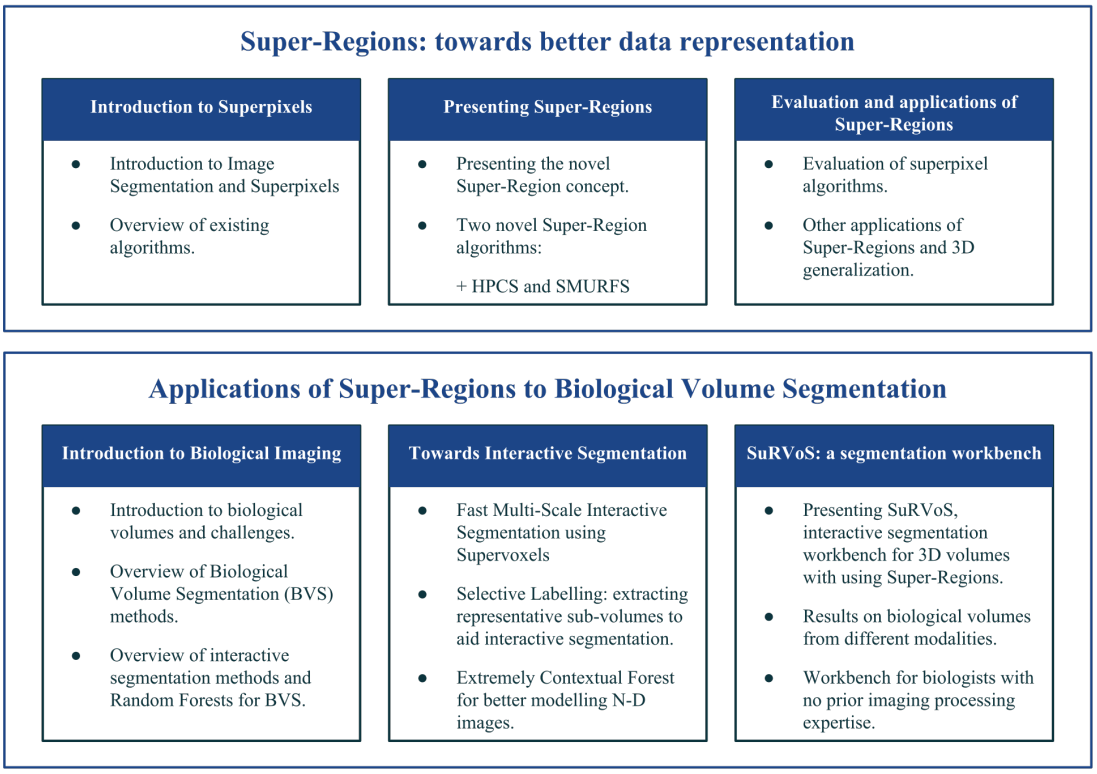


Figure 1.4: Overview of the structure of this Thesis. Divided in two parts: Super-Regions and their applications to biological volume segmentation.

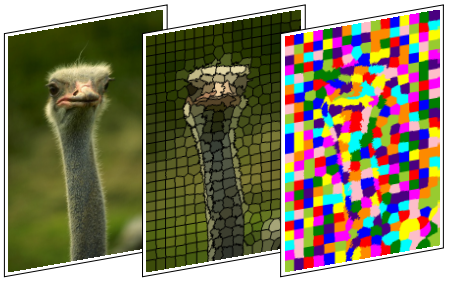
Part 1: Over the first part of the thesis superpixel algorithms will be categorized, discussed and evaluated. Two novel superpixel algorithms will be proposed, with the ability to generalize and cope with every possible scenario and subsequent application. These algorithms do not only match (or even outperform in some cases) state-of-the-art superpixels but do also generalize hierarchically, introducing Super-Regions. Finally, the applications of Super-Regions will be discussed and their generalization to 3D volumes will be presented.

Part 2: In the second part of the thesis more applied work will be discussed. Super-Regions will be taken as representative units for biological volumes and novel work on how to benefit from them towards interactive volume segmentation will be presented. A publicly and freely available workbench for 3D interactive volume segmentation will be introduced, with the aim of obtaining better and faster ground truth annotations.

Figure 1.4 provides overview of the structure of this project. The chapters of the first part will be separated as follows:

The following subsections will summarize the content of the rest of the Thesis on a per-chapter basis:

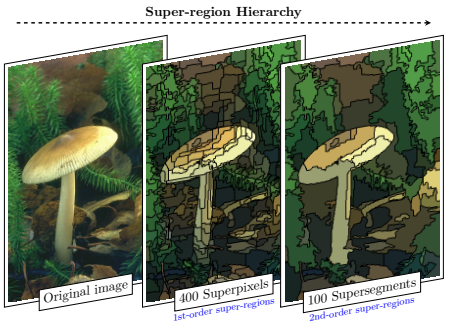
Chapter 2: Introduction to Image Segmentation and Superpixels



A thorough introduction to the problem of image segmentation will be given, with discussion about standard approaches for image segmentation and special focus on algorithms based on Markov Random Fields and interactive segmentation, as these are the foundation of the second part of this thesis.

The concept of superpixel will then be introduced, a group of similar and adjacent pixels that share similar characteristics. By using superpixels the same underlying data is represented with much fewer representative units (pixels, superpixels, etc..). This is particularly interesting looking forward to segmenting large 3D volumes, where the computational complexity of most of the current 2D segmentation algorithms and limitations in current hardware make them infeasible for large 3D volumes.

Chapter 3: HPCS: Hierarchical Piece-wise Constant Superpixels

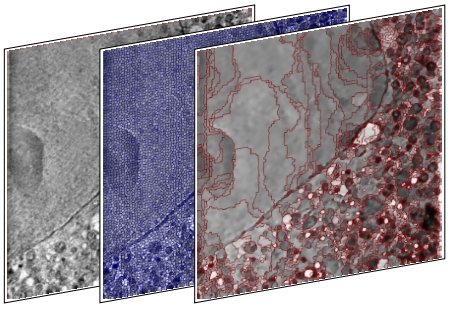


Superpixels are often refereed as over-segmentation methods, as they partition the image in meaningful regions while creating more regions than objects on the image. The concept of over-segmentation will then be generalized to form a hierarchy of over-segmentations that we call Super-Region Hierarchy. This hierarchy is formed by grouping similar and adjacent elements from the previous level.

This is, by grouping pixels (0-th order super-regions) superpixels (1st-order super-regions) are formed. Similarly grouping similar and adjacent superpixels will generate *super-segments* (as termed in the recent work of Ladickỳ et al. (2014)) as 2nd-order super-regions.

A novel algorithm, Hierarchical Piece-wise Constant Super-Regions (Luengo et al., 2016b) will be introduced and evaluated both qualitatively and quantitatively against the state-of-the-art.

Chapter 4: SMURFS: Superpixels From Multi-scale Refinement of Super-Regions

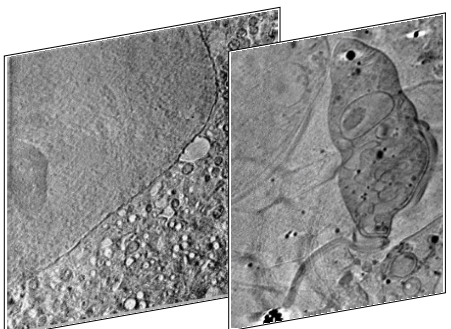


A novel super-region algorithm, SMURFS: Superpixels from Multi-Scale Refinement of Super-Regions (Luengo et al., 2016d), will be introduced, that not only create super-regions inherently, but also produce superpixels of similar or better quality than the state-of-the-art. SMURFS extends the concept of Super-Region Hierarchies to a whole new level and exploits their true potential.

A comparative study with the state-of-the-art superpixel approaches will validate the quality of our proposed super-region algorithms. Other extensions of SMURFS will be presented as validated, such as adding compactness constraints, a popular demanding characteristic for current superpixel algorithms (first introduced on SLIC (Achanta et al., 2012)), improved boundary adherence with a boundary prior and quick versions of SMURFS. The parameters of our algorithms then will be discussed and studied in depth, showing how different parameters affect the segmentation result.

Other application of our super-regions, such as using them as post-processing to improve the quality of other superpixel algorithms, or their extension to 3D to better represent biological volumes will then be discussed. I will show that by applying super-regions to 3D biological volumes the underlying biological data can be represented with much fewer supervoxels than using 3D SLIC.

Chapter 5: Introduction and Challenges of Biological Volume Segmentation

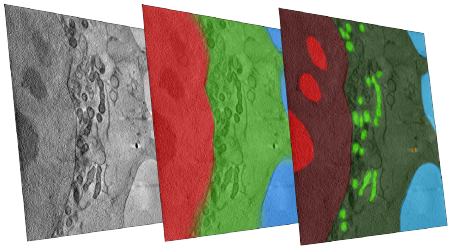


The second part of the Thesis will start by introducing the new challenges that arise when moving

from small 2D images (usually datasets are comprised of small images) to large 3D volumes. More insight will be given in cryo-SXT datasets to better understand how the data is collected and the complexity of it, including new particular problems that make standard segmentation algorithms fail to extract useful information from them.

Standard approaches towards 3D volume segmentation will be discussed, and special attention will be given to approaches based on interactive segmentation, active learning and Random Forests.

Chapter 6: Towards Interactive Segmentation of Biomedical Volumes



This chapter comprises most of the advances and discoveries completed during this research in the biological volume segmentation field. It is separated in three large sections, each of them representing a publication, detailing the contributions towards interactive segmentation during the term of this project.

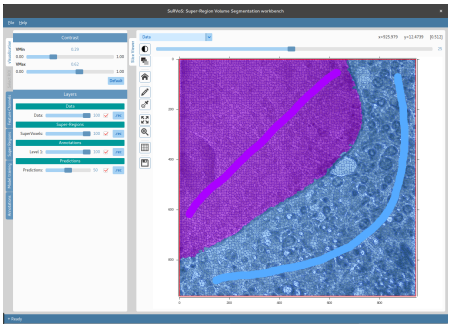
Starting with an initial novel attempt (Luengo et al., 2016a) to use supervoxels for interactive segmentation of sub-cellular organelles, Hierarchical Segmentation Layers will be introduced. They form a set of segmentation hierarchies where large cell areas are segmented in the first level of the hierarchy and more concrete, smaller and more challenging organelles are segmented later. The user input in the form of scribbles is used as the only training data to learn an appearance model of the different areas of the cell using Extremely Random Forests (Geurts et al., 2006). It will be shown that by splitting the segmentation problem into a two level hierarchy and using the result of the first level as contextual prior for the second level the segmentation accuracy is dramatically increased.

The limitations of our first attempt are presented, such as not being able to learn better and more complex segmentation model (popular in standard computer vision approaches) due to the scarce input training data. A novel algorithm to extract a massively reduced subset of representative sub-volumes is then introduced, Selective Labelling (Luengo et al., 2016c). We will show that this massively reduced subset of sub-volumes (which form only 5% of the whole training volume), combined with an active learning procedure, contain enough discriminative information to train a more

complex model and segment the rest of the volume. The quality of our representative sub-volumes will be validated both with fictitious agents and compared to the most similar state-of-the-art method trained on the whole available training data.

Last, a more robust segmentation model will be presented by introducing Extremely Contextual Forests (Luengo et al., 2017a), an ensemble method designed to learn more robust appearance models for voxels and supervoxels by exploiting the underlying grid-like structure of the datasets. This model is able to work well with biological datasets due to the previously mentioned Selective Labelling approach.

Chapter 7: SuRVoS: Super-Region Volume Segmentation Workbench



A large part of this project has been a collaboration with a group of biologists at Beamline B24: Cryo-Transmission X-ray Microscopy group at Diamond Light Source (B24). The main goal of this collaboration was for me to better understand the biological datasets, the collection method and their biological properties, as well as for them to understand how the knowledge and methods from the computer vision field could be applied to such challenging datasets.

As a result of this collaboration, a workbench has been developed and made publicly and freely available. SuRVoS: Super-Region Volume Segmentation workbench has been recently published in the Journal of Structural Biology (Luengo et al., 2017b) as a program to aid users to interactively segment challenging 3D data. The workbench is designed to work with many 3D volume and can be applied to many imaging modalities, as demonstrated in the visual experiment (Darrow et al., 2017). The final goal of SuRVoS is to make it easier for users to extract useful information from their 3D datasets, and ultimately to collect more *ground truth* (annotations) data easily so more complex machine learning methods could be applied in the future.

1.3 Contributions

The contributions of this work to the field are of two types: publications and open source software. Publications and pending publications in journals and conferences of computer vision and biomedical imaging are below:

Chapter 3 : Hierarchical Piece-wise Constant Super-Regions (Luengo et al., 2016b) and SMURFS: Superpixels from Multi-scale Refinement of Super-Regions (Luengo et al., 2016d) (Oral presentation at BMVC'16).

Chapter 4 : 3D SLIC superpixels on the GPU with dataset specific spacing, presented in (Luengo et al., 2016a).

Chapter 6 : Fast Interactive Volume Segmentation (Luengo et al., 2016a), Selective Labelling for Representative Sub-volume Extraction (Luengo et al., 2016c) and Extremely Contextual Forest for Multi-Scale N-D Image Segmentation Luengo et al. (2017a).

Chapter 7 : SuRVoS: a workbench for segmentation of biological volumes using Super-Regions (Luengo et al., 2017b) and a documented guide on the usage of SuRVoS for the segmentation of different datasets from imaging modalities (Darrow et al., 2017).

Appendix A : A Superpixel-based contribution for individual leaf segmentation in a collaborative study (Scharr et al., 2016).

Other talks have also be given to showcase work presented here, such as an oral presentation at the Tomography for Scientific Advancement conference (TOSCA 2016) (Luengo et al., 2016e) and another oral presentation by Michele Darrow at the European Microscopy Congress (EMC 2016) (Darrow et al., 2016), work highlighting using the cryo-SXT imaging modality and SuRVoS to find organelle changes in a Huntington's disease cell model. Additionally, two workshops have been organized on SuRVoS, to showcase the workbench to a group of interested biologist working with similar biological datasets.

In addition to publications, conferences and workshops, open source software has been released to support the research, validate it and make it accessible to the computer vision/biomedical imaging community.

SMURFS : A python and `C++` library of **S**uperpixels from **Mu**lti-scale **Ref**inement of **S**uper-regions (SMURFS), a novel superpixel algorithm presented in section section 4.2:

`https://github.com/imaluengo/SMURFS-Superpixels`

SuRVoS : **Su**per-**R**region **V**olume **S**egmentation Workbench. A `python/C++/CUDA` based Workbench for interactive segmentation of large biomedical volumes. Integrates standard computer vision/image processing techniques and some novel work along this thesis combined with efficient `CUDA` programming and `HDF5` files for efficient memory usage and computation:

https://diamondlightsource.github.io/SuRVoS/

https://github.com/DiamondLightSource/SuRVoS

Part I

Super-Regions: Towards Better Data Representation

2 Introduction to Image Segmentation and Superpixels

Images are everywhere. Pictures are constantly being taken by analog or digital cameras, uploaded to social media, framed on to the walls or broadcasted on the television. They form an important part of our every-day life. They help us remember the past, and allows us to dream of a better future. Our visual system is our best tool to understand the world, and yet we are excellent at interpreting images but awful at explaining our reasoning. Over the years images have been studied from different perspectives, such as psychology, mathematics and computer science with the aim to better understand how to create predictive models from images.

With such an amount of digital information collected in the form of images, learning how to properly analyse them has tremendous implications in future technology that could help us evolve towards a better future, from better environmental understanding or weather predictions from satellite images, to safer car driving via autonomous cars or psychological disorder diagnosis by studying videos of patients. The image analysis applications comprise a wide variety of areas, from biomedical imaging or plant phenotyping to robotics or even video-games.

We have learnt to develop intelligent systems that excel at image recognition or person tracking, but the state of the art is struggling to develop robust techniques towards image segmentation. Image segmentation is the problem of partitioning an image into a set of meaningful (non-overlapping) objects. It can be seen that image segmentation is a very ill-posed problem, as the proper definition leaves multiple interpretations, thus, it is very subjective and it is rarely limited to a single valid solution. The difficulty of image segmentation also varies dramatically from controlled environments to everyday, free form pictures, as shown in Figure 2.1. It can be seen that some problems have fairly obvious solutions, but others are nearly impossible to solve.



(a) Easy (b) Medium (c) Hard

Figure 2.1: Example of images to be segmented. The image in (a) clearly shows a plane, which is easy to delineate from the image. Image (b) shows a family of bears, which have a smooth boundaries (due to the fur) and some bears occlude others, which make the problem difficult. On the other hand (c) shows a highly challenging image as the texture of the lizard merges with the background. (Images from BSD500 dataset[1])

On top of that, image segmentation also has a wide variety of interpretations. For example, BSD500 is a dataset where the contours of all the objects in an image has been segmented by 4 to 5 humans. Figure 2.2 shows an example of why is segmentation very subjective and challenging. It can be seen that not even people do agree on a segmentation for a fairly simple image, suggesting that image segmentation is not only challenging, but dependant on the interests and perception of the segmenter. From all the human segmentations, we could say that none of them and all of them are right at the same time, as it would be difficult to argue why one is better than the others.

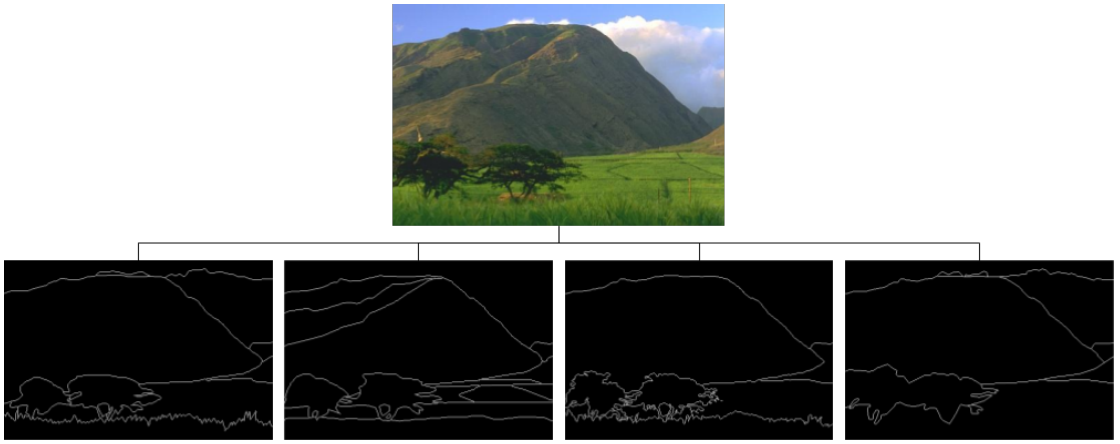


Figure 2.2: Example of a simple image from the BSD500 dataset with its corresponding annotations from 4 different people. It can be seen that although some of the boundaries match in all segmentations, some others completely differ. While some of the segmentations are more vague, some others are much more detailed. In some of them there are more abstract annotations from higher level regions while some others annotate more standard objects.

[1]BSD500: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

Lastly, there are many factors that can make object segmentation more challenging, such as complex textures, occlusion, truncation, missing boundaries or images taken from different viewpoints. The problem of image segmentation usually has different forms in the computer vision field and could be split into three categories:

Semantic Segmentation: a multitude of ground truth images and their corresponding labels are available for machine learning algorithms to learn from them. When testing the whole image needs to be segmented and classified into object labels. This is where the state of the art in Deep Convolutional Neural Networks (CNN) are having a great success lately. As there is plenty of training examples complex learning systems like CNNs are able to learn to jointly model object detection, recognition and segmentation (Chen et al., 2014), some of them using region proposals based on Perceptual Grouping (Ren et al., 2015)

Interactive Segmentation: Also refereed as semi-supervised image segmentation. It is the task of segmenting a given image using the user input (usually in the form of scribbles or contours) as the main input for the algorithms. Their strength is efficiency and final accuracy as the user can add more inputs until the result is perfect, however, most studies have only be applied to segmenting single objects (or salient objects) from images.

Perceptual Grouping: A form of unsupervised segmentation designed to split a given image into coherent regions. Since it is unsupervised, it rarely can extract the desired objects as one segment, but algorithms of perceptual grouping help to partition an image into fewer regions that could be used for subsequent image analysis methods. Superpixels, groups of similar and adjacent pixels, are the most common form of perceptual grouping techniques.

There are more variants of image segmentation, but those above are the main trends in the current state of the art, the last two being mainly of interest for this thesis. The reason is that the main goal of this thesis is to work towards generating more ground truth data for biological imaging, where lack of annotations prevent state of the art CNN techniques to be applied in most of the cases. Thus, interactive segmentation and perceptual grouping will be thoroughly discussed in this thesis. The next section will give a brief introduction to interactive segmentation, with more detailed content and applications to biological image segmentation on chapter 8 and chapter 6. The majority of the remainder of this chapter will then be based on an specific form of perceptual grouping termed superpixels.

2.1 Interactive Segmentation

Interactive segmentation methods have been popular in the literature since the late eighties, since the initial introduction of Region Growing (Garbay et al., 1986; Pavlidis and Liow, 1990; Adams and Bischof, 1994) and Watersheds (Vincent and Soille, 1991). There are different modalities based on the type of the input given by the user. The most popular for a few decades was Active Contours, both parametric (Kass et al., 1988) (Snakes) and non-parametric (Sethian, 1999) (Level Set). Active Contours initially used a rough boundary given by the user and then evolve it over time following the evolution described by Euler-Lagrange equations and collapsing to the desired object's boundary. The method used a data-term making the contour evolving towards the gradient of the image and a regularization term, preventing the contour from deforming too much from its original shape or collapsing completely. The main weakness of contour evolution methods, from the user point of view, is that they require contour initializations near the object of interest and their evolution is slow for large images. They also rely on following gradient maps extracted from the data and thus, tend to have worse performance when the images have very low signal to noise ratio, as they corrupt the gradient maps.

Of course many works on the literature have been proposed to overcome these problems, such as the Geodesic Active Contours (Caselles et al., 1995) that compute minimal surface curves (or geodesics) that improve the convergence when the gradient variations are too large. A Level Set method without reinitialization (Li et al., 2005) was also introduced to avoid reinitialization of the band around the level set (required in previous methods for computational reasons). Chan and Vese (2001) introduced (also extended to in (Vese and Chan, 2002)) a method for the active contour evolution without edges, by maximizing the intensity distribution inside and outside the contour. Shape priors also helped to evolve contours towards a previously known shape (Rousson and Paragios, 2002). And recently Li et al. (2010) introduced a new distance regularized level set with some advantages over other level set methods: 1) it completely eliminates the need for re-initialization, 2) it allows setting of large time-steps to speed-up evolution and 3) they shows computational superiority over other level set methods. Figure 2.3 shows an example extracted form the work of Li et al. (2010), showing the initialization and final evolution of level set curves.

For many reasons that will be discussed later on chapter 8, other forms of image segmentation that gained particular attention after the remarkable publication of Graph Cuts for ND-Image Segmentation (Boykov and Jolly, 2001; Boykov and Funka-Lea, 2006) are of more interest for this thesis. In there, Boykov et al. presented a framework towards image segmentation by using user scribbles as ground truth annotations and formulated

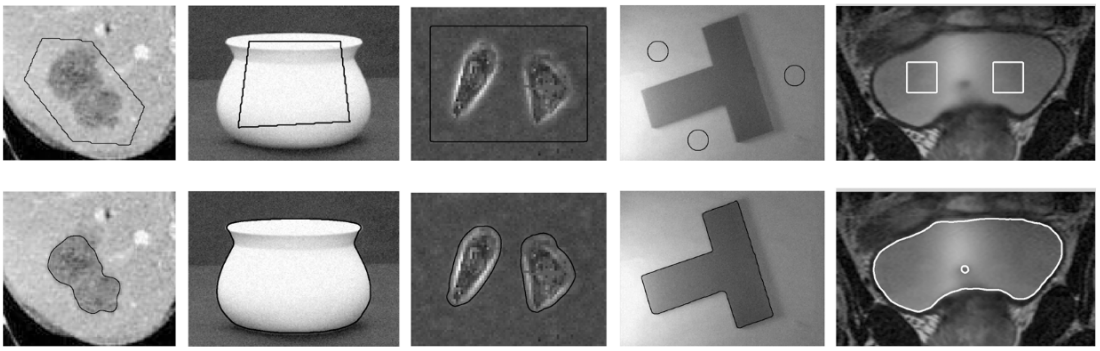


Figure 2.3: Example result image from Level Set evolutions, adapted from Li et al. (2010). The top row shows initialization contours while bottom row shows final results of the contour evolution. It can be seen that some drawbacks of contour methods have been addressed as the contours no longer need to be drawn close to the desired surface and they can also split into multiple objects.

the image segmentation methods as the Maximum A Posteriori (MAP) solution for a Markov Random Field (MRF) problem, where each pixel on an image is a node in the MRF and all the nodes are connected following an 4 (or 8)- neighbourhood pattern. The MAP-MRF was designed to predict the label of each of the nodes between *foreground* and *background* by using probability priors extracted from histograms taken from the user scribbles as unary potentials and a weighted Potts model (Potts, 1952) as a pairwise term; that is, a consistency regularization for neighbouring nodes. Finding a solution to a discrete MRF problem, however, is NP-Hard and very computationally expensive. The success of their approach relied on their previous work (Boykov and Kolmogorov, 2004; Boykov et al., 2001) that showed how to use the computation of maximum flows in a graph to very efficiently approximate the energy minimization on a MRF model. More details, both technical and practical, about Markov Random Fields and their application to image segmentation are available in Appendix A. Figure 2.4 shows an example taken from (Boykov and Funka-Lea, 2006) that illustrates the ability of these algorithms to model the appearance of the foreground and background from simple histograms and segmenting whole objects with a robust MRF-based framework.

Since then, multiple works have been proposed to extend the Graph Cut framework for interactive segmentation, such as Grabcut (Rother et al., 2004), now present in multiple image processing suites such as Photoshop, which included a Expectation-Maximization (EM) iterative refinement for estimating color distributions for foreground and background without the use of user scribbles. Other works, such as (Freedman and Zhang, 2005) added shape priors by using a signed distance function of a previously registered mask of the object of interest to the MRF formulation, making Graph Cuts solution converge towards a certain mask. Other more simple shape priors were also added in (Slabaugh and Unal, 2005), by using a narrowband (similar to early Level Set

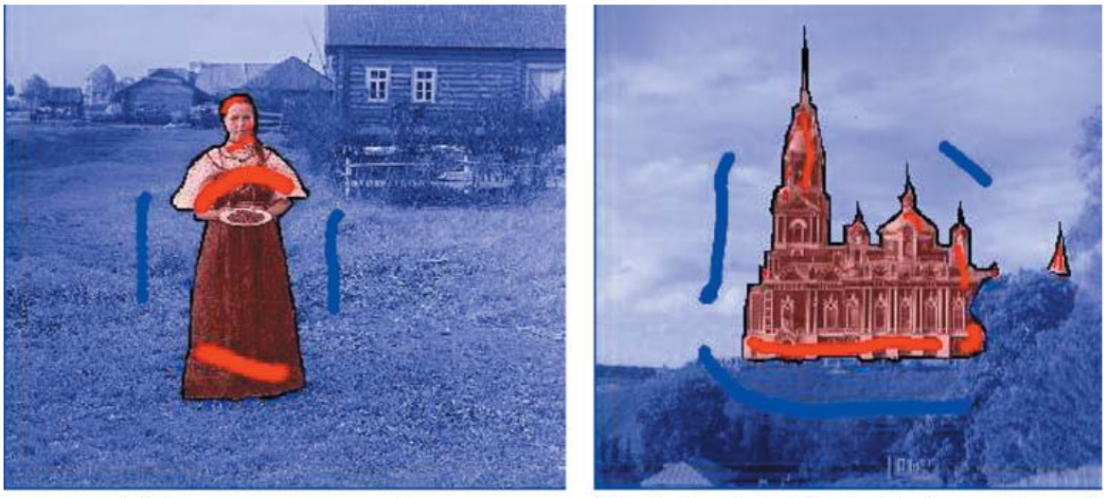


Figure 2.4: Example of image segmentation with user annotation as input using Graph Cut. Images adapted from (Boykov and Funka-Lea, 2006).

method) to limit the Graph Cut optimization to find ellipse-like objects. More complex approaches were introduced towards adding shape priors to Graph Cut include (Malcolm et al., 2007), which used Kernel PCA to model multiple shape masks in a non-linear fashion to better model the variability in the shape of objects of interest. Then the most reliable learnt shape is projected to the object to be segmented as a Bayesian Prior.

Other trends went towards adding connectivity priors to the graph-cut segmentation model (Vicente et al., 2008) by including Dijkstra's shortest path (Dijkstra, 1959) as a connectivity constrain to the optimization. Other similar methods to improve object detection and segmentation with minimal user input included Star Shape priors (Veksler, 2008), which is a generic shape prior to find convex objects that acts as a distance regularizer to prevent the graph cut from exploding, or Geodesic Graph Cuts (Price et al., 2010), which similar to Geodesic Active Contours previously mentioned, added the geodesic distance information to the graph cut minimization to add better localization and segmentation of objects with less sensitivity of where seeds are located.

One of the most recent papers extending Graph Cut is OneCut (Tang et al., 2013), where the authors proposed a new formulation for the GrabCut (Rother et al., 2004) problem that is solved with a single graph-cut, making it not only more efficient, but also more performant. The algorithm, takes the assumption that there is a salient object in the image and aims to extract it, a very interesting approach that can also be extended by using saliency maps extracted with any state-of-the-art saliency detection algorithms (they used the ones from (Perazzi et al., 2012)) for a better and scribble-less segmentation.

Other related works of particular interest to this thesis are the ones of (Santner et al., 2009, 2011) where learning-based approaches are taken to learn discriminative appearance models from user seeds using a Random Forest (Breiman, 2001). The image segmentation was then formulated as Total Variation problem using the convex relaxation of (Pock et al., 2009), that is, a continuous version of the Potts model that can be minimized with Total Variation.

Of course, Graph Cut and Active Contour based approaches are not the only ones in the literature that address the interactive segmentation problem, there are others such as Intelligent Scissors (Mortensen and Barrett, 1998), Lazy Snapping (Li et al., 2004), LiveWire (Falcão et al., 2000) or Total Variation (Unger et al., 2008), and many variants of either of them, but their discussion is out of the scope of this thesis.

2.2 Superpixels

Superpixels, are by definition a disjoint set of partitions formed by groups of similar and adjacent pixels. That is, a given image is partitioned into multiple non-overlapping regions in which pixels share some characteristics (e.g. they are close and have similar color/texture). As shown in (Luengo et al., 2016d), there is an increasing trend to use superpixels as building blocks for many computer vision applications such as image segmentation Ren and Malik (2003), image parsing Tighe and Lazebnik (2010), semantic labelling Kohli et al. (2009), object detection and tracking Wang et al. (2011), or even hierarchical segmentation (Ladickỳ et al., 2014). Superpixel algorithms group pixels into perceptually meaningful regions, which are more aligned with the human visual cognition system. They not only reduce the redundancy and noise effects in the standard pixel grid, but are also especially useful for high computational cost problems, as operating over a superpixel graph reduces dimensionality (and thus computational complexity) by several orders of magnitude. To enable this reduction in resolution to be helpful, there are some generally accepted properties that a superpixel algorithm should offer in order to provide quality results in the subsequent higher-level applications:

- Superpixels should adhere to image boundaries.
- Each superpixel should be contained in a unique higher level object. That's is, a superpixel should not overlap more than one object in the image.
- In most applications superpixels are used as a preprocessing step for further applications; therefore they should be fast to compute and memory efficient.

There are several ways of visualizing superpixels, as shown in Figure 2.5 by splitting an image into 400 superpixels using SLIC (Achanta et al., 2012). The visualization option (a) is not very common, as it shows the underlying labelling but it makes difficult to visually assess the quality of the superpixels. The most common one in the literature is (b), which shows the boundaries of the superpixels overlaid in the image. However, we often prefer (c), which is a reconstructed image by assigning to every pixel in a superpixel the mean color of all the pixels belonging to it. This is, (c) can be viewed as a reconstruction of the image using only superpixels and their mean color, and in many cases gives a very good visual indicator of their quality, as it shows how well the objects are reconstructed by the superpixels and how much detail is preserved. Along the thesis both methods (b) and (c) will be used interchangeably.



(a) (b) (c)

Figure 2.5: Example of superpixel visualization strategies. (a) shows a color labelling where each superpixel is described by a color. (b) shows the superpixel boundaries overlaid on the original image. And (c) reconstructs the original image by using only the superpixels and their mean color.

Let $I : W \times H \to \mathbb{R}^3$ be a color image of width W and height H, with a total number of pixels $N = HW$. Superpixel algorithms expect a desired amount of superpixels K as parameter to find the most representative K groups of pixels within the image. Superpixels are described through a mapping S of shape $W \times H$ from pixels to superpixels

$$S : \{1, \ldots, N\} \to \{1, \ldots, K\}, \tag{2.1}$$

that is, each pixel $i \in \{1 \ldots N\}$ is mapped by an integer to the superpixel it belongs to $k \in \{1 \ldots K\}$. $|S| = K$ then defines the number of superpixels extracted from image I and S_k represents all the pixels assigned to superpixel k. For this over-segmentation to be valid, every pixel has to be assigned to a single superpixel and each superpixel must be formed by adjacent pixels over the image grid, thus, forming its own connected component.

It can be seen that superpixel over-segmentation is an ill-posed problem. The problem of creating a mapping from pixels to a much narrower space of superpixels is subject to an interpretation of what *similarity* and *adjacency* mean, and which one is more important. Through the literature several works have been proposed that choose alternative interpretations of both, resulting in a wide variety of very different algorithms that obtain very different over-segmentations by applying differing criteria.

Overall superpixel algorithms can be separated in three different categories depending on the underlying data representation used to conduct the over-segmentation:

Graph-based split/merge: starting with a fully connected or an unconnected graph, where nodes represent pixels, the goal is to either cut the graph (remove edges) or the opposite, merge nodes together (add edges) to form K connected regions.

Clustering: K seeds, each one representing a superpixel. *Seeds* are distributed along the image plane and iteratively these centroids are updated until they form superpixels.

Moving Boundaries: the over-segmentation is initialized with uniformly distributed superpixel blocks and their boundaries are *moved* until steady-state is achieved.

Figure 2.6 shows a time-line of the most relevant state-of-the-art algorithms that will be discussed in the following subsections.

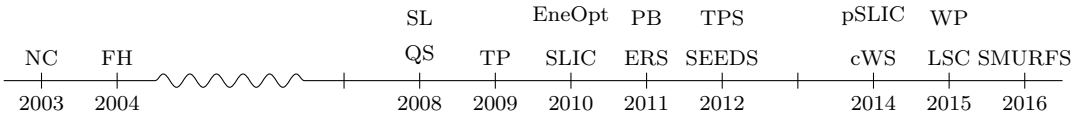


Figure 2.6: Time-line of the most relevant work in superpixel over-segmentation

2.2.1 Normalized Cuts (NC)

As mentioned before, Ren and Malik (2003) were the first to introduce the term *superpixel* as an over-segmentation framework. They used Normalized Cuts originally proposed by Shi and Malik (2000). Normalized Cuts defines the perceptual grouping as a graph partitioning problem. Being $G = (\mathcal{V}, \mathcal{E}, W)$ a non-negative weighted graph where $\mathcal{V}$ are the vertices (or nodes), $\mathcal{E}$ the set edges connecting them and w_{pq} the weight of the edge connecting nodes p and q, a partitioning of G in two disjoint sets $\mathcal{A}, \mathcal{B}$ with $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$ and $\mathcal{A} \cap \mathcal{B} = \emptyset$ is defined as removing the edges connecting both sets. The cost of a cut is then defined as the sum of the weights of the removed edges as,

$$cut(\mathcal{A}, \mathcal{B}) = \sum_{p \in \mathcal{A}, q \in \mathcal{B}} w_{pq}. \tag{2.2}$$

The problem of finding the optimal partition can then be defined as finding the minimum cost cut in the graph. This minimum-cut problem is known to prefer small segments as they cost will be lower, thus, to avoid isolated small regions to be preferred, they proposed a normalized cuts defined as follows:

$$Ncut(\mathcal{A},\mathcal{B}) = \frac{cut(\mathcal{A},\mathcal{B})}{assoc(\mathcal{A},\mathcal{V})} + \frac{cut(\mathcal{A},\mathcal{B})}{assoc(\mathcal{B},\mathcal{V})} \tag{2.3}$$

where $assoc(\mathcal{A},\mathcal{V}) = \sum_{p\in\mathcal{A},q\in\mathcal{V}} w_{pq}$ is the cost associated with every node in $\mathcal{A}$ and every node in the graph G. They show that an optimal two-way partitioning can be found by analysing the Laplacian matrix, $L = D - W$ where D is a diagonal matrix and W is the affinity matrix of the graph G, and discretizing the eigenvector with the second smallest eigenvalue of the generalized eigenvalue problem

$$(D - W)x = \lambda Dx \tag{2.4}$$

This leads to an initial hierarchical partitioning algorithm where a region is iteratively split in two until convergence. A simultaneous k-way multi-cut is then derived from analysing the top k-eigenvectors, discretizing them using k-means and merging connected components until k regions are found.

The quality of the generated superpixels adheres very well to the image boundaries and due to the normalized cut formulation, the over-segmentation tends to produce regions of regular shape. It is however very computationally expensive compared to later methods.

2.2.2 Efficient Graph-based Segmentation (FH)

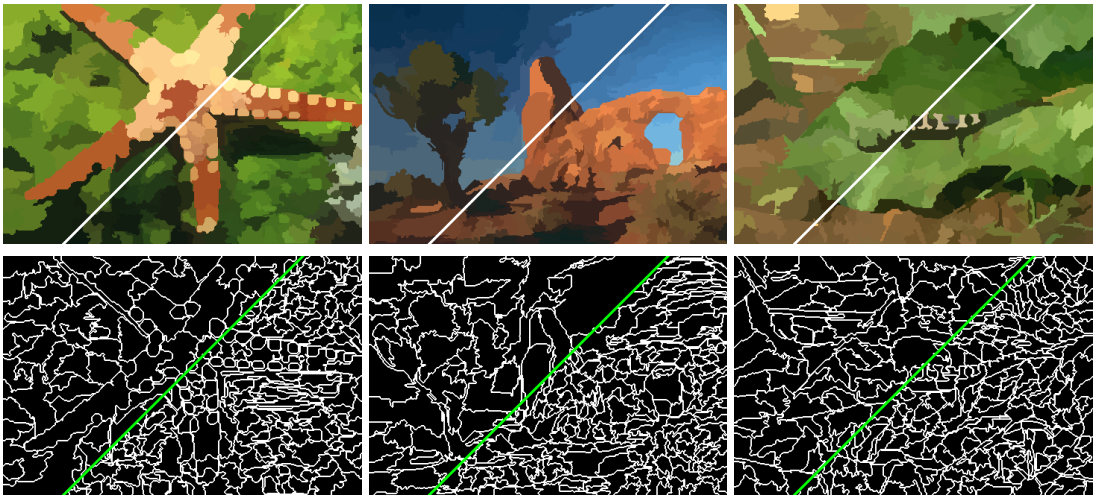


Figure 2.7: Sample over-segmentation of FH with 100 (above diagonal) and 400 (below diagonal) superpixels.

Felzenszwalb and Huttenlocher (2004) proposed a greedy graph-based merging method derived from the greedy Minimum Spanning Tree (MST) of Kruskal (1956) to over-segment an image. The segmentation is formulated as finding the Minimum Spanning Forest (MSF) of a graph $G = (\mathcal{V}, \mathcal{E}, w)$ over the image grid. A MSF is a set of MST, where each MST is a Minimum Cost Tree connecting all the nodes in the tree without loops. Starting with an unconnected graph, where each node is its own component, edges of the graph are first sorted by weight and then, in order, components that satisfy the merging criterion τ will be merged to the same component, with τ defined as follows

$$\tau(i,j) = w_{i,j} < min(C_{int}(A) + \frac{\rho}{|A|}, C_{int}(B) + \frac{\rho}{|B|}). \tag{2.5}$$

Here, $w_{i,j}$ defines the edge cost between pixels i and j, A and B are the components to which i and j belong respectively, $|A|$ is the number of nodes in the component A and $C_{int}(A)$ the internal cost of the component A, defined by the maximum edge cost in the MST of A

$$C_{int}(A) = \max_{(p,q) \in MST(A)} w_{p,q}. \tag{2.6}$$

The controllable parameter ρ in Equation 2.5 controls the size of the objects being targeted, penalizing objects merging components larger than ρ. The use of a Disjoint Set structure[2] allows for a fast lookup to find the MST of a pixel which serves to avoid making cycles during the merging process and making this algorithm one of the quickest superpixel methods to date. The drawback of FH is that it has no control over the number of superpixels generated and the generated regions are very irregular in size and shape. As the merging happens at the pixel level in a greedy manner, it suffers from large under-segmentation in images with no strong boundaries.

2.2.3 Quickshift (QS)

MeanShift (Cheng, 1995; Comaniciu and Meer, 2002) and Quickshift (Vedaldi and Soatto, 2008) are mode seeking algorithms from the family of clustering approaches optimized through gradient ascent. Representing an image as d dimensional data points $x_1, \ldots, x_n \in \mathcal{X} = \mathbb{R}^d$, mode seeking algorithms start from a *Parzen density estimate*:

$$P(x) = \frac{1}{N} \sum_{i=1}^{N} k(x - x_i), \quad x \in \mathbb{R}^d \tag{2.7}$$

where $k(x)$ is a window function (e.g. Gaussian window) around the data point x. Then, a mode seeking algorithm is optimized by iteratively moving each point x_i towards the mode of $P(x)$, following the trajectory of the gradient $\nabla P(y_i(t))$, where $y_i(0) = x_i$.

[2] More details on Disjoint Sets later in section 4.2

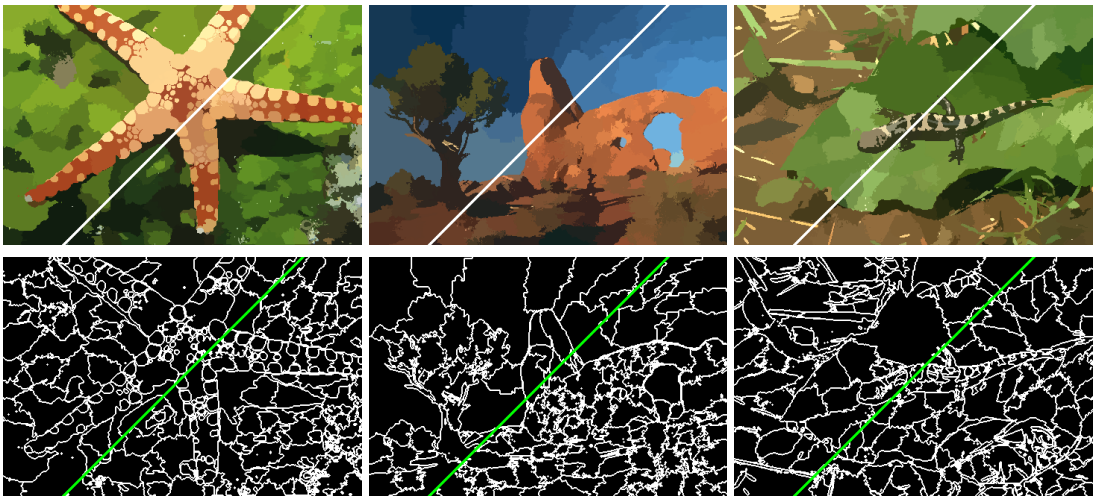


Figure 2.8: Sample over-segmentation of QuickShift with 100 (above diagonal) and 400 (below diagonal) superpixels.

Quickshift replaces the expensive iterative gradient ascent optimization by a pre-computation of the distances between all the data points in the kernelized space (also known as the *kernel trick*),

$$y_i(1) = \underset{j:P_j>P_i}{\operatorname{argmin}} D_{ij}, \quad \text{with,} \quad P_i = \frac{1}{N}\sum_{j=1}^{N}\phi(D_{ij}) \tag{2.8}$$

resulting in a single-step optimization that connects every node in a tree structure T. Modes are then recovered by thresholding large branches in the generated tree T. In order to over-segment images, Quickshift represents pixels as data points in the $\{x, y, l, a, b\}$ space,

$$D_{ij} = \|I_i - I_j\|_2 + \lambda \|p_i - p_j\|_2 \tag{2.9}$$

where λ weights the relative importance of colour I (in the $L*a*b$ colour-space) against position p (x-y coordinates). Additionally, a Gaussian kernel is chosen to transform data-points into a non-linear space ϕ controlled by the width parameter σ and a parameter τ controls the size of the Gaussian window[3] where to look for neighbourhood modes.

2.2.4 Turbo Pixels (TP)

Motivated by the extensive work on Active Contours, both parametric (Snakes (Kass et al., 1988)) and non-parametric (Level Set (Malladi et al., 1995)), Levinshtein et al. (2009) proposed a creation of superpixels by *growing* geometric flows from seeded points. These seed points are separated uniformly along the image plane at distance R, where $R \simeq \sqrt{N/K}$, K is the desired number of superpixels and N the number of pixels in the image. From each of the seeds, a geometric flow defined by the level set of a smooth and

[3]Note the special case that a Gaussian window is a kernel itself.

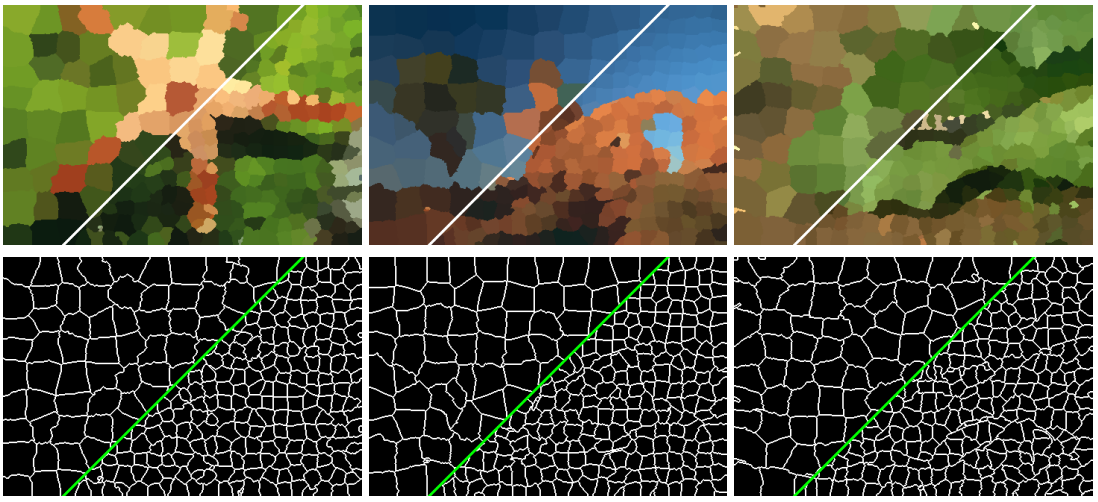


Figure 2.9: Sample over-segmentation of Turbopixels with 100 (above diagonal) and 400 (below diagonal) superpixels.

continuous function $\Psi : \mathbb{R}^2 \times [0, \tau) \rightarrow \mathbb{R}^2$ is evolved over time according to

$$\frac{\partial \Psi}{\partial t} = -v \left\| \nabla \Psi \right\|, \tag{2.10}$$

where t is time and v is the speed updating function. The level set Ψ is defined over the image plane and the zero level set is used to identify superpixel boundaries. The speed function v prevents two level sets (growing from different seed points) to overlap while growing the level sets towards their normals preserving image boundaries (Caselles et al., 1997). The algorithm converges when level sets boundaries collapse and the evolution reaches a steady state.

2.2.5 Simple Linear Iterative Clustering (SLIC)

SLIC Superpixels (Achanta et al., 2012) formulates the superpixel problem as a local k-means clustering in the $\{x, y, l, a, b\}$ feature space. The aim is to split an image of size $N = width \times height$ into K superpixels of size $R \times R$, with $R \simeq \sqrt{N/K}$. To do so, K cluster centres $C_k = \{x_k, y_k, l_k, a_k, b_k\}$ are uniformly distributed along the image plane, spaced R pixels from each other; that is, forming a checkerboard pattern. A local variant of k-means, where each of the cluster centroids C_k searches for pixels in a $2R \times 2R$ window centred at $\{x_k, y_k\}$, is then performed. The distance $D(i, k)$ used to evaluate the membership of a pixel i to a centroid k is defined as the a weighted

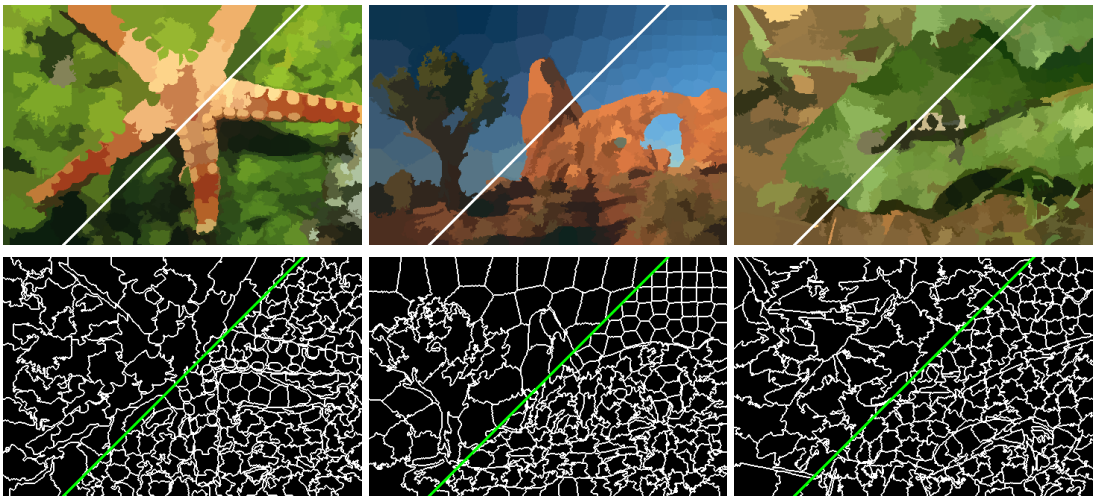


Figure 2.10: Sample over-segmentation of SLIC with 100 (above diagonal) and 400 (below diagonal) superpixels.

Euclidean distance in spatial and colour spaces:

$$\begin{aligned} d_c(i,k) &= \sqrt{(l_i - l_k)^2 + (a_i - a_k)^2 + (b_i - b_k)^2}, \\ d_p(i,k) &= \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}, \\ D(i,k) &= \sqrt{d_c(i,k)^2 + \lambda^2 \left(\frac{d_p(i,k)}{R}\right)^2}, \end{aligned} \tag{2.11}$$

with d_c being the Euclidean distance in the colour space ($\{l, a, b\}$ in the original formulation) and d_p the Euclidean distance in the spatial space. The spatial distance is normalized by R and weighted by λ, the compactness constant. Larger values of λ will prefer spatial to colour similarity in the superpixels, hence, encouraging more regular superpixels, while lower values will encourage superpixels of arbitrary size but similar colour. The modified Expectation-Maximization (EM) algorithm to find optimal clusters is detailed in algorithm 1.

SLIC has been one of the most adopted superpixel algorithms due to its simplicity and reusability. It can be seen from Equation 6.1 that adapting the algorithm to N-Dimensional data or different channels (e.g. Gray-scale or Spectral data) is straightforward. It also is one of the quickest superpixel algorithms as it takes only a few hundred milliseconds to over-segment an image of the BSD dataset, for example. Compactness can be directly controlled by λ which makes easy to adjust the compactness of the superpixels to deal with the subsequent target task.

2.2.6 Superpixels in an Energy Optimization Framework (EneOpt)

Veksler et al. (2010) present a superpixel algorithm formulated as an energy optimization

Algorithm 1: SLIC Pseudocode

```
L = 0_N ;                                              // L: store memberships
repeat
    E = ∞_N ;                                          // E: store minimal distances
    /* Expectation - assign each pixel to most similar superpixel          */
    for each cluster C_k do
        for each pixel i ∈ 2R × 2R around C_k do
            if D(i,k) < E_i then
                E_i = D(i,k);    L_i = k
            end
        end
    end
    /* Maximization - using memberships in L                               */
    UpdateClusterCenters(L)
until convergence or T iterations;
```

problem solved via Graph-cut (Boykov et al., 2001). Many computer vision problems can be reformulated as labelling problems. Representing the image as a weighted graph $G = \{\mathcal{V}, \mathcal{E}, w\}$ where $p \in \mathcal{V}$ represent the set of pixels p and $\mathcal{E}$ represents edges connecting adjacent pixels p, q with weight $w_{p,q}$, the goal is to find the optimal labelling $l \in \mathcal{L}$ for every pixel p. With f_p containing the label l assignment to pixel p, a Markov Random Field (MRF) defines unary psi_p and pairwise $\psi_{p,q}$ potentials (representing constraints) over the labelling set $\boldsymbol{f}$,

$$\min_{\boldsymbol{f}} E(\boldsymbol{f}) = \sum_{p \in \mathcal{V}} \psi_p(f_p) + \lambda \sum_{p,q \in \mathcal{E}} w_{p,q} \cdot \psi_{p,q}(f_p, f_q). \tag{2.12}$$

In order to extract superpixels, a set of overlapping square patches of size $2R \times 2R$ are uniformly distributed along the image plane, each of them representing a superpixel $S_k \in \mathcal{S}$, where $R \simeq \sqrt{N/K}$ with $N = |\mathcal{V}|$ the number of pixels in the image and K the desired amount of superpixels. Initially, to each of the superpixels, its intensity S_i is assigned to be the pixel in the centre of the patch c_k. Each pixel can then belong to one or more of the overlapping patches, so in order to find the optimal superpixel representation the unary potential encodes constraints between each of the pixels and each of the surrounding superpixels,

$$\psi_p(k) = \begin{cases} |I_p - I_{c_k}| & \text{if } p \in S_k \\ \infty & \text{otherwise.} \end{cases} \tag{2.13}$$

Similarly, the pairwise potential encourages adjacent pixels to be assigned to the same superpixel (overlapping patch),

$$\psi_{p,q}(f_p, f_q) = \begin{cases} 1 & \text{if } f_p = f_q \\ 0 & \text{otherwise.} \end{cases} \tag{2.14}$$

The contrast-sensitive weight $w_{p,q}$ in Equation 2.12 is set inversely proportional to the intensity difference of p and q by using a Gaussian Kernel. The optimization scheme attempts then to find the optimal labelling $\boldsymbol{l}^*$ in a multi-label optimization scheme, where each label denotes a superpixel $|\mathcal{L}| = K$ and pixels can be assigned to any of the ones they overlap with. Optimizing Equation 2.12 is NP-hard but can be efficiently approximated (with an approximation factor of 2) by means of α-expansion (Boykov et al., 2001), where each of the binary problems is solved with (Boykov and Kolmogorov, 2004).

2.2.7 Entropy Rate Superpixels (ERS)

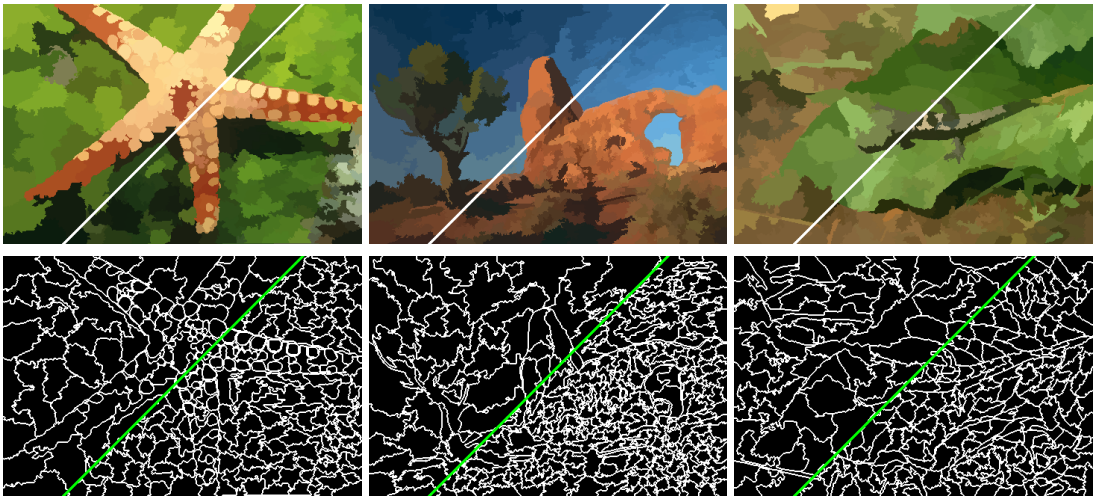


Figure 2.11: Sample over-segmentation of ERS with 100 (above diagonal) and 400 (below diagonal) superpixels.

Liu et al. (2011) present another graph-based algorithm to extract superpixel that uses the entropy rate of a random walk to generate homogeneous regions. Let $G = (\mathcal{V}, \mathcal{E}, w)$ be a non-negative weighted graph where $\mathcal{V}$ are nodes, $\mathcal{E}$ edges and $w_{p,q}$ the similarity weight of the edge connecting nodes p and q. A random walk in a graph (Cover and Thomas, 2012) is defined as the transition probability $p_{i,j}$ from node i to node j,

$$p_{i,j} = \frac{w_{i,j}}{w_i}, \quad with \quad w_i = \sum_k w_{i,k} \tag{2.15}$$

Similarly, the stationary distribution is given by

$$\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N)^T = (\frac{w_1}{w_T}, \ldots, \frac{w_N}{w_T}), \tag{2.16}$$

where $w_T = \sum_i w_i$ is the normalization constant, the sum of all the edges in the graph.

The superpixel problem is then formulated as finding a subset of edges $\mathcal{A} \subseteq \mathcal{E}$ that separate the nodes of the graph G into exactly K connected components, with K the desired amount of superpixels. This graph construction rewrites Equation 2.15 as

$$p_{i,j} = \begin{cases} w_{i,j} & \text{if } i \neq j \text{ and } (i,j) \in \mathcal{A} \\ 0 & \text{if } i \neq j \text{ and } (i,j) \notin \mathcal{A} \\ 1 - \frac{\sum_{j|(i,j)\in\mathcal{A}} w_{i,j}}{w_i} & \text{if } i = j. \end{cases} \tag{2.17}$$

The entropy of a random walk for the subset of edges $\mathcal{A}$ can then be described by

$$H(\mathcal{A}) = -\sum_i \mu_i \sum_j p_{i,j}(\mathcal{A}) \cdot \log(p_{i,j}(\mathcal{A})). \tag{2.18}$$

Last, a balancing term is introduced which encourages clusters (superpixels) of similar sizes:

$$B(\mathcal{A}) = H(Z_\mathcal{A}) - N_\mathcal{A} \quad \text{with} \quad p_{Z_\mathcal{A}}(i) = \frac{|S_i|}{|V|}. \tag{2.19}$$

Here $N_\mathcal{A}$ defines the number of components in $\mathcal{A}$, $|S_i| : i \in \{1, \ldots, N_\mathcal{A}\}$ is the number of edges in each of the components respectively and $Z_\mathcal{A}$ defines then the distribution of cluster memberships. The final superpixel formulation is defined as

$$\begin{array}{ll} \max\limits_{\mathcal{A}} & H(\mathcal{A}) + \lambda B(\mathcal{A}) \\ \text{subject to} & \mathcal{A} \subseteq \mathcal{E} \text{ and } N_\mathcal{A} \geq K. \end{array} \tag{2.20}$$

To solve their formulation they use the lazy greedy algorithm from (Leskovec et al., 2007) that iteratively adds to $\mathcal{A}$ the edge that maximizes the largest gain and keeps a heap to efficiently update the costs.

2.2.8 Superpixels Extracted via Energy-Driven Sampling (SEEDS)

Van den Bergh et al. (2012) describe the superpixel problem as a boundary optimization algorithm. Superpixels are created from the very first iteration by uniformly spacing superpixels over the image plane (creating a chequerboard pattern). Pixels from the boundaries of superpixels are then moved from one superpixel to another until boundaries stabilize. To account for both long-range and short-range updates, SEEDS creates a hierarchy of blocks $l \in \{1, \ldots, L\}$ where pixels at level $l = 1$ are grouped into a 2×2

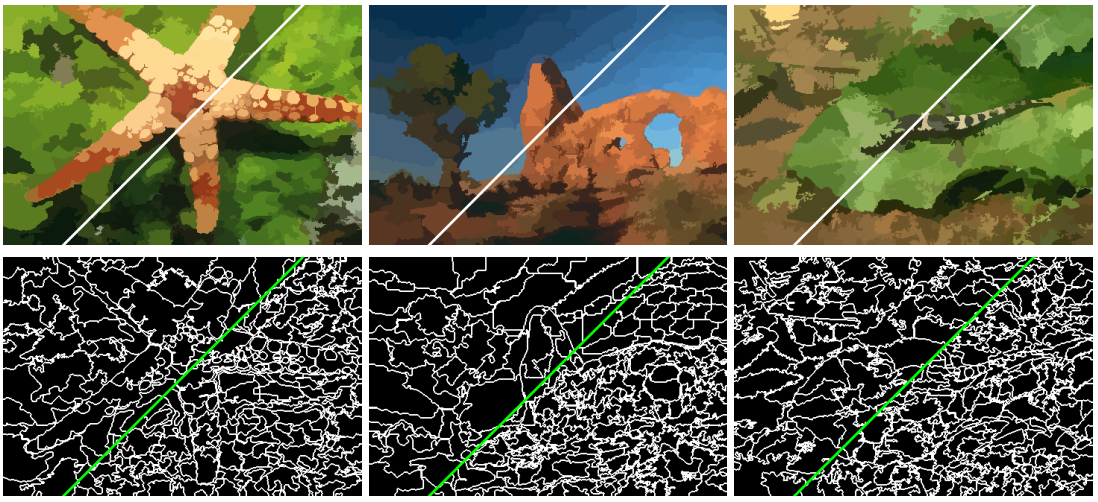


Figure 2.12: Sample over-segmentation of SEEDS with 100 (above diagonal) and 400 (below diagonal) superpixels.

or 3×2 pixel blocks, these blocks are then grouped further in larger 2×2 blocks at level $l = 2$, and grows further until level $l = L$ which determines the current superpixels boundaries, with the amount of blocks in it approximating the number of desired superpixels $|L| \simeq K$. A hill climbing optimization is then proposed to iteratively (and hierarchically) perform block-level and pixel-level updates that maximize the energy described as

$$E(\mathcal{S}) = H(\mathcal{S}) + \lambda B(\mathcal{S}), \tag{2.21}$$

where s is the current superpixel partition, $H(\mathcal{S})$ evaluates the color distribution of each of the superpixels $S_k \in \mathcal{S}$ and $B(\mathcal{S})$ is a prior of the shape of the superpixel which acts as a boundary term, penalizing irregularities in the boundaries of superpixels.

The colour distribution of each superpixel is approximated by discretizing the colour space into colour histograms $\mathcal{H} = \{h_1, \ldots, h_M\}$ in M bins. The colour distribution of superpixel S_k is then described by

$$\psi(S_k) = \sum_{j}^{M} c_{S_k}(j) \quad \text{and} \quad c_{S_k}(j) = \frac{1}{Z} \sum_{p \in S_k} \delta(I(p) \in h_j), \tag{2.22}$$

where $\dot{\delta()}$ evaluates to 1 or 0 depending on whether the inside condition is true or false and, thus, $c_{S_k}(j)$ counts the number of pixels p in the superpixel S_k that fall into the bin h_j and normalizes them by Z. It can be seen then that ψ encourages superpixels with homogeneous colour distribution, as $\psi(S_k) = 1$ if all the pixels $p \in S_k$ fall into the same bin h_j. $H(\mathcal{S})$ then becomes

$$H(\mathcal{S}) = \sum_{S_k \in \mathcal{S}} \psi(S_k) \tag{2.23}$$

The boundary term is defined by centring a $R \times R$ window at each pixel p, denoted as $\mathcal{N}_p$ the set of pixels within the window centred at p. The boundary term then penalizes irregular boundaries by calculating a histogram of different labels (superpixels) at each pixel, and adding them for all the pixels $p \in S_k$:

$$\phi(S_k) = \sum_{p \in S_k} \sum_{j=1}^{K} b_{\mathcal{N}_p}(j) \quad \text{where} \quad b_{\mathcal{N}_p}(j) = \frac{1}{Z} \sum_{q \in \mathcal{N}_p} \delta(q \in S_j). \tag{2.24}$$

Similarly to Equation 2.22, Equation 2.24 counts the number of labels within the neighbourhood of each pixel, within each of the superpixels. In practice, the term $\phi(S_k)$ penalizes superpixels with irregular boundaries, as superpixels with regular boundaries will have less boundary pixels and, thus, the distribution of labels within the superpixel will be focused in a few bins (indicating superpixels). The overall boundary term is defined as

$$B(\mathcal{S}) = \sum_{S_k \in \mathcal{S}} \phi(S_k), \tag{2.25}$$

and Equation 2.21 is optimized through a multi-scale hill climbing algorithm that iteratively performs updates at block and pixel level that improve the energy $E(\mathcal{S})$.

2.2.9 Linear Spectral Clustering superpixels

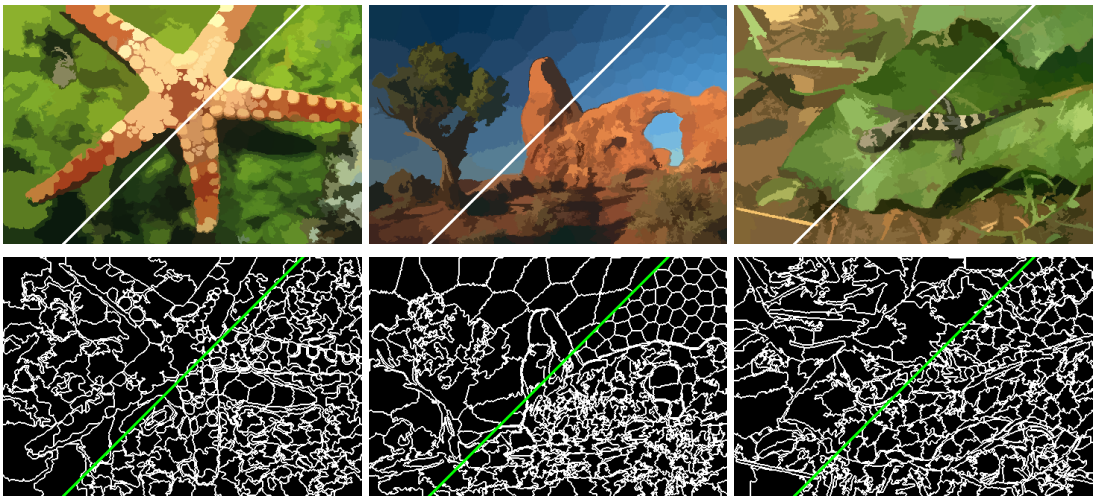


Figure 2.13: Sample over-segmentation of LSC with 100 (above diagonal) and 400 (below diagonal) superpixels.

Linear Spectral Clustering (Li and Chen, 2015) can be seen as an improvement or extension of SLIC. They formulate the superpixel over-segmentation as a clustering problem, but instead of performing int in the lab-xy space, they map the input feature space to a higher dimensional one ϕ. In this high dimensional space (of dimension 10 for lab colour images) the solution of k-means clustering approximates the solution of spectral

clustering in the original image domain. Thus, by performing a simple linear transformation to the input values a much more robust solution is inferred, as normalized cuts usually gives much better results than SLIC (at the expenses of computational time). The feature mapping ϕ for the pixel $\boldsymbol{p} = \{l, a, b, x, y\}$ is defined as follows:

$$\phi(\boldsymbol{p}) = \frac{1}{w(\boldsymbol{p})} \begin{pmatrix} C_c \cos\frac{\pi}{2} l_p, & C_c \sin\frac{\pi}{2}, \\ 2.55C_c \cos\frac{\pi}{2} a_p, & 2.55C_c \sin\frac{\pi}{2} a_p, \\ 2.55C_c \cos\frac{\pi}{2} b_p, & 2.55C_c \sin\frac{\pi}{2} b_p, \\ 2.55C_c \cos\frac{\pi}{2} a_p, & 2.55C_c \sin\frac{\pi}{2} a_p, \\ C_s \cos\frac{\pi}{2} x_p, & C_s \sin\frac{\pi}{2} x_p, \\ C_s \cos\frac{\pi}{2} y_p, & C_s \sin\frac{\pi}{2} y_p, \end{pmatrix} \tag{2.26}$$

with $w(p)$ a normalization factor defined as the weighted average over all the pixels assigned to the same cluster as p. By mapping the input to this high dimensional feature space, the same iterative algorithm than the one of SLIC can be used to create superpixels of higher quality. At each local k-means clustering the weights w for every cluster have to be updated.

2.2.10 Discussion on Superpixels

We have shown that superpixel over-segmentation is a very ill-posed problem. The combination of all possible superpixel over-segmentations for a single image is so broad that it ends up being subject to interpretation. All the algorithms discussed above take different interpretations about how to model a superpixel, and thus, result in very different superpixel segmentations.

The above sections didn't cover all the superpixel works to date, as the multiple different methodologies used to create superpixels would require their own chapter to be properly explained. Thus, the in depth algorithmic over-view above was limited to the most relevant works over the last decade. It included three different selection of algorithms:

Foundation: Normalized Cuts (Shi and Malik, 2000), Efficient Graph-based Segmentation (Felzenszwalb and Huttenlocher, 2004), Quickshift and Meanshift (Vedaldi and Soatto, 2008; Cheng, 1995) and Turbo Pixels (Levinshtein et al., 2009) as the main works that are known as the foundation of the superpixel algorithms (although only Normalized Cuts and Turbo Pixels where applied with superpixels in mind and the others don't offer a practical way of controlling the number of resulting superpixels).

MRF superpixels: Superpixels based on Markov Random Field based energy optimization, Compact Superpixels (eneOpt1) and Constant Intensity Superpixels (eneOpt2), both from the work of Veksler et al. (2010), as the next two chapters (chapter 3 and chapter 4) will introduce two novel superpixel algorithms through MAP-MRF optimization.

Top performant: SLIC (Achanta et al., 2012), SEEDS (Van den Bergh et al., 2012), ERS (Liu et al., 2011) and LSC (Li and Chen, 2015) as the superpixels with the best results on state-of-the-art benchmarks to date.

Other works on superpixels include Pre-emptive SLIC (Neubert and Protzel, 2014) which is an extension over SLIC to improve the performance by controlling the convergence of individual superpixels and preventing them from being recomputed every iteration.

The classic image segmentation algorithm Watershed (Vincent and Soille, 1991) can also be used to create superpixels by instantiating seeds in a regular grid (similar to the initialization on SLIC). Other works such as Compact Watershed (cWS) (Neubert and Protzel, 2014) and WaterPixels (WP) (Machairas et al., 2015) work on extending the standard watershed implementation to adapt it to create better superpixels by enforcing compactness or other long range priors.

Superpixel Lattices (SL) (Moore et al., 2008), Superpixels via Pseudo Boolean Optimization (PB) (Zhang et al., 2011) and Topology Preserved Regular Superpixels (TPS) (Tang et al., 2012) propose different points of view for superpixel creation by sampling horizontal and vertical lines over the image (until they create a grid with the desired amount of superpixels) and each of the vertical and horizontal lines is deformed to follow the boundaries of the image. Although interesting approaches, their performance (both in computational time and accuracy) is far behind the other methods.

3 Hierarchical Piecewise-Constant Superpixels

We have seen in chapter 2 that most of the state-of-the-art algorithms have either a very high computational cost or approximate global functions via greedy or lazy solvers in order to make efficient solutions. Most of these approximated solutions fall in local optima and are not able to properly detect the global properties of the image.

Motivated by the multi-scale boundary updates of SEEDS (Van den Bergh et al., 2012) described at subsection 2.2.8, two novel superpixel algorithms with the same basis will be proposed:

- **H**ierarchical **P**iecewise-**C**onstant **S**uperpixels (HPCS), a graph-based Markov Random Field (MRF) formulation for superpixel extraction that is inherently generalizable to hierarchical constructions, thus, being suitable for the extraction of Super-Region Hierarchies.
- **S**uperpixels from **Mu**lti-Scale **Ref**inement of **S**uper-regions (SMURFS), extends the formulation of HPCS to incorporate cues from an iterative multi-scale refinement with the aim of better capturing both local and global image properties.

HPCS is defined as a single-shot bottom-up approach to generate a hierarchy of over-segmentations (starting from the pixel grid) by merging together elements from the previous level. Although the HPCS is novel in practice and achieves state of the art performance, the idea of extracting a hierarchy of over-segmentations has already been applied in practice, e.g. by stacking mean-shift over-segmentations hierarchically in (Ladickỳ et al., 2014). However, with SMURFS we propose a completely novel approach towards the extraction of hierarchical over-segmentation layers that follows an iterative approach to refine each of the layers. This is, unlike most of the region extraction

algorithms, that follow either a bottom-up (from pixels through merging to regions) or top-down (from regions through splitting to pixels) and thus fail to either capture global or local properties of the image, SMURFS iterates both splitting and merging steps to further refine the over-segmentation hierarchy.

HPCS will be detailed in high detail here in this chapter, while SMURFS will be described fully in chapter 4. Both algorithms create a define a new concept of a **super-region hierarchy**, which will be briefly introduced in the next section, and further extended in chapter 4 with SMURFS. The rest of the chapter will be organized as follows: the next section 3.1 will define the super-region hierarchies, then section 3.2 will describe the basis and related methods on graph splitting that will set the grounds for both our HPCS and SMURFS algorithms, finally, the rest of the sections will describe HPCS, its extensions, and a detailed qualitative and quantitative evaluation.

3.1 Super-Region Hierarchy

A Super-Region Hierarchy is defined as an over-segmentation hierarchy. A Super-region Hierarchy is a set of hierarchical segmentations over the image grid where

- Each level of the hierarchy is composed by grouping similar and adjacent elements from the previous level.
- Over-segmentations of the hierarchy preserve strong boundaries of the image.
- Further layers of the hierarchy represent most relevant objects of the image with fewer representative units.
- The standard pixel grid represents the 0-level of the hierarchy.

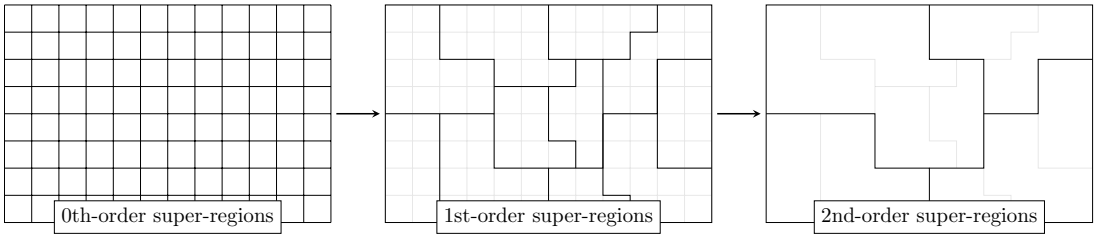


Figure 3.1: An overview of a Super-region Hierarchy. The image discretized as a pixel grid is set to represent the 0th-level of the hierarchy. 1st-order super-regions are then formed by grouping adjacent elements of the level 0 together. Similarly, the 2nd-order super-regions are obtained by grouping elements of the 1st level.

Formally, let a *representative unit* denote the element used to define the data in the current level of the hierarchy (e.g. pixels for level 0 as illustrated in Figure 3.1, superpixels for level 1 as shown in Figure 3.2, etc). Then, a super-region hierarchy is defined as $\mathcal{H} = \{\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_L\}$ of L levels if and only if the level $\mathcal{R}_i$ is formed by forming connected components through grouping elements of the level $\mathcal{R}_{i-1}$, with the exception of the level $\mathcal{R}_0$, which is set to be the standard voxel grid.

Several algorithms in the literature have been proposed to create such over-segmentation hierarchies with the aim of capturing higher-level information from images, such as the Associative Hierarchical Random Fields (Ladickỳ et al., 2014) which uses hierarchical over-segmentations to refine a scene parsing algorithm, Image Segmentation by Cascaded Region Agglomeration (Ren and Shakhnarovich, 2013) which learns a cascade of classifiers to hierarchically merge superpixels or Selective Search (Uijlings et al., 2013) which hierarchically merges super-regions with the aim of detecting objects in the image at multiple scales (all of them will be discussed in depth in later sections). However, most of the algorithms that use or create a hierarchy of over-segmentations optimize these regions in a single direction, this is, regions are created either in a bottom-up (merging) or top-down (splitting) fashion.

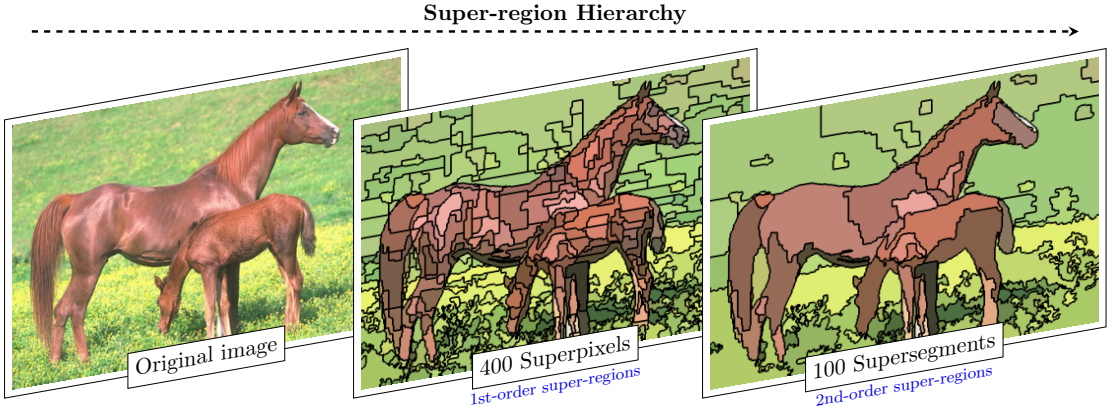


Figure 3.2: Example Super-region Hierarchy on a standard image of the BSD500 dataset. Pixels form the 0th-order super-region level. Superpixels are formed by grouping similar and adjacent pixels. Their inherent boundary adherence allow to represent high level objects of the image with much less representative units. Super-segments, similarly, are formed by grouping similar and adjacent superpixels together. This super-region hierarchy reduces the number of representative units needed to reconstruct an image.

3.2 From Graph Partitioning to Image Segmentation

Spliting regions from $\mathcal{R}_2$ into regions in $\mathcal{R}_1$ can be seen as grouping similar and adjacent pixels that belong to the same region $r \in \mathcal{R}_2$ into new subsets of smaller super-regions $\mathcal{R}_1$. Let $G = (\mathcal{V}, \mathcal{E}, w)$ the weighted graph where nodes $\mathcal{V}$ are pixels within the image and $\mathcal{E}$ are the edges connecting them. Let $\mathcal{A} = \{(p, q) \in \mathcal{E} \,:\, s_{0\to2}(p) = s_{0\to2}(q)\}$, with $\mathcal{A} \subseteq \mathcal{E}$, be a subset of the total edges, where pixels $(p, q) \in \mathcal{E}$ that don't belong to the same region in $\mathcal{R}_2$, are not connected in $\mathcal{A}$. And last, let $G_k \subseteq G$ denote a subgraph of G with nodes $\mathcal{V}_k = \{p \in \mathcal{V} : s_{0\to2}(p) = k\}$ and edges $\mathcal{A}_k = \{(p, q) \in \mathcal{A} : s_{0\to2}(p) = k \wedge s_{0\to2}(q) = k\}$, denoting the subgraph of nodes and edges where every node maps to the higher-level region $\mathcal{R}_2$ through $s_{0\to2}$. Splitting each of the super-regions in $\mathcal{R}_2$ can be seen as removing edges from each of the subgraphs G_k to form K connected regions,

$$split(G) = \sum_{k}^{|\mathcal{R}_2|} split(G_k). \tag{3.1}$$

As seen in subsection 2.2.1, *normalized cuts* (Shi and Malik, 2000) could be used to split a subgraph G_k in two. They also propose a k-way multicut to split the graph into K connected regions by finding the optimal solution to a generalized eigenvalue problem over the normalized Laplacian matrix and then merging if the resultant number of regions is larger than K, however, it is computationally very expensive. Finding optimal cuts in a graph is an active area of research, as graphs are used to represent multiple large-scale problems (e.g. indexing the `www`). A review of graph clustering algorithms can be found in (Schaeffer, 2007), here we will focus on the min-cut/max-flow approximations. It is known that finding the minimum-cut algorithm is equivalent to computing the maximum-flow of a graph (Ford and Fulkerson, 1962). The maximum-flow can be solved efficiently, the first greedy algorithm dating back to 1956 by Ford and Fulkerson (1956, 1962), later improved with the augmented shortest paths (Edmonds and Karp, 1972). Saran and Vazirani (1995) proposed 3 algorithms that hierarchically partition a graph with k cuts, with an approximation factor of 2 to the optimal solution. Flake et al. (2004) proposed a graph-based partition algorithm that splits a graph into connected components with a single cut. To do so, they create an augmented minimum-cut tree (Gomory and Hu, 1961) T_{G_α}, which is a weighted tree that represents the minimum source s to sink t cuts for every (s, t) pair in G_α and has an equivalent flow to G_α, where G_α is an aumgented graph of G with an additional artificial node called *sink* and every node $v \in \mathcal{V}$ is connected to t with constant weight α. Finding the minimum-cut in T_{G_α} will result in a partition of connected components on both T_{G_α} and G. This formulation, based on multi-cuts of the Gomory-Hu Tree are, however, infeasible for large scale vision

problems, as constructing such tree requires $N-1$ max-flow computations, which makes it computationally very expensive.

Back in 1962 Ford and Fulkerson (1962) demonstrated that max-flow in a weighted graph $G = (\mathcal{V}, \mathcal{E})$ is a powerful technique that can be used to solve combinational problems in the form of Maximum a Posteriory of a Markov Random Field (MAP-MRF) ((Boykov et al., 1998) discusses how to approximate MRF models with max-flow algorithms). This combinatorial problem can be written as a general combinatorial optimization (labelling) problem of labels $\boldsymbol{f} = \{f_1, \ldots, f_{|\mathcal{V}|}\}$ with unary constrains ψ and pairwise constrains ϕ

$$E(\boldsymbol{f}) = \sum_{p \in \mathcal{V}} \psi_p(f_p) + \lambda \sum_{(p,q) \in \mathcal{E}} w_{p,q} \cdot \phi_{p,q}(f_p, f_q), \tag{3.2}$$

where $f_p \in \{0, 1\}$ is the label assignment to node p, $\boldsymbol{f}$ is the labelling of al nodes, ψ_p is a data penalty function and $\phi_{p,q}$ is an interaction similarity potential between adjacent nodes, encouraging adjacent nodes to have the same label. Finding the minimum of $E(\boldsymbol{f})$ is then equivalent to finding the maximum flow of G. Boykov and Kolmogorov (2004) proposed a new max-flow algorithm that out-performs most of the max-flow algorithms in practical computer vision applications, despite having higher theoretical upper-bound complexity. These kind of formulations and graph constructions have since been used in numerous occasions to solve a wide variety of vision problems, such as image segmentation or stereo matching(Boykov and Funka-Lea, 2006), color image segmentation (Rother et al., 2004), scene parsing Tighe and Lazebnik (2010), hierarchical parsing (Ladickỳ et al., 2014) and many others.

Efficient binary partition of the graph can then be computed assuming that there is additional knowledge available. In vision it is usual to have additional knowledge about the problem to be solved, as each of the nodes in the graph (e.g. pixels) contain intensity or color properties that can be used to drive the optimal partition. As an example, image binarization with pairwise smoothing potential can be computed by selecting the label space as $\mathcal{L} = \{0, 1\}$ (for black and white respectively) and setting the unary and pairwise potential

$$\psi_p(f_p) \begin{cases} I_p & \text{if } f_p = 0 \\ 255 - I_p & \text{if } f_p = 1 \end{cases}, \quad \text{and} \quad \phi_{p,q}(f_p, f_q) = \begin{cases} 1 & \text{if } f_p = f_q \\ 0 & \text{otherwise} \end{cases}, \tag{3.3}$$

to penalize assigning to label 0 (black) bright pixels and similarly for label 1 (white). The pairwise term adds a penalty if adjacent pixels have different label. Minimizing Equation 3.2 with the potentials in Equation 3.3 will yield a smooth partition (depending on the weight λ).

Similarly, a gray-scale image can be denoised by selecting the set of labels as the gray scale labels $\mathcal{L} = \{0, 1, \ldots, 255\}$ and perform de-noising by penalizing labels for each pixel depending on their initial intensity, and encourage nearby similar pixels to have similar label (gray scale intensity) by adding a contrast-sensitive pairwise potential,

$$\psi_p(f_p) = \|I_p - f_p\|_2^2 \quad \text{and} \quad \phi_{p,q}(f_p, f_q) = |f_p - f_q| \,. \tag{3.4}$$

Here ψ acts as a data fidelity term, penalizing pixels if they change the intensity from their initial state, while ϕ encourages nearby pixels to have similar intensity. Multi-label problems cannot be solved with a single graph-cut as in the image binarization problem; in this special case, however, where the labels are ordered it can be solved with a single max-flow computation by constructing an adequate graph using Ishikawa's graph construction (Ishikawa, 2003). However, when the label-set does not have a meaningful order (e.g. color image restoration) or represent categories (e.g. image segmentation in foreground/background) the problem is intractable, but efficient approximations exist, such as α-expansion (Boykov et al., 2001).

The above MRF-based denoising can be seen as the discrete formulation of the Total Variation (ROF-model) (Rudin et al., 1992) problem,

$$TV(u) = \|u - f\|_2^2 + \lambda \, |\nabla u| \,, \tag{3.5}$$

where f is the original image and u the optimization variable. The first term constrains u to be similar to the orignal image (as in the discrete version of the MRF above) and the second term encourages smoothness in the denoising by penalizing the gradient in the solution u (exactly as the pairwise term of the MRF). The solution and formulation of a TV problem lies in a continuous domain, making it hard to adapt to a multi-label segmentation problems. It has been shown (Unger et al., 2008; Santner et al., 2009) that TV can be used to extract texture from the image, however, only binary segmentation (foreground/background) has been proven to be efficient to solve but require an additional edge map, which is sometimes unreliable. Santner et al. (2010), latter extended by Nieuwenhuis and Cremers (2013), introduced a variational formulation of various computer vision multi-label problems (such us interactive image segmentation or color segmentation), by using the continuous convex relaxation of the discrete Potts potential introduced by Pock et al. (2009); however, running times for a sample 512×512 image using accelerated CUDA code (which requires special hardware) are reported to be about a minute.

3.3 PCS: superpixels from Piece-wise Constant Super-Regions

Following the trends in graph partitioned detailed in the previous section, we found that superpixels can be created by treating an image (in this case, a graph of connected adjacent pixels) by over-denoising the image until homogeneous regions are created.

We formulate region splitting as a graph-based energy minimization problem over a MRF framework. General max-flow and Gomory-Hu tree based graph partitioning solutions have very large computational cost for practical computer vision applications. However, exploiting the known topology of an image and the prior information that superpixels ought to be of similar appearance and composed of adjacent pixels, efficient approximations can be computed. Prior knowledge can be added in the form of representative colours, as adjacent pixels in an image belonging to the same object tend to have similar colour/texture. Thus, let us define a sampling function $\Gamma(\mathcal{R}^i, D^i, k) = \Theta \in \mathbb{R}^{k\times M} = \{\theta_1^i, \ldots, \theta_k^i\}$ that samples k data points (centroids) from a region $\mathcal{R}^i \subseteq I$. Lets assume for now that these k sampled points belong to k high density points in the graph, where all the neighbouring pixels have very similar colours. Simple partitioning of the region $\mathcal{R}^i$ could be done by assigning to each of the data-points $p \in \mathcal{V}$ the label $f_p \in \{1, \cdots, k\}$ that minimizes the Euclidean distance with the corresponding centroids,

$$f_p = \underset{k}{\operatorname{argmin}} \|D_p - \theta_k\|_2^2 \tag{3.6}$$

where D_p is the feature space of the point $p \in \mathcal{V}$ and θ_k is k-th centroid sampled in the same feature space. It can be seen that, with a proper definition of the feature mapping $D_p = \{\mu \cdot x_p, \mu \cdot y_p, l_p, a_p, b_p\}$ (weighted position + colour), this local partitioning of the region $\mathcal{R}^i$ is then very similar to the proposed local k-means clustering in SLIC (Achanta et al., 2012) (Equation 6.1). SLIC, however, uniformly distributes centroids θ_k over the image and generates spurious non-connected pixels assigned to arbitrary cluster centres. In SLIC every pixel is evaluated independently, without consideration of neighbouring pixels, which leads to a local optimal solution that lies near the initialization of the cluster centres (refining their boundaries in practice) and a severe under-segmentation. In order to enforce similar and adjacent pixels to have the same label and to prevent formation of spurious pixels, we formulate the region splitting with a MRF prior

$$E(\mathcal{R}^i, \boldsymbol{f}) = \sum_{p\in\mathcal{V}^i} \left\| D_p^i - \theta_{f_p}^i \right\|_2^2 + \lambda \sum_{(p,q)\in\mathcal{E}^i} w_{p,q} \cdot \left\| \theta_{f_p}^i - \theta_{f_q}^i \right\|_1, \tag{3.7}$$

where $T^i = (\mathcal{V}^i, \mathcal{E}^i, w)$ is the graph that connects pixels in the region $\mathcal{R}^i \subseteq I$. The unary term is identical to the one above, which resembles the expectation of a k-means clustering, and the pairwise term encourages adjacent pixels to be assigned to the same

label by enforcing an L1-similarity constraint over the cluster centres of each of the labels. The parameter λ will control the strength of the pairwise potential, recovering SLIC-like formulation when $\lambda = 0$, generating smaller regions for small positive values and creating large and consistent regions for larger values.

The key differences from the MRF-formulation above and clustering or SLIC-like local k-means methods are as follows:

1. SLIC operates in local overlapping windows uniformly distributed over the whole images, while the MRF-based formulation is constrained to a region.
2. SLIC needs to run an iterative Expectation-Maximization (EM) algorithm to iteratively refine the cluster centres and optimal labelling. With an adequate sampling function Γ, a single iteration is sufficient to find a proper split by minimizing Equation 3.7.
3. Our MRF formulation is based on a graph with connected nodes and edges, thus, is inherently generalizable for hierarchical segmentation or segmentation of non-uniform grids.

Additionally, the above formulation doesn't rely on clusters being uniformly distributed, thus, as will be seen later, different choices for sampling function Γ and feature space D will result in splitting algorithms with interesting properties. The algorithm also has links with image over-denoising. Let $\mathcal{R} = I$ be the whole image as a region with its corresponding weighted graph $G = (\mathcal{V}, \mathcal{R}, w)$, $D_p = \{l, a, b\}$ the feature space of $p \in \mathcal{I}$ as the $L \cdot a \cdot b$ colour-space and $\Gamma = k$-means that extracts k-colours $\{\theta_1, \ldots, \theta_k\}$ from the image, then Equation 3.7 is rewritten as

$$PCS(\boldsymbol{f}) = \sum_{p \in \mathcal{V}} \left\| I_p - \theta_{f_p} \right\|_2^2 + \lambda \sum_{(p,q) \in \mathcal{E}} w_{p,q} \cdot \left\| \theta_{f_p} - \theta_{f_q} \right\|_1 \tag{3.8}$$

where $I_p = \{l, a, b\}$ represents the colour of the pixel p and $\boldsymbol{\Theta} = \{\theta_1, \ldots, \theta_k\}$ are k representative colours sampled from the image. Reconstructing superpixels from the above formulation, however, would create superpixels with very irregular shapes and of non-controllable number and size, as shown in Figure 3.3.b. However, given the number of desired superpixels K, constraints can be imposed on the connected components algorithm to extract the exact amount of superpixels and enforcing similar shape and size on them. Let $S_K^- = 0.5 \cdot \mu_K$ and $S_K^+ = 2 \cdot \mu_K$ be the maximum and minimum size constraints for the target superpixels, where $\mu_K = \frac{|\mathcal{V}|}{K}$ is the average size of the desired number of superpixels. The connected components above is then replaced by a breadth first constrained connected components that stops adding nodes to a component once

it has reached its maximum size S_K^+, which in practice splits large regions in a rhombus pattern (as shown in Figure 3.3.c) generating a new set of components $\mathcal{C} = \{\mathcal{C}_1 \ldots, \mathcal{C}_t\}$ with $t > K$. Last, a graph of connected components is created, mean colour from each component $D_{\mathcal{C}_i}$ is extracted and nearby components $(\mathcal{C}_p, \mathcal{C}_j)$ are joined together by an edge whose weight is the euclidean distance between colours

$$D_{c_i} = \frac{1}{N_{\mathcal{C}_i}} \sum_{p \in \mathcal{C}_i} D_p \quad \text{and} \quad w_{\mathcal{C}_p, \mathcal{C}_q} = \left\| D_{\mathcal{C}_p} - D_{\mathcal{C}_q} \right\|. \tag{3.9}$$

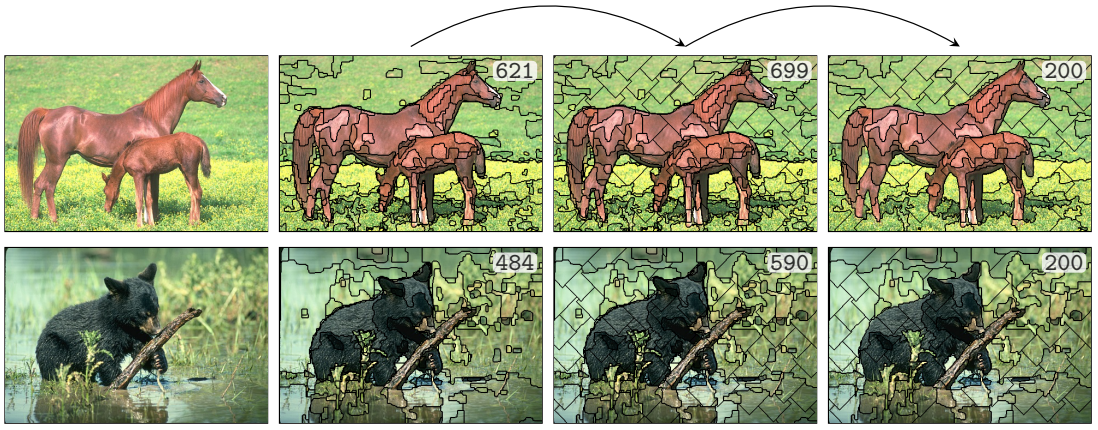
where $N_{\mathcal{C}_i}$ is the number of elements assigned to the component i. Then, with edges ordered in ascending order, small components $\{\mathcal{C}_i : N_{\mathcal{C}_i} < S_K^-\}$ are merged to their most similar neighbour until K components are left, yielding the desired superpixels.

The problem is then similar to a discrete version of the continuous Total Variation (TV, Equation 3.5) in the colour domain, with the main difference that the result of the minimization of E is not a piece-wise smooth image, but the optimal labelling for each node, which can be then reconstructed as a piece-wise constant image, where each of the regions has a discrete colour $\{\theta_1, \ldots, \theta_k\}$. Similar to a TV formulation, the parameter λ will control the strength of the over-denoising, once again, validating our premise that larger values for λ will result in larger and fewer regions. The reconstruction of the optimal labelling of E results in an over-denoised image. We show in (Luengo et al., 2016b) that the optimal labelling energy minimization formulation can be used to create superpixels with further post-processing, named (Hierarchical) Piece-wise Constant Superpixels (H-PCS).

Hierarchical Piece-wise Constant Superpixels (PCS or HPCS) is a novel superpixel algorithm based on the splitting procedure described above. PCS aims to extract superpixels from a given color image. With $G = (\mathcal{V}, \mathcal{E}, W)$ the graph corresponding to the image $\mathcal{I}$, and $\boldsymbol{f}^*$, the optimal labelling of Equation 3.3, superpixels can be extracted by post-processing $\boldsymbol{f}^*$ and identifying the superpixels as connected components of the optimization problem in .

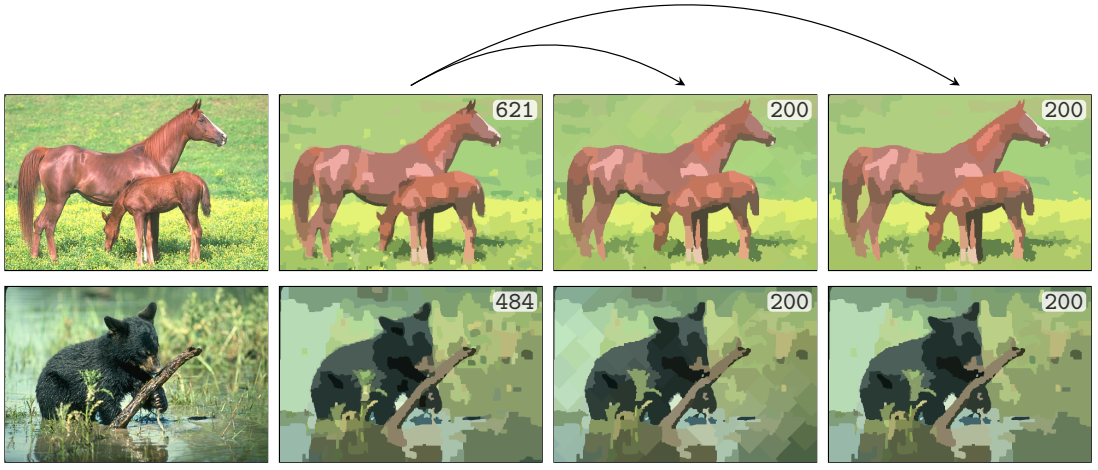
Figure 3.3 shows example results of discrete image over-denoising by treating the whole image as a region, minimizing Equation 3.7 and post-processing the regions. Similarly, Figure 3.4 shows reconstruction of the image using only the mean colour for each superpixel and assigning all the pixels to that colour.

A summary of Piece-wise Constant Superpixels is shown in algorithm 2. It can be seen from the algorithm that both λ and the choice k will vary the result of the algorithm. While larger k will create more connected components, larger λ will further over-smooth them and thus, join some components together. It is the goal then to find a balance between over-denoising strength and the number of centroids selected. In practice, the



(a) Original Image (b) Initial Regions (c) Split Large (d) Merge Small

Figure 3.3: PCS algorithm creates Piece-wise Constant Superpixels from original image (a). (b) shows initial piece-wise constant regions. (c) Shows result that splits large regions from (b) in a simple breadth first connected components algorithm. (d) Merges small regions from (c) to result in the desired amount of 200 *superpixels.*



(a) (b) No constrain (c) Large & Small (d) Only Small

Figure 3.4: Result of the PCS algorithm. (a) shows original image, (b) shows optimization of Equation 3.7 followed by a simple connected components algorithm. (c) Shows results by splitting large regions and merging small ones in (b). (d) Shows similar post-processing results merging only small regions.

pairwise is regularized by $\lambda = 1$ and $k = 16$ colours are sampled from the image, as it has been shown to give a good performance trade-off (more details shown in section 3.5).

Algorithm 2: Piece-wise Constant Superpixels

Input : image: I, num regions: K, smooth: λ, num clusters: k

Output: Superpixels: $\mathcal{R}_1 = s_{0\to 1}$

$\mu_K = \frac{|I|}{K}$, $S_K^- = 0.5 \cdot \mu_K$, $S_K^+ = 2 \cdot \mu_K$;

Obtain graph $G = (\mathcal{V}, \mathcal{E}, w)$ connecting adjacent pixels in I;

Map image I to (l, a, b) feature space $I \to D \in \mathbb{R}^{N\times 3}$; // $N = |I|$

/* Find regions by over-denoising */

Sample k centroids: $\mathbf{\Theta} = \Gamma(G, k) = \{\theta_1, \dots \theta_k\}$; // $\Gamma = k$-means over D

Obtain optimal labelling $\boldsymbol{f}^*$ by minimizing

$$E(\boldsymbol{f}) = \sum_{p\in\mathcal{V}} \left\| D_p - \theta_{f_p} \right\| + \lambda \sum_{(p,q)\in\mathcal{E}} w_{p,q} \cdot \left| \theta_{f_p} - \theta_{f_q} \right| ;$$

$\mathcal{C} = \{\mathcal{C}_1 \dots, \mathcal{C}_t\}$ = Find connected components of maximum size S_K^+ ; // t > K

/* Extract component properties */

Find graph of connected components $T = (\mathcal{C}, \mathcal{E}_\mathcal{C}, w)$;

Extract mean colour and edge weights for each component

$$D_{c_i} = \frac{1}{N_{\mathcal{C}_i}} \sum_{p\in\mathcal{C}_i} D_p \quad \text{and} \quad w_{\mathcal{C}_p,\mathcal{C}_q} = \left\| D_{\mathcal{C}_p} - D_{\mathcal{C}_q} \right\|_2 ;$$

Sort Edges $(\mathcal{C}_p, \mathcal{C}_q) \in \mathcal{E}_\mathcal{C}$ by weight;

$\mathcal{R} = \mathcal{C}, i = 0$;

/* Merge small regions */

while $i < |\mathcal{E}_\mathcal{C}|$ *and* $|\mathcal{C}| > K$ **do**

  $\mathcal{C}_p, \mathcal{C}_q = \mathcal{E}_\mathcal{C}(i)$; // Next edge in increasing weight

  if $|\mathcal{C}_p| < S_K^-$ *or* $|C_q| < S_K^-$ **then**

    $merge(\mathcal{C}_p, \mathcal{C}_q)$

  end

  $i++$;

end

3.4 Hierarchical generalization

As the PCS framework relies on a Graph as a data structure, and doesn't have any imposed grid constraint related to the image domain, it is easily generalizable to an arbitrary graph topology. This is specially useful for building hierarchies, as any layer of an over-segmentation can be modelled as a graph of connected regions, and thus, serve as input for further layers.

Let us then define Hierarchical Piece-wise Constant Super-regions (HPCS) as a novel set of hierarchical over-segmentation layers, where the generalized PCS algorithm is applied recursively. For that, we first generalize the algorithm first defined on algorithm 2 by considering the number of desired number of superpixels for the first layer ($K = 1000$

by default), the number of layers ($L = 2$ by default) and the constraint ratio for super-regions in further layers ($r = 0.5$). With the initial image I transformed into a feature space X (by default X refers to the image in a normalized $L{\cdot}a{\cdot}b$ space), the first iteration of PCS uses a 4-connected Graph representation of the image to extract superpixels from it. The PCS algorithm returns a Region Adjacency Graph (RAG) for the connected components found, along with the new feature representation of each of the components. Thus, PCS can iteratively be applied to its own output, as the output is a RAG of the input graph, thus, creating in practice a hierarchy of segmentations.

Algorithm 3: HPCS Algorithm

Function `HPCS`(X, Γ, M, L)**:**
  $G_0, D_0 \leftarrow$ `GridGraph`(X)
  for $i = 1 \dots L$ **do**
    $G_i, D_i \leftarrow$ `PCS`$(\Gamma, G_{i-1}, D_{i-1}, M)$
    $M \leftarrow M \cdot r$
  end
  return G, D

Algorithm 4: Generalized PCS

Function `PCS`(Γ, G, D, M)**:**
  $\boldsymbol{\Theta} \leftarrow$ Sample k centres from $\Gamma(G, D)$
  $\boldsymbol{f} \leftarrow$ Minimize Equation 3.7
  $\mathcal{C} \leftarrow$ Process M components from $\boldsymbol{f}$
  $T \leftarrow$ Extract RAG of $\mathcal{C}$
  $Q \leftarrow$ Descriptor of components in $\mathcal{C}$
  return T, Q

With G_i a graph that connects nodes at level i and D_i the corresponding feature representation of each of its nodes, the algorithm hierarchically splits a graph into less components. The pseudo code of the HPCS algorithm, described in algorithm 3 shows its relative simplicity and natural hierarchical generalization. Alternating the sampling function and feature mapping of the algorithm will have a strong influence in the shape and quality of the extracted regions. This will allow enforcing different properties on the superpixels, such as shape compactness (regularity) or better boundary adherence, or even alternating between uniform or quick sampling and non-local sampling of centres, resulting in a trade-off between quality and computational complexity.

Additionally, due to the Graph based generalized nature of our algorithm, HPCS can be applied on top of an already existing over-segmentation to further reduce the number of regions while preserving high-level object boundaries. Figure 3.5 shows how HPCS can be used to further refine both HPCS itself and SLIC (Achanta et al., 2012) over-segmentations. For HPCS the algorithm runs with $M = 200$ and $L = 2$, for SLIC, after first superpixels have been extracted, RAG and its corresponding region description (as the mean color of all the pixels inside each superpixel) is extracted from them and then PCS is applied. It can be seen that a single iteration of PCS is sufficient to reduce the number of regions, while still maintaining the robust region-based representation of the image.

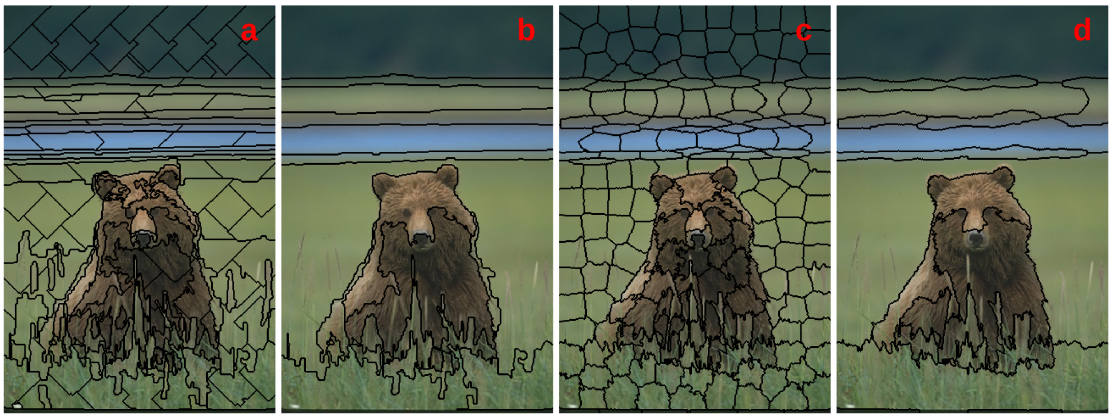


Figure 3.5: HPCS as a post-processing of PCS and SLIC superpixels. (a) our HPCS algorithm generating 200 superpixels (1st-order super-regions), (b) our HPCS over-segmentation over (a) with 17 resulting 2nd-order super-regions, (c) SLIC superpixel over-segmentation with 200 superpixels and (d) our HPCS over-segmentation over (c) resulting in 22 2nd-order super-regions. By applying our algorithm as a post-processing step the image can be represented with less superpixels.

3.5 Effect of parameters of HPCS

The main parameters of the algorithm are the number of desired superpixels M, the smoothness constant for the partition step λ, the number of clusters to be sampled during the split step K and the scale μ that controls how similar regions have to be in appearance and size to be merged. Sensible defaults are listed in Table 3.1 for HPCS:

Parameter	**Description**	**HPCS**
λ	Controls strength of the denoising	0.5 [0.01, 5]
K	Number of centroids sampled per region	16 [2, 100]
μ	Ratio for the size of the regions to be merged (proportional to the average size N/M)	0.1 [0.01, 2]

Table 3.1: Default tunable parameters for HPCS. N indicates the total number of pixels in an image while M the desired number of superpixels. Parameter μ only applies for HPCS if used with MSF-based post-processing. Brackets indicate expected parameter range.

3.5.1 Controlling the strength of the pairwise splitting term, λ

The parameter λ is one of the key parameters of HPCS. It controls the strength of the pairwise term in the MRF optimization. Larger values of λ will encourage all nearby nodes of the graph to be assigned to the same cluster, generating fewer resulting regions

and with too large values of λ assigning all the nodes of the graph to the same cluster forming a unique set. Small values of λ, conversely, will form more but smaller sub-regions within the region to be split, with $\lambda = 0$ approximating the maximization step of a k-means. Figure 3.6 and Figure 3.7 show the result of selecting different λ for HPCS.

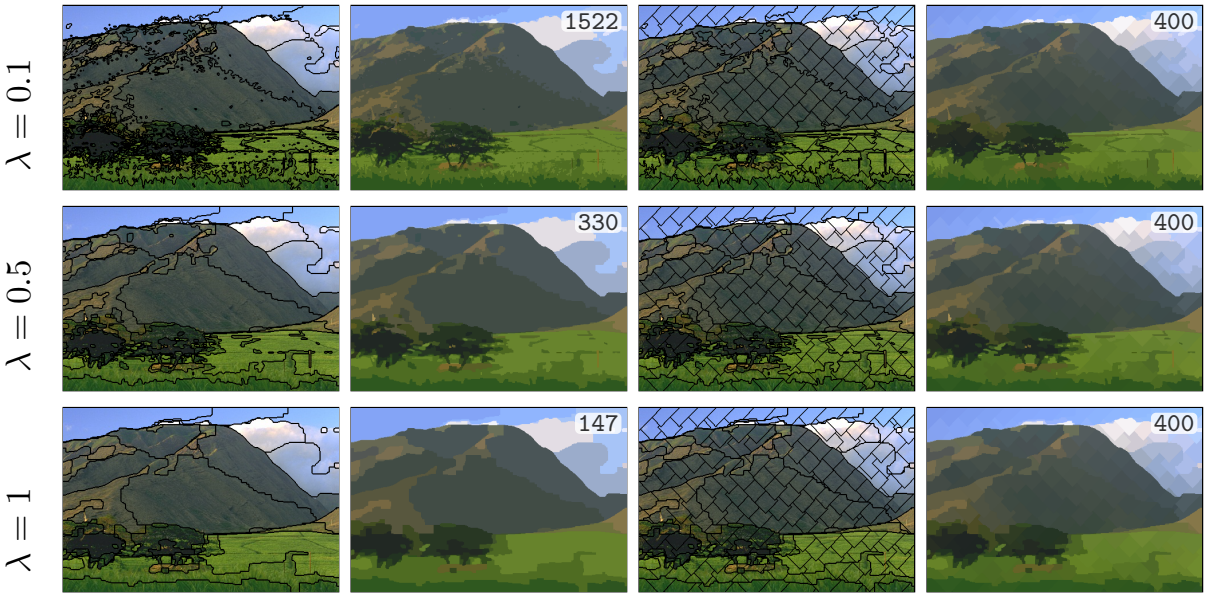


Figure 3.6: Result of applying HPCS with different λ values to a texture-less image. First two columns show initial results after partitioning the image, last two columns show final results after post-processing. It can be seen that low λ values generate much detailed superpixel segmentation while preserving all the image boundaries. Large values, instead, tend to over-denoises the image too much, making it miss some important image boundaries.

Figure 3.6 shows the results of applying HPCS to an image with large piece-wise smooth areas with little texture, except for the leaves and trees. Figure 3.7 instead, shows the result of applying HPCS in a highly textural image. Both figures show results for different values of λ over the same image in order to give insight onto how λ behaves for different kind of images. It can be seen that for non-textural images, small $\lambda = 0.1$ values tend to generate smaller but more regions that better represent the image, while large $\lambda = 1$ values over-denoise the image too much, generating too few regions and missing some important image contours. Conversely, on highly textural images, small $\lambda = 0.1$ generates too many small regions for practical purposes that end up with a poor superpixel representation, as result of the image having too much *textural noise* and the pairwise potential not having enough strength. Larger $\lambda = 1$ works better in this kind of images, as the pairwise term deals better with local noise, over-denoising the image in practice resulting in larger piece-wise constant regions while maintaining important image contours. It can be seen that in both kind of images, $\lambda = 0.5$ (as default value in the algorithm) gives good results as the pairwise term gains enough

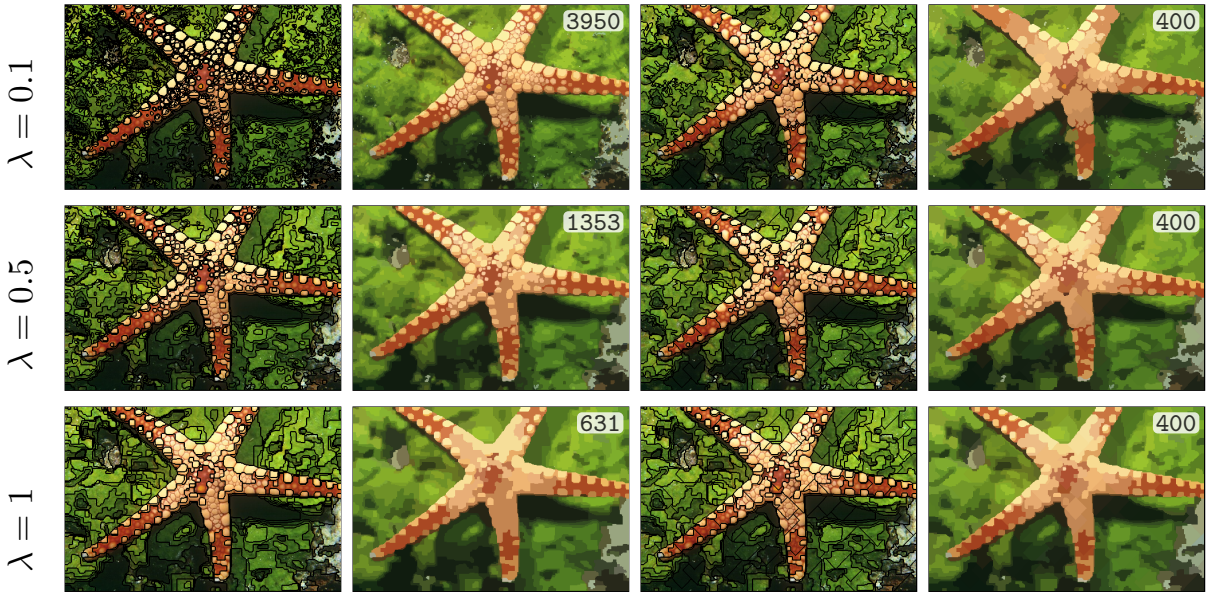


Figure 3.7: Result of applying HPCS with different λ values to a highly textured image. First two columns show initial results after partitioning the image, last two columns show final results after post-processing. It can be seen that low λ values initially generate too many regions that are difficult to further reduce to just 400*. Large values, instead, tend to remove textural noise from the image allowing for a much coarser segmentation while still retrieving image contours properly.*

strength to remove textural features, but not enough to remove important contours in texture-less images. Different values for λ could determine the usability of HPCS for different applications. For example, if the subsequent application is related to object detection or object recognition, a higher value of $\lambda = 1$ might give better result, as the superpixels will have a more smooth/homogeneous shape and thus be easier to use for object training purposes. If the aim is to use superpixels for image segmentation, having high detail is crucial and thus, lower lambda $\lambda = 0.1$ is likely to give more detailed segmentation results.

3.5.2 Controlling the number of centroids to be sampled

The number of centroids sampled have similar effect during the splitting than the one of λ, just inversely proportional. This is, larger values of K will sample more centroids per region, generating more smaller regions after the split is applied. Smaller values instead, will generate larger blobs. It is also to note that K and λ counter act each other, which makes finding a balance between the two important, but not essential since for a fixed K a range of values of λ will give a wide variety of possible solutions to the current image,

each suitable for different applications. Thus, understanding how K behaves and fixing it to a certain number is essential to use HPCS properly.



Figure 3.8: Result of applying HPCS with different K values to an image. First two columns show initial results after partitioning the image, last two columns show final results after post-processing. It can be seen that low K values generate much coarser superpixel segmentation while preserving very strong image boundaries. Large values, instead, tend recover much finer details that also better preserve boundaries.

As shown in Figure 3.8 and Figure 3.9 K is inversely proportional to λ, however, unlike λ, K correlates directly with the complexity of the optimization solution, as in the worst case scenario K maxflow iterations (one per each labe in α-expansion) are needed for each optimization iteration to solve the energy in Equation 3.3, thus, K increases the computational complexity in practice. $K = 16$ is set to default in HPCS as it offers a good trade-off between result accuracy and computational time.

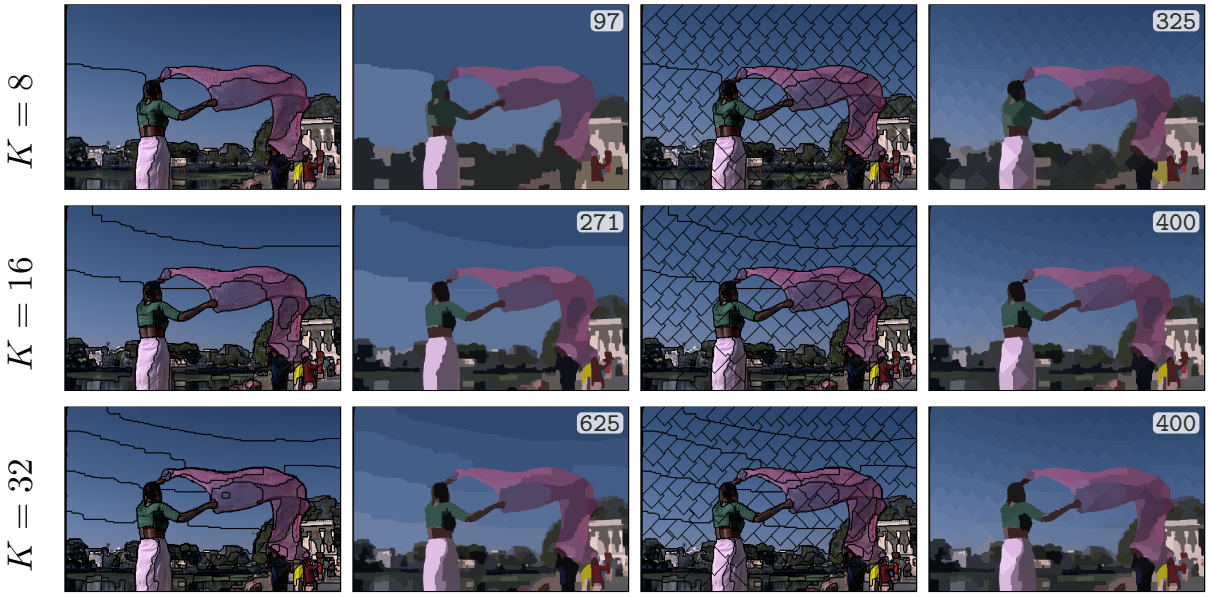


Figure 3.9: Result of applying HPCS with different K values to an image. First two columns show initial results after partitioning the image, last two columns show final results after post-processing. It can be seen that low K values initially generate too few and large regions that hardly recover details from the people on the lower part of the image. Large values, instead, recover much more detail from the image.

3.6 Experimental Comparison of HPCS with state of the art

For both qualitative and quantitative comparison of superpixel algorithms the Berkeley Segmentation Dataset (BSD500) (Arbelaez et al., 2011) is used. BSD500 consists of 500 colour images separated in 200/100/200 sets for training, validation and testing respectively. Images are all of size 481×321 for landscapes and 321×481 for portrait images. All the images have been segmented by up to 5 human annotators. It is the standard dataset used for contour detection and over-segmentation benchmarks. Figure 3.10 shows example images from BSD500 with their corresponding annotations.

3.6.1 Qualitative Evaluation

In this section we will qualitatively compare HPCS algorithm to other state of the art methods. Most of the algorithms are evaluated against SLIC (Achanta et al., 2012), ERS (Liu et al., 2011), SEEDS (Van den Bergh et al., 2012) and LSC (Li and Chen, 2015) as they are not only the latest but also the most performant superpixels from the state of the art at the moment of writing this thesis.

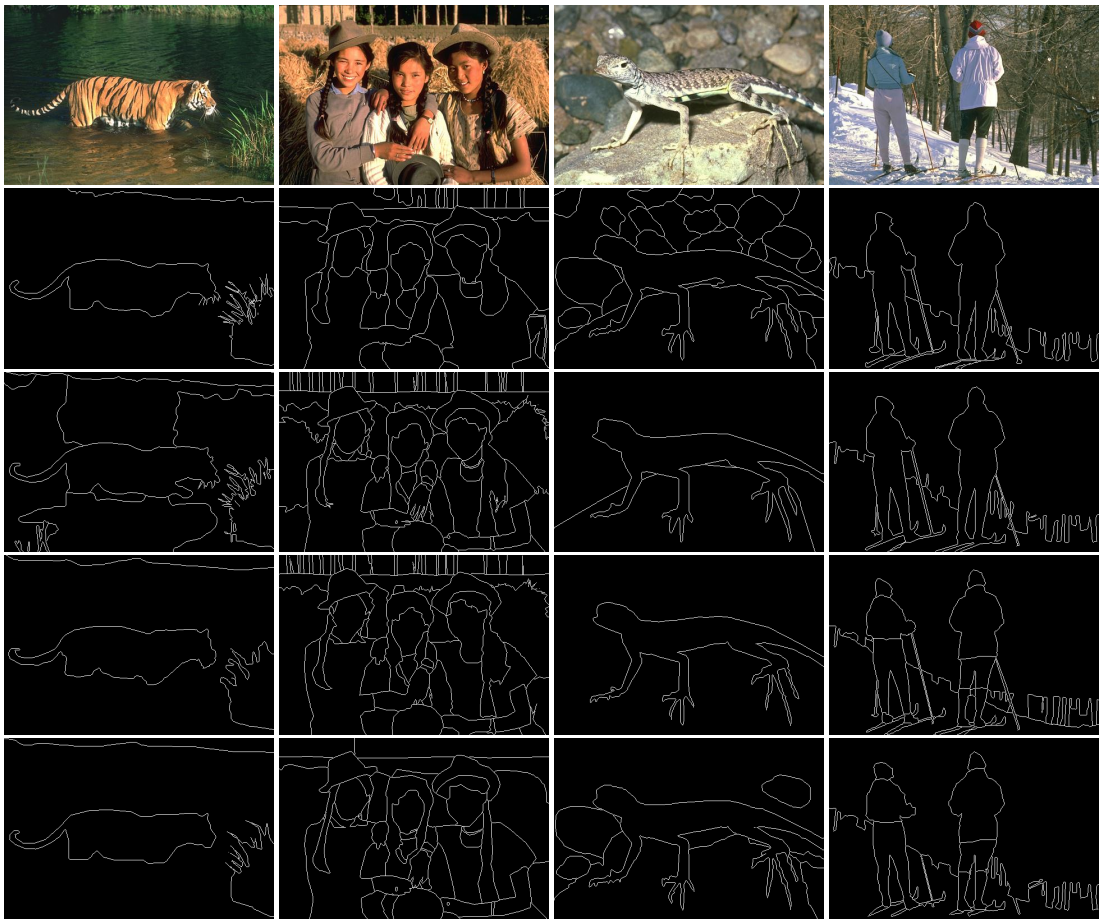


Figure 3.10: Example of four images of BSD500 dataset. Showing not only the large variance in images but also in detail of ground truth annotations.

Figure 3.12, Figure 3.13 and Figure 3.14 show qualitative comparison of HPCS to other state of the art superpixels for $M = 400$ superpixels. Figure 3.12, Figure 3.13 contain images of single figures, this is, salient objects with either cluttered or plain background. It can be seen that the performance is very similar to the other state of the art algorithms and figures are extracted satisfactorily, generating a lot of regular superpixels in the background where attention to detail is not needed. The kind of boundaries that HPCS generates is somewhere between SLIC and LSC (that generate compact superpixels) and SEED and ERS, whose boundaries are chaotic in noisy environments. HPCS generates regulat superpixels in background or texture-less areas but irregular superpixels in complex areas. Figure 3.14 on the other hand shows images with mainly landscapes or mostly background images, ant it can be seen that HPCS creates much more regular superpixels, as most of the areas don't require irregular superpixels to follow complex boundaries.

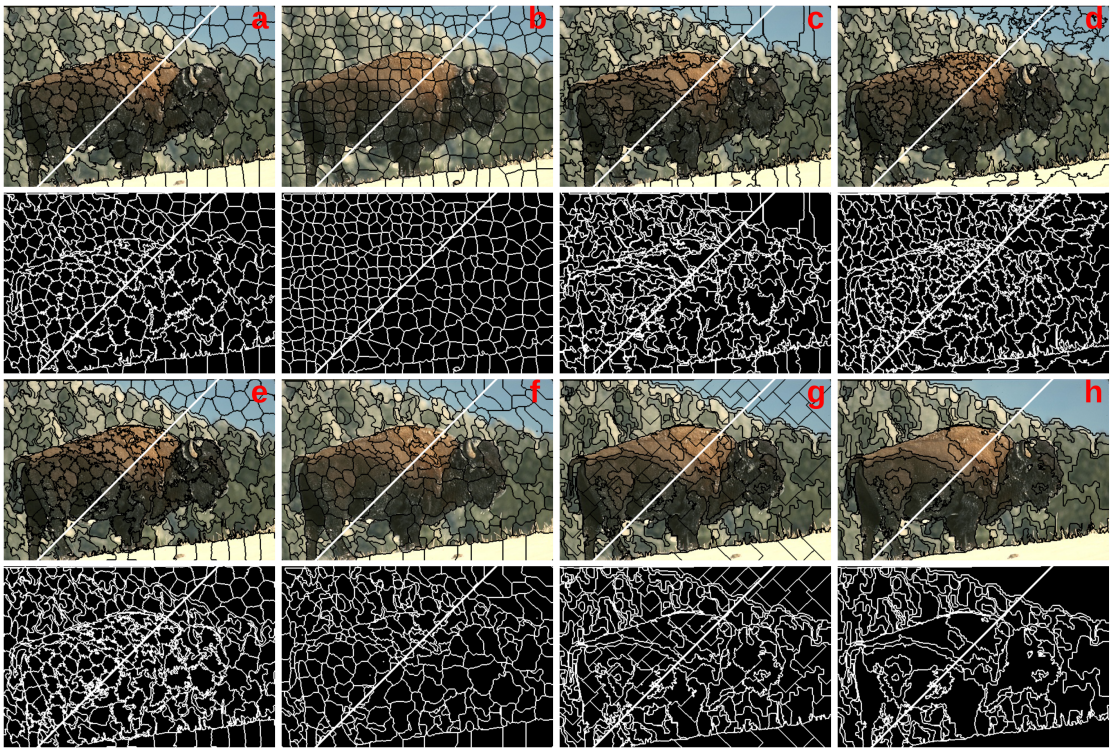


Figure 3.11: Comparison of segmentation algorithms with their correspondent boundary map with 400/200 superpixels. (a) SLIC, (b) Turbopixels, (c) SEEDS, (d) ERS, (e) LSC, (f) EneOpt1, (g) Ours (HPCS) and (H) Ours (HPCS) without enforcing the maximum size s^+ of superpixels (yielding less amount of superpixels)

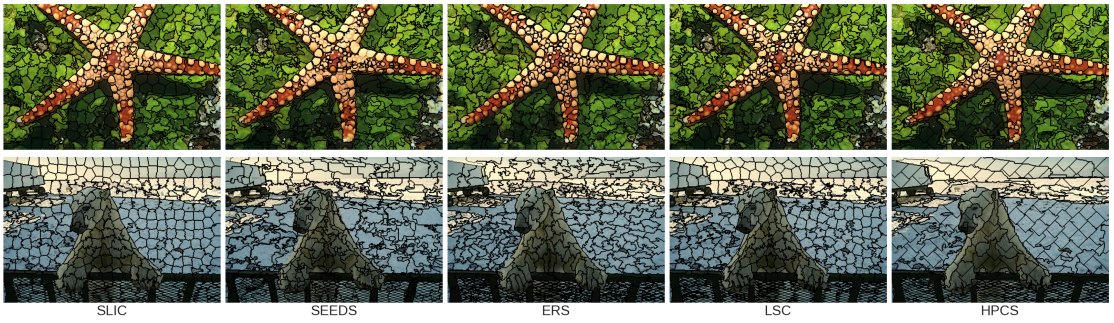


Figure 3.12: Overview and comparison of PCS superpixels to state of the art with $M = 400$ superpixels.

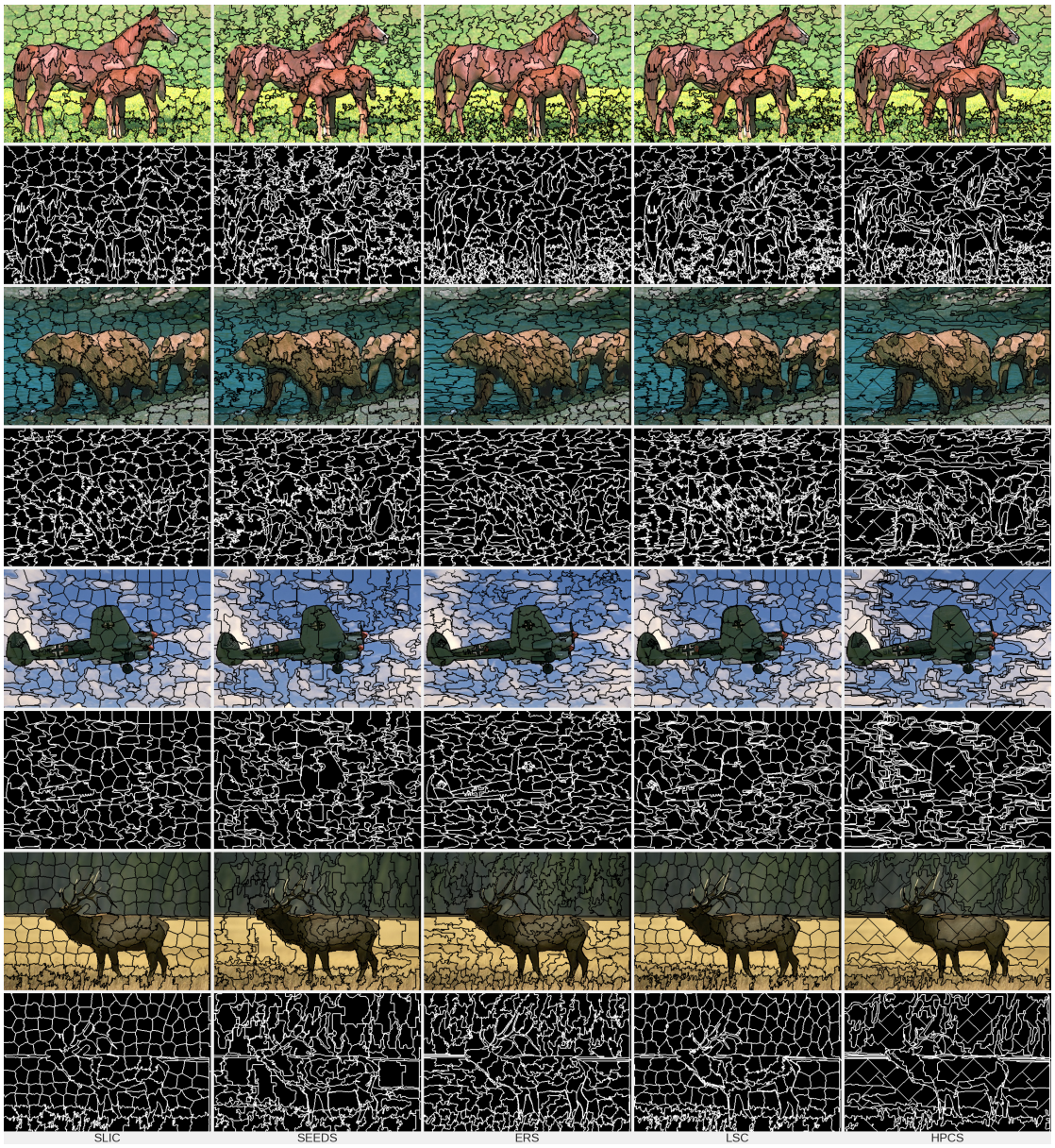


Figure 3.13: Overview and comparison of PCS superpixels to state of the art with $M = 400$ superpixels.

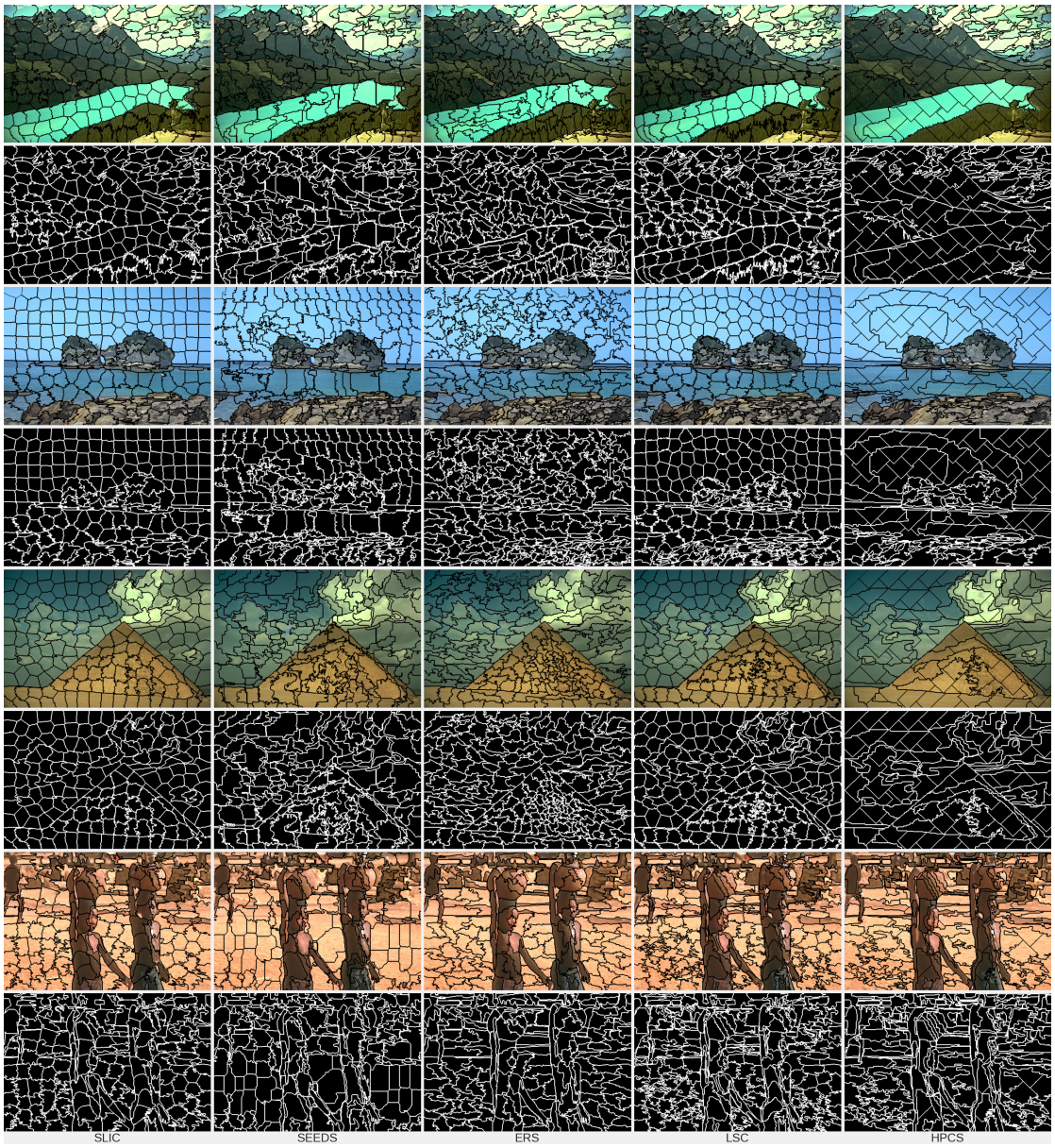


Figure 3.14: Overview and comparison of PCS superpixels to state of the art with $M = 400$ superpixels.

3.6.2 Quantitative Evaluation

We also compared HPCS algorithm using the standard metrics:

Boundary recall (BR) (Liu et al., 2011): measures the percentage of the natural boundaries overlapped by the superpixel boundaries. It is computed as:

$$BR_{\mathcal{G}}(\mathcal{S}) = \frac{\sum_{p \in \delta\mathcal{G}} I[\min_{q \in \delta\mathcal{S}} \|p - q\| < \epsilon]}{|\delta\mathcal{G}|} \tag{3.10}$$

where $\delta\mathcal{G}$ and $\delta\mathcal{S}$ denote union sets of ground truth boundaries and superpixel boundaries respectively and I returns 1 if the criteria is met. It evaluates the ratio of ground truth boundaries that lay in within ϵ pixel distance to the nearest superpixel boundary. As (Liu et al., 2011), we've set $\epsilon = 2$.

Corrected under-segmentation error (CUE) (Van den Bergh et al., 2012): evaluates the error of assigning one superpixel to more than one ground truth labels. Each superpixel is assigned to the largest ground truth match, and is penalized by 'pixel leak'. It is computed as:

$$CUE(\mathcal{S}) = \frac{\sum_k |\mathcal{S}_k - \mathcal{G}_{max}(\mathcal{S}_k)|}{|\mathcal{G}|} \quad \text{with} \quad \mathcal{G}_{max}(\mathcal{S}_k) = \arg\max_i |\mathcal{S}_k \cap \mathcal{G}_i| \tag{3.11}$$

where $\mathcal{S}$ and $\mathcal{G}$ indicate superpixel and ground truth segments.

Achievable segmentation accuracy (ASA) (Liu et al., 2011): is an upper-bound measure that gives the maximum possible segmentation accuracy when segmenting the image using superpixels. It is defined as:

$$ASA(\mathcal{S}) = \frac{\sum_k \max_i |\mathcal{S}_k \cap \mathcal{G}_i|}{|\mathcal{G}|} \tag{3.12}$$

Figure 3.15 show quantitative evaluation of HPCS as superpixel algorithms. HPCS is benchmarked against related and state-of-the-art algorithms in the BSD500 dataset. Average BR, CUE and ASA per image is first calculated (as every image has 5 different ground truths) and then the final value is averaged across the whole dataset. HPCS is similar in performance to the top 2 of the state of the art. Interestingly, for small number of superpixels HPCS does better than the state of the art, as $K = 16$ colors seem to extract the import top object of the image. However, as the number of superpixels goes up the quality of HPCS goes to an steady state (still better than most of the state of the art) but LSC and ERS outperform in every metric. This is due to the ability of other superpixels to focus better on local texture as size of the superpixels decreases, as the solution of the HPCS problem will always be a global formulation.

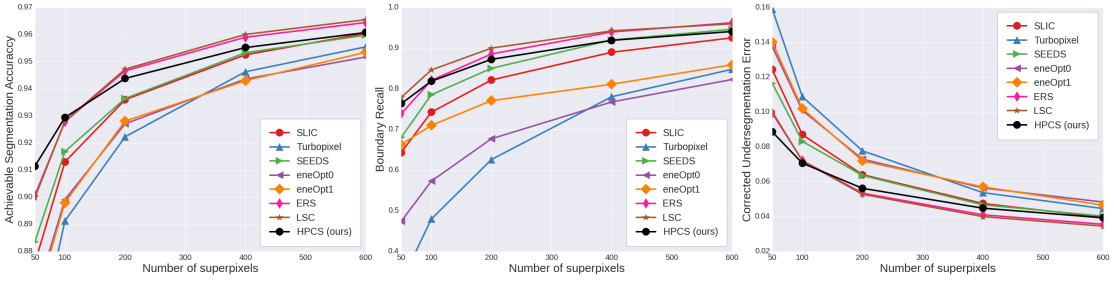


Figure 3.15: Quantitative evaluation of HPCS.

3.7 HPCS as post-processing for superpixel algorithms

Both our algorithms, HPCS and SMURFS can be used as post-processing for other superpixel algorithms. HPCS could be used to underlay superpixel algorithms as the 1st-order super-regions, and the use HPCS on top of that to generate over-segmentations with fewer super-regions as 2nd-order super-regions. Figure 3.16 shows an example of using HPCS to reduce the amount of representative units from other superpixel algorithms. It can be seen that using HPCS to post-process an already existing over-segmentation often preserves the same information with much fewer superpixels than using the source algorithm set to $M = 50$.

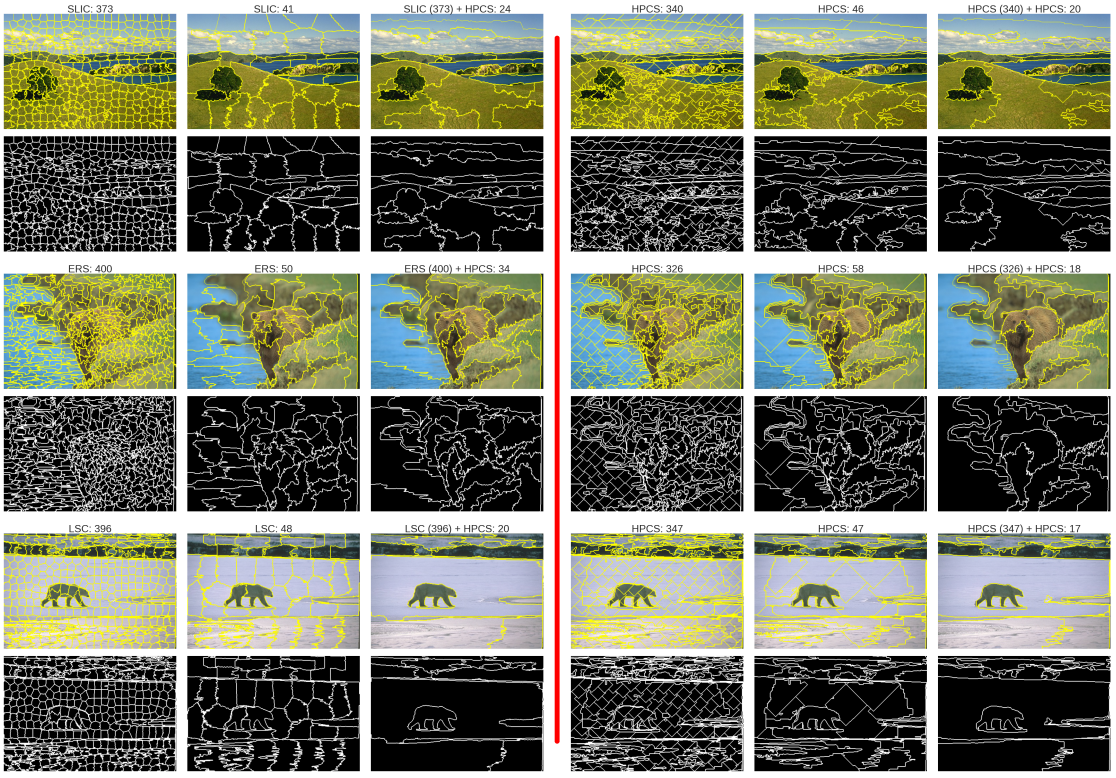


Figure 3.16: Example of using HPCS as post-processing algorithm for extracting 2nd-order super-regions. Left side shows 3 columns corresponding left to right: (1) image over-segmented with $M = 400$ superpixels, (2) image over-segmented with the same algorithm set to $M = 50$ superpixels and, result of using over-segmentation of (1) as input for HPCS to further reduce the number of superpixels. Right side shows the same procedure with HPCS as the source algorithm.

3.8 Conclusions

In this chapter we have proposed a novel algorithm that creates superpixels within a MRF formulation. The algorithm shows links with image over-denoising. Its graph formulation allows it to be generalizable to non-uniform grids and any other layout in a non-uniform space. Also, as the name states, HPCS is generalizable hierarchically, allowing to generate hierarchies of over-segmentations where each layer in the hierarchy is created by grouping elements of the previous layer, being the lowest layer the pixel grid. HPCS shows results comparable to the state of the art (better than most), while being faster and more generalizable. As shown in experimental results, HPCS can be used on top of an existing over-segmentation to further reduce the number of superpixels to represent the same detail.

4 SMURFS: Superpixels from Multi-scale Refinement of Super-Regions

In the previous chapter we introduced the concept of Super-Region Hierarchies and proposed an algorithm, HPCS, to create them. However, HPCS can be seen as a greedy bottom-up approach to generate these hierarchies, and thus, it doesn't exploit their full potential.

With the aim of creating better super-regions at every scale, we present SMURFS, an algorithm that produces super-regions by iteratively performing refinement updates at various levels of the hierarchy, in both bottom-up and top-down fashion. Given a set of super-regions at level i, $\mathcal{R}_i$, representative units from $\mathcal{R}_i$ are merged to create the level $\mathcal{R}_{i+1}$. Then, representative units from level $\mathcal{R}_{i+1}$ are split to generate a new over-segmentation for the level $\mathcal{R}_i$. By iteratively alternating split and merge steps at two scales of over-segmentations, super-regions that better represent the local neighbourhood and better capture the global properties of the image will be created. Figure 4.1 gives an overview of the algorithm. The following section will describe how to create a super-region hierarchy and obtain high accuracy superpixels from it. A three-level ($L = 2$) hierarchy will be discussed to extract superpixels, both for simplicity and because it is sufficient to create state-of-the-art superpixel in practice. section 4.2 will then generalize the process to multiple scales. Last superpixels and super-regions will be evaluated both qualitative and quantitatively in section 4.6 and compared to the state-of-the-art, and then some extensions and improvements will be proposed.

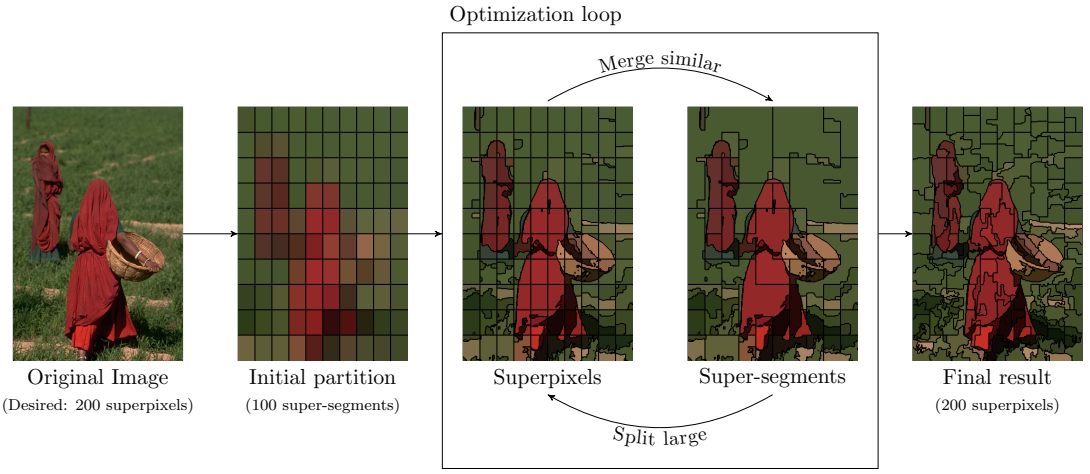


Figure 4.1: General overview of SMURFS algorithm. Superpixels (1st-order super-regions) and super-segments (superpixels of superpixels as 2nd-order super-regions) are optimized iteratively to better capture local information (through further splitting) and global properties (through merging) of the image.

4.1 Superpixels from a two-level Super-region Hierarchy

Let I be an image, $N = |I| = W \times H$ be the number of pixels in the image and $L = |\mathcal{H}| - 1 = 2$ the number of over-segmentation layers in the super-region hierarchy $\mathcal{H} = \{\mathcal{R}_0, \mathcal{R}_1, \mathcal{R}_2\}$. A super-region hierarchy is then represented as mappings

$$s_{0\to1} : \{1, \dots, N\} \to \{1, \dots, M_1\} \quad \text{and} \quad s_{1\to2} : \{1, \dots, M_1\} \to \{1, \dots, M_2\}, \tag{4.1}$$

where $M_1 = |\mathcal{R}_1|$ and $M_2 = |\mathcal{R}_2|$ represent the number of representative units in levels 1 and 2 of the hierarchy respectively and $N \gg M_1 > M_2$. Then, $s_{0\to1}$ maps each of the pixels p to a components in $\mathcal{R}_1$ and, similarly, $s_{1\to2}$ maps each of the component in $\mathcal{R}_1$ to a larger one in $\mathcal{R}_2$. It can be seen that finding the mapping $I \to \mathcal{S}_2$, denoted as $s_{0\to2}$, is straight forward by the chain rule.

This mapping enforces bottom-up membership and thus, prevents one component to be linked to more than one component in the next level of the hierarchy, thus, preserving our definition of a super-region hierarchy. Let us then, for completeness, denote as $\mathcal{S}_{i,j} = \{k | s_i(k) = j\}$ all the representative units $k \in \mathcal{R}_{i-1}$ whose mapping upwards in the hierarchy is the component $j \in \mathcal{R}_i$ (e.g. $\mathcal{S}_{1,5}$ will denote the set of all the pixels in the image assigned to superpixel 5).

The goal is then to find a disjoint set of K superpixels in the image. To do so, instead of iteratively updating/refining an initial set of K superpixels, as most of the state-of-the-art algorithms do, refinement steps will be alternated through the super-region hierarchy

to provide better quality superpixels. In practice, we found that there are two possible formulations to extract superpixels from a two-level super-region hierarchy:

1. If the number of desired superpixels K is *very small*[1], superpixels can be extracted as a level $L = 2$ super-regions. Mapping from N pixels to K superpixels with $N \gg K$ is very complicated, as the K superpixels would be too large to capture local textural information of the images, and thus, resulting in large under-segmentation. By adding an intermediate over-segmentation mapping $\mathcal{R}_1$ between pixels and superpixels, pixels will first be merged into this intermediate layer with $M_1 > K$ components that better preserve local textural information. Superpixels as $\mathcal{R}_2$ layer will be then extracted by merging elements from $\mathcal{R}_1$, better capturing global properties while preserving local textural features.

2. On the other hand, if K is large enough, each of the superpixels will better represent local information on the image. Thus, superpixels will be extracted as a $\mathcal{R}_1$ over-segmentation from the hierarchy, capturing local information of the image, while having a second level of super-segments $\mathcal{R}_2$ that better capture global information of the image and will prevent superpixels from falling into a local optima solution.

4.1.1 Splitting large regions

SMURFS can be seen as an extension and generalization of HPCS. The image partitioning of SMURFS follows the same over-denoising nature that the one in HPCS. Furthermore, let us recover the PCS Equation 3.7 (shown below to make this chapter self-contained):

$$E(\mathcal{R}^i, \boldsymbol{f}) = \sum_{p \in \mathcal{V}^i} \left\| D_p^i - \theta_{f_p}^i \right\|_2^2 + \lambda \sum_{(p,q) \in \mathcal{E}^i} w_{p,q} \cdot \left\| \theta_{f_p}^i - \theta_{f_q}^i \right\|_1, \tag{4.2}$$

For completeness, let us generalize the above equation to an arbitrary unary and pairwise potentials, as well as the feature space:

$$PCS(\boldsymbol{f}) = \sum_{p \in \mathcal{V}} \psi_p(\Theta, D) + \lambda \sum_{(p,q) \in \mathcal{E}} w_{p,q} \cdot \phi_{p,q}(\Theta, D) \tag{4.3}$$

The above optimization scheme can be seen as a spatial clustering, as nearby similar colors are assigned to the same cluster center. The optimization is then followed by a quick connected components approach to identify new regions as connected components with the same optimal label.

[1] Empirically we refer to a very few number of superpixels if $K \leq \sqrt{N}$.

In SMURFS, however, this optimization scheme is applied to every large region of the image, yielding for each of the regions a subset of smaller components that better represent the region. Each of the regions can be separately optimized in a completely parallel fashion, which is very important for high dimensional images or volumes, but can also be minimized globally with a single graph cut Boykov and Kolmogorov (2004) and an appropriate graph setup for standard color images, as explained in the next section. Let us, for completeness, write the completion of Equation 3.7 for splitting all the regions on an image:

$$E(\mathcal{I}, \boldsymbol{f}) = \sum_{i}^{|\mathcal{R}|} E(\mathcal{R}^i, \boldsymbol{f}) = \sum_{i}^{|\mathcal{R}|} \left(\sum_{p \in \mathcal{V}^i} \left\| D_p^i - \theta_{f_p}^i \right\|_2^2 + \lambda \sum_{(p,q) \in \mathcal{E}^i} w_{p,q} \cdot \left\| \theta_{f_p}^i - \theta_{f_q}^i \right\|_1 \right), \tag{4.4}$$

where $\mathcal{I}$ is the original image and $\mathcal{R} = \{\mathcal{R}^i, \ldots, \mathcal{R}^n\}$ the set of non-overlapping regions of $\mathcal{I}$, so that $\bigcup_{i=1}^{|\mathcal{R}|} \mathcal{R}^i = \mathcal{I}$. Last, in order to ease derivations of the method here presented in the following sections, let us generalize the MRF graph splitting of Equation 3.7 and Equation 4.4

$$E(\mathcal{I}, \boldsymbol{f}) = \sum_{i}^{|\mathcal{R}|} E(\mathcal{R}^i, \boldsymbol{f}) = \sum_{i}^{|\mathcal{R}|} \left(\sum_{p \in \mathcal{V}^i} \psi_p(\Theta^i, D^i) + \lambda \sum_{(p,q) \in \mathcal{E}^i} w_{p,q} \cdot \phi_{p,q}(\Theta^i, D^i) \right), \tag{4.5}$$

where ψ and ϕ represent unary and pairwise constraints functions of the sampled centres Θ^i from the sampling function $\Gamma(\mathcal{R}^i, D^i)$ and D^i is the feature mapping function that maps a pixel p to a predefined feature space.

Note that labels f_p and f_q of the MRF unary and pairwise potentials are implicitly omitted in Equation 4.3 and Equation 4.5 for compactness purposes.

Efficient simultaneous partition of every region

Let us, for simplicity, consider the case with $K = 2$ colors, where each of the regions will be separated in a binary fashion. The optimal labelling for each of the regions can be obtained very efficiently with the well known maxflow method Boykov and Kolmogorov (2004). However, this formulation would require M graph creations and M optimization schemes, where $M = |\mathcal{R}_2|$ is the actual number of regions to be split. This is a very efficient solution for high dimensional images where M and the number of pixels/voxels N are very large, as every sub-problem can be solved in parallel independently. For standard natural images, however, the cost of constructing different graphs for every region is as expensive as the cost of calculating the optimal cut once the graph is constructed.

Thus, we propose a method for calculating the optimal cut for each of the regions simultaneously. First, a standard connected graph $\mathcal{G}$ is extracted from the image and pixel differences are calculated according to W_{pq} in equations 4.4 and 4.5. Next, for every split iteration, the following modifications are applied to $\mathcal{G}$ to form the current graph to be split $\mathcal{F}$:

1. Pixels assigned to different super-segments are disconnected, isolating the max-flow problem inside each region.
2. Every pixel assigned to a small super-segment (which is not a candidate to be separated) is disconnected from its neighbours and connected to the *background* label with infinity weight. This will force all the pixels of that super-segment to be assigned the same label, and thus, being in the same superpixel after the optimization.
3. Each of the pixels is connected to each of the cluster centres θ_k (in the binary example θ_1 and θ_2) with their corresponding weight (as ψ_p in equation 4.4)

Note that, although all the pixels are connected to $K = 2$ labels, each of the cluster centres θ_k is different for every region $\mathcal{R}_2^i$, as different cluster centres are sampled for each region. By transforming the energy minimization to a min-cut/max-flow problem of a graph efficient solutions can be found. This yields, in practice, the optimization of all the regions simultaneously, since the pixels belonging to different regions in $\mathcal{R}_2$ are disconnected, and thus, the maxflow can't propagate from one region to another. Finding the optimal solution to the above min-cut/max-flow problem in the graph $\mathcal{F}$ is then equivalent to finding the solution for each of the individual regions. It is also considerably faster than computing a *standard* graph cut (maxflow) in an image of the same size, since the graph $\mathcal{F}$ contains less edges than $\mathcal{G}$. Extension to multiple labels $K > 2$ is straightforward by means of α-expansion (Boykov et al., 2001; Boykov and Kolmogorov, 2004) or FastPD (Komodakis and Tziritas, 2007; Komodakis et al., 2008). Figure 4.2 illustrates this process for an artificial setup with $K = 2$ and 3 super-segments.

4.1.2 Merge similar regions

On the previous section an algorithm for the splitting of image regions was provided. Qualitative visual results showed that the splitting procedure is in practice sufficient to produce superpixels with basic post-processing. In some cases, such as images with high textural content or weak boundaries, the algorithm could benefit from an iterative

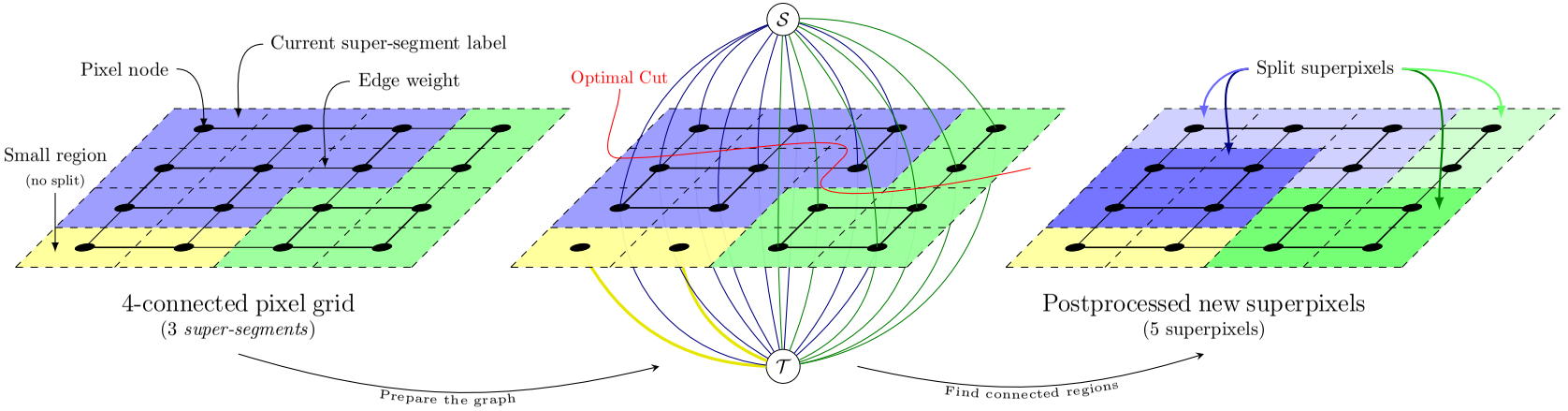


Figure 4.2: Region splitting procedure with $K = 2$ cluster centres. Large regions (blue & green) are optimally separated in smaller ones while small regions (yellow) are not further separated. To do so, first edges connecting nodes (pixels) of different region (coloured) are removed. Then, each of the nodes is connected to source and sink artificial nodes (except small regions, which are only connected to sink). Last, the optimal split for all the regions is obtained simultaneously with a single cut for all of them. Edge weight is indicated by their thickness. Note the missing edges between nodes of different colors, disconnecting the regions during the optimization.

refinement procedure. We will then define a region merging procedure with the aim of better capturing global consistencies on the image from previously extracted regions $\mathcal{R}_1$.

The goal is then to merge a set of regions $\mathcal{R}_1$ into a new set of regions $\mathcal{R}_2$, where $|\mathcal{R}_1| > |\mathcal{R}_2|$. First, a Region Adjacency Graph (RAG) $T = (\mathcal{V}_1, \mathcal{E}_1, W)$ is constructed from $\mathcal{R}_1$. Here $\mathcal{V}_1$ represent the nodes of the RAG, which correspond to each of the regions $\mathcal{R}^k \in \mathcal{R}_1$. $\mathcal{E}_1$, respectively, represents edges connecting adjacent regions $\mathcal{R}^p$ and $\mathcal{R}^q$ in $\mathcal{R}_1$ with weight $W_{p,q}$. Two regions are adjacent if and only if they share boundary pixels.

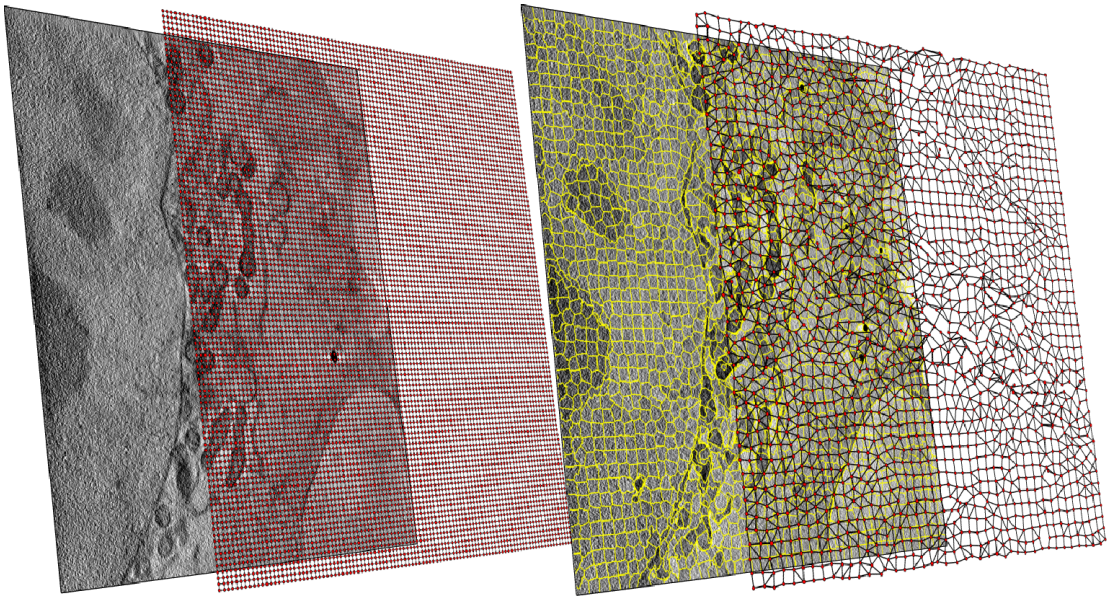


Figure 4.3: An example Region Adjacency Graph (RAG) calculated on an slice of a cryo-SXT dataset. Left shows the original image with the corresponding pixel grid overlay on it. Right shows the same image with superpixels computed and a RAG overlaid on it by connecting together adjacent superpixels.

With the RAG constructed, as shown in figure Figure 4.3, a suitable similarity/dissimilarity function has to be found in order to be able to evaluate how likely two regions are to merge. Extracting descriptors from superpixels has been extensively studied in the literature and the most common approach towards description of superpixels is to collect pixel descriptors and combine them into a single, more robust, superpixel descriptor. Superpixel descriptors can be formed in different ways, from a mean descriptor vector obtained by averaging the values for all the pixels inside the superpixel (assuming that the features are real valued standardized vectors), to covariance-like descriptors or more robust histogram of textonized features. A range of feature encoding techniques exist for different posterior application targets with different interesting properties. More in depth details into feature building for super-regions will be given in chapter 6 and chapter 7.

For the purpose of computing a region description to assess the likelihood of two components belonging to the same higher-level object and, thus, merging them, we found that a mean feature encoding provides a very good trade-off between accuracy and computational complexity. Especially considering that region descriptors will be computed iteratively as part of a multi-scale refinement process, small errors introduced during the merging process will later be fixed with further splitting and merging iterations.

For a region $\mathcal{R}^k \in \mathcal{R}_1$, where every pixel $p \in \mathcal{R}^k$ is described by the $L \cdot a \cdot b$ color-vector $D_p = \{l, a, b\}$, then the region descriptor Q_k (where k refers to $\mathcal{R}^k$) is defined as the mean vector

$$Q_k = \frac{1}{N} \sum_{p}^{\mathcal{R}^k} D_p, \quad \text{and} \quad W_{p,q} = \|Q_p - Q_q\|_2^2, \tag{4.6}$$

with $W_{p,q}$ being the Euclidean distance, measuring dissimilarity between regions p and q. Despite its simplicity, later will be shown that it computes discriminative enough descriptors for merging purposes while being extremely fast. It is important to note that since this merging step will later be part of an iterative procedure, mean descriptors can be computed at no cost by using as mean descriptors features sampled from $\Gamma(\mathcal{R}^i, D^i)$ in the previous splitting iteration, which essentially encodes the most discriminative colour (feature vector) to which the region is assigned to.

Once a suitable mapping from pixel to region description Q has been found, and a dissimilarity function has been defined for the new feature space, the merging process follows an agglomerative clustering approach similar to the one in (Felzenszwalb and Huttenlocher, 2004), but applied to the RAG and with different constraints, preventing it from extracting sparse irregular regions. The problem of finding a group of $|\mathcal{R}_2| < |\mathcal{R}_1|$ regions is then formulated as finding the Minimum Spanning Forest (MSF) of the RAG T. A Minimum Spanning Tree of a graph is defined as the minimum-cost edge path that connects all the nodes in a graph without creating any loop. The problem is very well understood and is solvable very efficiently with the greedy Kruskal algorithm (Kruskal, 1956; Cheriton and Tarjan, 1976), which orders edges $W_{p,q}$ of the graph T by weight, creates a disconnected graph T' and iteratively merges nodes p and q by adding its edge to T' if adding the edge $\{p, q\}$ to the graph doesn't create a loop. The algorithm of (Felzenszwalb and Huttenlocher, 2004) follows the same idea, but adds a threshold function acting as a constrain to prevent large regions from merging if they are very different. This results in some disconnected regions, creating a Minimum Spanning Forest, which is a set of Minimum Spanning Trees that connect the whole graph, but are disconnected between them.

In order to merge regions previously extracted, edges in the RAG are ordered by weight $W_{p,q}$ and, iteratively, the two most similar regions are merged. However, as the aim is

to obtain a reduced subset of regions, regions are only merged if they satisfy a merging condition. Constraints to merging can be imposed by color similarity, restricting the size of the region, or even their proximity. Each of these options would result in a different merging with different properties as will be seen later in section 4.5. Initially a disconnected graph T' is created, where each of its nodes (regions in this case) represents an unique component or MST. An internal similarity or homogeneity measure H keeps record of the homogeneity of each of the regions as they grow by keeping the edge with largest cost while merging regions

$$H_k = \max_{(p,q)\in \mathrm{MST}_k} W_{p,q}. \tag{4.7}$$

It is straightforward to see that H_k is a very efficient and yet robust measure of region homogeneity, as an MST contains a set with the minimum-cost edges. Low values of H_k indicate a very homogeneous region, while large values of H_k indicate a region with high texture or intensity differences.

The merging condition then uses an internal cost C_k, for example defined as a function of the region size, and a thresholding function τ,

$$C_k = H_k + \frac{\mu}{S_k}, \quad \text{and} \quad \tau(p,q) = W_{p,q} < \min(C_p, C_q), \tag{4.8}$$

where S_k is the region size (in pixels), and μ is a parameter that describes the desired size for the target regions. The internal cost C_k for each region evaluates how likely a region is to be merged as a function of their homogeneity and size. Two regions are merged if they satisfy the merging condition τ, which essentially allows similar regions (in both size and appearance) to be merged.

The merging criterion is implemented using Disjoint-Sets (more details below) for highly efficient access to the graph, as checking for loops in the graph can be done in $O(1)$. The merging can be solved in linear time (in the number of edges) by starting with $H_k = 0$ and $S_k = |\mathcal{R}^k|$ for every region $\mathcal{R}^k$ updating $H_k = W_{p,q}$ and $S_k = S_p + S_q$ every time two nodes p and q are merged.

Disjoint-Sets for efficient MST/MSF computation

Disjoint-Sets are a dynamic lookup structure ideal for merging purposes. They are used to represent trees with a fixed-size table where each node points towards its parent. This lookup table allows to perform addition, re-routing, merging and splitting operations in a tree-like structure very efficiently. The region merging procedure in an image (which is a planar graph) is first generalized to RAG, and then the formulation of the problem

is transformed to a MSF problem, which makes it suitable for using with Disjoint-Set optimizations. The initial representation of a disconnected tree is given by a list, where each entry of the list represents a node in the disconnected graph T' and each of the nodes points towards its parent (initially themselves). Find below a table explaining how to represent a few operations within the tree: From the table it can be seen that

Table 4.1: Disjoint-Set operations. Ten disconnected nodes and merge/split operations applied to them. It can be seen that merging/splitting operations in a graph takes a single operation without having to transverse the tree.

1	2	3	4	5	6	7	8	9	10	Node Id
1	2	3	4	5	6	7	8	9	10	Initial state
1	2	3	4	5	6	5	8	9	10	Merge nodes 5 and 7
1	2	3	4	2	6	5	8	9	10	Merge nodes 2 and 5
1	2	2	4	2	6	5	8	9	10	Merge nodes 2 and 3
1	2	2	4	5	6	5	8	9	10	Split nodes 2 and 5

Disjoint-set provide a very simple, yet powerful, tree representation that can be used to compute merging operations very efficiently. It is important to note that in the above, most simple, example we are dealing with a 1D array, where each node has a single property (parent), but the array can be extended to a 2D table in a straight forward, where each of the nodes would have several features (parent, size, R, G, B, L, a, b) and merging two regions, while updating their internal properties, can be done again as efficiently as before by updating just the *parent* node.

Finding the *parent* of node in a Disjoint-Set will then be a very important part in any further procedure. In order to properly merge regions, without creating loops in the MST, a Disjoint-Set can be used to replace $merge(p, q)$ by $merge(parent(p), parent(q))$ and merging corresponding regions iff $parent(p) \neq parent(q)$.

Let us then give a proper definition of *parent* and *merge* function over a Disjoint-Set:

Algorithm 5: Find parent in a DS

```
1 Function parent(p):
2     q ← p
3     while q ≠ DS(q) do
4         q ← DS(q)        // parent(DS(q))
5     end
      /* Quick Lookup                    */
6     DS(p) ← q
7     return q
```

Algorithm 6: Merge two nodes in a DS

```
1 Function merge(p, q):
2     p ←parent (p); q ←parent (q)
      /* Union by rank                   */
3     if S_p < S_q then
4         DS(p) ← q
5         S_p ← S_p + S_q
6     else
7         DS(q) ← p
8         S_q ← S_p + S_q
9     end
```

On the algorithms above, DS stands for a Disjoint-Set and S_i the size of the node i. It can be seen that the above two simple optimizations (based on (Tarjan, 1975; Cormen et al., 2009)) speed-up the MSF by several orders of magnitude. Since the final aim of the MSF for region merging is to just create connected components, the full tree structure doesn't need to be preserved as long as each of the nodes references its top level parent. That is what the *Quick Lookup* optimization does in the *parent* function: instead of traversing the tree every time to find the parent of a node, every time this happens the parent gets updated in order to speed-up future searches. Similarly, *Union by rank* optimization makes sure always the smaller node (the one with fewer children in the tree) is merged to the other one, again, reducing the amount of tree traversal operations needed in the future. Further optimizations exist, such as the commented *Path Compression* of line 4 to recursively update the paths for every node, however, we found that since the MSF will be created only for finding connected components and wont be further re-utilized; this does not provide any speed-up in practice.

4.1.3 Similarities and differences between splitting and merging

It is important to note that the previous splitting and merging procedures are related each other. From subsection 4.1.1 could be used to split the connected RAG T into disconnected regions. It would only require the definition of a descriptor $D_{\mathcal{R}^i}$ mapping for each of the region $\mathcal{R}^i$ and a sampling function ρ to sample cluster centres from T. This shows that region splitting and region merging are linked concepts in our Graph-based Hierarchical framework. That is, splitting large regions from $\mathcal{R}_2$ into regions $\mathcal{R}_1$ is equivalent as merging nodes from $\mathcal{R}_0$ into larger nodes in $\mathcal{R}_1$, which is essentially what subsection 4.1.1 does by considering each of the pixels as a unit and removing edges from the graph during the post-processing of the energy based optimization.

This then proves that for a given connected graph $G = (\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = N$, finding a subset of $M < N$ nodes can be done by either splitting or merging:

- A splitting procedure would require one to remove edges from G until M regions are disconnected from the rest of the graph.
- A merging procedure would start with a disconnected graph $G' = (\mathcal{V}, \emptyset)$ with the same nodes as G but no edges, and edges from G would be added to G' until M connected regions are formed on G'.

4.2 SMURFS Algorithm

(H)PCS algorithm described in the previous chapter already achieves similar performance in both speed and quality compared to the current state of the art, outperforming most of them in standard benchmarks. It also has some advantages compared to the state of the art algorithms, since it considers global properties of the image by means of a MRF formulation and a prior clustering to identify representative colors in the image, and operates by default in the normalized $L \cdot a \cdot b$ color-space, it automatically adjust the scale of the parameters of the algorithm to each of the individual images to be processed.

However, performing global decisions comes at an expensive computational cost, which in the case of PCS is relaxed by first selecting k-means as an approximate fast sampler to select k representative colors from the image, and using an approximate solver such as FastPD (Kolmogorov et al., 2007) to find the optimal labelling to the partition problem. However, due to the global balance of colors that determines how likely neighbouring pixels are to merge, local small changes might be missed and thus, locally disjoint sections end up joint together when global appearance of the image is taking in to account.

Here is where the state of the art in superpixels converges. Most of the superpixel algorithms perform local optimization moves (e.g. SLIC (Achanta et al., 2012), TurboPixels (Levinshtein et al., 2009), ...) which causes them to fall in local optima solutions that misses some representative contours of the image. Some of the last algorithms include rebalancing steps between Expectation-Maximization iterations, such as Linear Spectral Clustering (Li and Chen, 2015) that adds a rebalancing step by means of a linear spectral clustering to the standard SLIC formulation trying to better capture long-range interactions within SLIC iterations. Others such as Entropy Rate Superpixels (Liu et al., 2011) directly minimize a global formulation based on the entropy rate of merging two sets, which is very interesting formulation, but relies on a greedy algorithm to approximate the solution of the problem in an acceptable time (being still more computationally expensive than most of the algorithms with similar performance). Last, SEEDS (Van den Bergh et al., 2012) performs a top-down approach to hierarchically perform fast updates to a predefined set of superpixel boundaries.

To benefit from both, local discriminative moves and at the same time capture long-range information hidden in the images, Superpixels from Multi-Scale Refinement of Super-Regions (SMURFS) (Luengo et al., 2016d) is proposed as an iterative superpixel refinement algorithm that combines both top-down (to further split regions in locally discriminative areas) and bottom-up (to merge similar local regions in larger, more-representative ones) approaches.

To that end, we recover the initial formulation explained at the beginning of the chapter, where instead of treating the image as a whole, and splitting/merging it to form superpixels, a set of regions $\mathcal{R}_2$ is assumed to already be extracted from the image. Then, each of these regions will be further split into $\mathcal{R}_1$ following Equation 4.5 to better capture the details within them and after that, each of the generated smaller regions will be merged as in subsection 4.1.2 to their similar neighbours, generating a set of larger regions $\mathcal{R}_2$. Alternating both steps will iteratively improve the quality of regions at both levels of the hierarchy $\mathcal{R}_1$ and $\mathcal{R}_2$, thus, generating a Super-Region Hierarchy that iteratively improves all of the levels itself.

Algorithm 7: SMURFS Superpixels

Input : image: I, #superpixels: K, smooth: λ, #nclusters: k, #iter: T

Output: Superpixels: $\mathcal{R}_2 = s_{0\to 2}$

$\mu_K = \frac{|I|}{K}$, $S_K^- = 0.5 \cdot \mu_K$, $S_K^+ = 2 \cdot \mu_K$;

Obtain graph $G = (\mathcal{V}, \mathcal{E}, W)$ connecting adjacent pixels in I; // Graph of $\mathcal{R}_0$

Map image I to normalized $\{L \cdot a \cdot b\}$ feature space $I \to D \in \mathbb{R}^{N\times 3}$; // $N = |I|$

Create initial grid of $|\mathcal{R}_2| = M_G = M/2$ uniformly distributed regions.;

for *number of iterations* **do**

/* Split large regions */

Sample k centroids for each region: $\mathbf{\Theta}^i = \Gamma(\mathcal{R}_2^i, k) = \{\theta_1^i, \dots \theta_k^i\}$; // $\Gamma = k$-means

Obtain optimal labelling $\boldsymbol{f}^*$ by minimizing Equation 4.5 as in Equation 4.4

$$E(\mathcal{I}, \boldsymbol{f}) = \sum_i^{|\mathcal{R}|} E(\mathcal{R}_2^i, \boldsymbol{f}) = \sum_i^{|\mathcal{R}|} \left(\sum_{p\in\mathcal{V}^i} \left\| D_p^i - \theta_{f_p}^i \right\|_2^2 + \lambda \sum_{(p,q)\in\mathcal{E}^i} w_{p,q} \cdot \left\| \theta_{f_p}^i - \theta_{f_q}^i \right\|_1 \right),$$

$\mathcal{R}_1 = \{\mathcal{R}_1^1 \dots, \mathcal{R}_1^t\}$ = Find connected components of $\boldsymbol{f}^*$; // t > K

/* Merge similar regions */

Create RAG $\mathcal{T} = (\mathcal{V}^\mathcal{T}, \mathcal{E}^\mathcal{T}, W^\mathcal{T})$ of regions in $\mathcal{R}_1$ with weights according to Equation 4.6

$$Q_k = \frac{1}{N} \sum_p^{\mathcal{R}_1^k} D_p^k, \quad \text{and} \quad W_{p,q}^\mathcal{T} = \|Q_p - Q_q\|_2^2 ,$$

Order edges of the RAG $\mathcal{T}$ by weight $W_{p,q}^\mathcal{T}$;

Iteratively merge the edges that satisfy the merging criterion on Equation 4.8;

$\mathcal{R}_2 = \{\mathcal{R}_2^1, \dots, \mathcal{R}_2^m\}$: Find mapping from pixels to regions $s_{0\to 2}$; // $t > m \geq K$

end

/* Post-process: Merge small regions */

while $i < \left|\mathcal{E}^\mathcal{T}\right|$ *and* $|\mathcal{R}_2| > K$ **do**

$\mathcal{R}_2^p, \mathcal{R}_2^q = \mathcal{E}^\mathcal{T}(i)$; // Next edge in increasing weight

if $|\mathcal{R}_2^p| < S_K^-$ *or* $|\mathcal{R}_2^q| < S_K^-$ **then**

merge$(\mathcal{R}_2^p, \mathcal{R}_2^q)$

end

$i++$;

end

The algorithm 7 summarizes the SMURFS algorithm by combining the derivations of the

previous sections. It alternates split and merge operations at two levels of the hierarchy $\mathcal{R}_1$ and $\mathcal{R}_2$ to return the final superpixels as a post-processing of $\mathcal{R}_2$. It is important to note that in the current algorithm, the implementation of Formulation 2 (F2) in case the aim is to obtain small superpixels, that is straight forward by just adding another additional split operation and returning the post-precessed $\mathcal{R}_1$ as the superpixels.

4.2.1 Hierarchical Generalization

Due to the graph based formulation, hierarchical generalization of SMURFS to further layers of the hierarchy is straightforward for both Formulation 1 (F1) and Formulation 2 (F1):

- Hierarchical F1: by stacking intermediate layers between outputs it allows to easily generalize SMURFS to higher levels by adding intermediate layers between each of the outputs as shown in Figure 4.6, resulting in independent SMURFS iterations that allow to refine each of the outputs further independently in a bottom-up fashion.
- Hierarchical F2: directly stacking levels one above the other generates simpler hierarchies (Figure 4.7), at the expenses of not being able to refine independent levels.

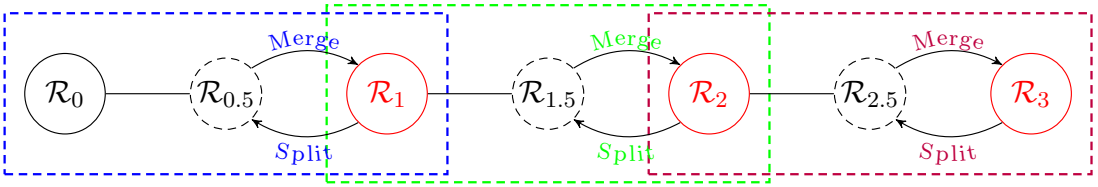


Figure 4.6: Higher level SMURFS using Formulation 1 (F1). Intermediate levels are stacked between outputs (red circles) adding a temporary layer to better capture local environment of the previously extracted regions. Generates independent SMURFS steps between outputs which is straight forward.

4.2.2 Hierarchical Updates and Hierarchical Consistency

In order to keep a valid Super-Region Hierarchy modifications at a given level $\mathcal{R}_i$ are not allowed without throwing a cascade of updates to preserve super region mappings $s_{i \to j}$ for every $j > i$. This is, remembering our Super-Region Hierarchy definition, each of the layer is composed from elements of the previous layer, thus, once the the full hierarchy

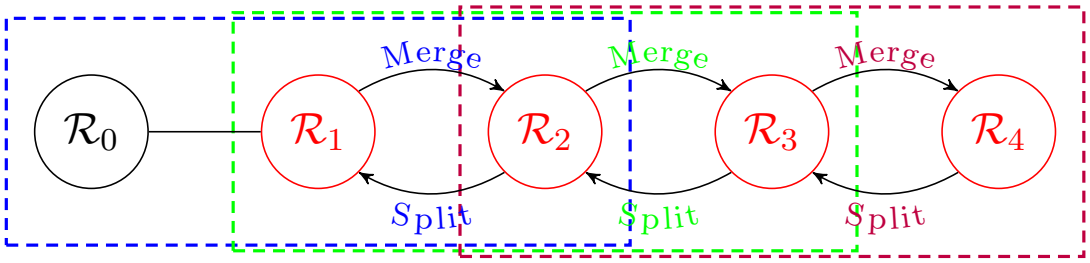


Figure 4.7: Higher level SMURFS using Formulation 2 (F2). Straight forward bottom-up hierarchy representation, each level resulting in a larger, more representative regions that represent a standard Super-Region Hierarchy

is constructed $\mathcal{H} = \{\mathcal{R}_0, \ldots, \mathcal{R}_i, \ldots, \mathcal{R}_M\}$ precautions have to be taken to update the level i while preserving all the other levels $j > i$.

It can be seen from algorithm 7 that 3 levels of the hierarchy are needed to perform an iteration of SMURFS: $\mathcal{R}_{i-1}, \mathcal{R}_i$ and $\mathcal{R}_{i+1}$. While the merge operation is a standard bottom-up approach that generates $\mathcal{R}_{i+1}$ regions from the level $\mathcal{R}_i$, the split operation is designed to maintain hierarchical consistency by using representative units from level $\mathcal{R}_{i-1}$ to split larger regions from $\mathcal{R}_{i+1}$ into the resulting set $\mathcal{R}_i$. In other words, larger regions candidates to be split from the above level ($i+1$, e.g. super-segments) are going to be separated using the units from the previous level ($i-1$, e.g. pixels) to result in updated units for the current level (i, superpixels).

Because of this, we say that our split operations preserve the consistency of the Super-Region Hierarchy. In practice, an arbitrary level $i|0 < i < M$ can be safely updated by a split operation given the previous $i-1$ and next $i+1$ levels of the hierarchy.

The merging operation is designed to update higher levels given the information of the current level. By doing that it tries to enhance the long-range information preserved within the algorithm. Unlike the split operation, merging is a single bottom-up update, which would break the hierarchical consistency if not properly handled. Level i will be used to update level $i+1$, however, this is not a valid operation if level $i+2$ exists, because by updating $i+1$ through a merge operation, $i+2$ would no longer be formed of elements from $i+1$. In order to prevent this from happening, the merge operation has to be redefined to propagate update operations up in the hierarchy until all the levels $j > i$ have been properly updated. To this end, we redefine the split and merge operations of

SMURFS to work hierarchically as follows:

Algorithm 8: SMURFS HSplit

Function `HSplit`$(\mathcal{H}, i, \Gamma, K)$**:**
- $G_{i-1}, D_{i-1} \leftarrow$ Create RAG from $\mathcal{R}_{i-1}$
- **for** $\mathcal{R}^j_{i+1} \in \mathcal{R}_{i+1}$ **do**
  - $T^j \leftarrow$ subset of G_{i-1} corresponding to $\mathcal{R}^j_{i+1}$, this is: $\{k \,|\, s_{i-1 \to i+1}(k) \in \mathcal{R}^j_{i+1}\}$
  - $Q^j \leftarrow$ subset of D_{i-1} corresponding to $\mathcal{R}^j_{i+1}$
  - $\boldsymbol{\Theta}^j \leftarrow$ Sample K centres from $\Gamma(T^j, Q^j)$
  - $\boldsymbol{f}^* \leftarrow$ optimize Equation 4.5 given T^j, Q^j, Θ^j
- **end**
- $G_i, D_i \leftarrow$ Find connected components in $\boldsymbol{f}^*$.
- **return** G_i, D_i

The algorithm 8 shows how to perform an splitting step on an arbitrary level i given level $i-1$ and $i+1$. It takes as an input the parameters $\mathcal{H}$ (with $|\mathcal{H}| - 1 = M$) for the current hierarchy, $i \,|\, 0 < i < M$ as the level to be updated, Γ as the sampling function and K, the number of representative units to sample from each region. The algorithm requires sampling colors from each of the regions to be split, although representative colours from previous merge iterations could be used to avoid constant sampling. For every subregion in $\mathcal{R}_{i+1}$, its corresponding sub-graph T along with the feature mapping Q is extracted from G_{i-1}, D_{i-1} by selecting all the nodes and vertices from G_{i-1} that map upward to the current region $\mathcal{R}^j_{i+1}$.

The optimization of Equation 4.5 can then be performed in parallel for every region $\mathcal{R}^j_{i+1}$ or jointly with a single graph cut as described in subsubsection 4.1.1.1.

Algorithm 9: SMURFS HMerge

Function `HMerge`$(\mathcal{H}, i)$**:**
- $G_i, D_i \leftarrow$ Create RAG from $\mathcal{R}_i$
- `/* Propagate merge operations upward in the hierarchy */`
- **for** $j \in \{i, i+1, \ldots, M-1\}$ **do**
  - $G_j, D_j \leftarrow$ get RAG from $\mathcal{R}_j$
  - Order edges of G_j by weight $W_{p,q}$
  - $G_{j+1}, D_{j+1} \leftarrow$ Iteratively join nodes whose edges satisfy the merging criterion in Equation 4.8
- **end**
- **return** T, Q

The merge operation in algorithm 9 for a hierarchy with more than 2 layers is redefined as an atomic recursive operations for all the levels above i. This ensures that after

the recursive merging operation, all the upper levels are still formed by elements of the previous levels, maintaining this way the hierarchical consistency.

4.3 The sampling function Γ: towards a robust deterministic solution

In the previous section the details for both superpixel algorithms have been discussed. In this section we will focus in particular on the sampling function Γ of the split step, that plays a crucial part in the algorithms and was given for granted in the previous sections. It should be easy to note that both superpixel algorithms described in this chapter are dominated by a sampling function used to sample centroids from each of the regions. If the sampled centroids are not good descriptors of the respective regions, the over-segmentation result will fail and thus, have a poor boundary accuracy.

When it comes to centroid selection, there are few straight forward options to select a subset of k centroids $\Theta = \{\theta_1, \ldots, \theta_k\}$ from a set of N samples $D = \{D_1, \ldots, D_N\}$:

- Basic Sampling: choose k samples given a criterion (e.g. first k samples)
- Random Sampling: choose k samples from N at random.
- Uniform Sampling: choose k samples uniformly distributed across the samples D (i.e. a sample every N/k)

However in theory, either of the above sampling criterion would hardly result in robust centroid selection, thus, the accuracy of the resulting superpixels would be diminished.

In practice however, SMURFS works well (and extremely quick) by using either Random or Uniform sampling. This is because unlike other cluster-based algorithms, SMURFS is more robust to noise and parameter tunning since,

- It adds pairwise constrains in the MRF formulation, taking into account the neighbourhood of each pixel during the optimization before making a decision,
- It is an iterative refinement, where mistakes in previous iterations are likely to be fixed in future iterations.

This way, SMURFS is less sensitive to noise and works well even with randomly selected centroids. Random and Uniform sampling methods are good enough to obtain results

comparable to the state of the art, however, these results can be further improved with more advanced sampling functions.

A vast literature in clustering algorithms exists, where the aim is to identify different clusters or colonies of samples in within large populations. Here our aim is not to identify these whole clusters, as that would be the whole purpose of the superpixel algorithm, but to identify a subset of representative data points.

To do so, an initial solution would be to run k-means and select the k centroids, extracting k discriminative centroids from each of our large regions with N samples. The k-means algorithm tries to separate data in clusters or groups with similar variance, minimizing the well known within-cluster sum-of-squares (or *inertia*) criterion:

$$\sum_{i=0}^{N} \min_{\mu_j} \|x_i - \mu_j\|_2^2 \tag{4.9}$$

where $\boldsymbol{X} = \{x_1, \ldots, x_N\}$ are the samples or observations, $\mathcal{S} = \{S_1, \ldots, S_k\}$ is the set of k clusters to which each of the samples have to be assigned and μ_i is the mean of all the samples assigned to cluster S_i.

The inertia criterion can be seen as a measure that identifies how coherent each of the clusters are internally. Optimizing the k-means algorithm will result in the cluster centres $\{\mu_1, \ldots, \mu_k\}$ as representative centroids for each region.

The inertia criterion assumes the clusters to be isotropic and theoretically it performs poorly as a clustering algorithm when

- The number of clusters is unknown or wrong.
- The clusters have unequal variance or are of different size.
- Inertia is not normalized and doesn't work well in high-dimensional spaces.

All the above problems are present in our formulation, as regions are not known a priori and thus it is impossible to determine how many an which are the cluster centres for each of the regions.

Other more robust clustering alternatives exist, such as Meanshift (Cheng, 1995), Spectral Clustering (Shi and Malik, 2000; Stella and Shi, 2003), DBSCAN (Ester et al., 1996) or Birch (Zhang et al., 1996) that take into account the structure and density of the data, some support cluster discovery (without needing a prior number of clusters k) and many more improvements over the simplest k-means that could assist to determine those unknown centroids. However, the whole purpose of our sampling function is to extract

a subset of *representative* centroids quickly. Any of the aforementioned clustering algorithms would be overkill for such task, as they have a too large computational demands for such purpose.

In practice, we found that using k-means as a sampling source for SMURFS, despite being simple, gives a performance improvement in all the standard quantitative metrics over the current state of the art (at the time of writing this thesis). For the same reason given at the beginning of the section, we found that centroids are important and better centroids improve the quality of the regions, but are not critical to the algorithm and the centroids given by the k-means algorithm work well in practice.

Having selected a k-means as the default sampling function introduces another source of randomness to the algorithm. While the randomness effect is small for SMURFS due to the iterative MRF refinement, the algorithm would return different output superpixels for the same input image. A recent study Celebi et al. (2013) compares different initialization methods existing for k-means, along with the most popular one k-means++ (Arthur and Vassilvitskii, 2007). They found that two deterministic methods introduced in (Su and Dy, 2007), VAR-Part and PCA-Part both based in hierarchical partitioning, achieve results comparable to the most popular randomized selection methods.

The VAR-Part algorithm, used as default in SMURFS due to being quicker while achieving good approximations for initial centroids, starts with the whole data as a single cluster, and then iteratively splits a cluster in two along the axis (feature) with higher variance. Then, the mean of all the samples inside each of the centres are returned as the initial centroids, and k-means is run on top of them until convergence, which can be summarized in the following steps:

1. Start with the whole data as a cluster.
2. Select the cluster $C_i \in \mathcal{C}$ and feature/axis j with higher variance:

$$\underset{i,j}{\operatorname{argmin}} \sum_{x_n \in \mathcal{C}_i} \|x_{n,j} - \mu_{i,j}\| \tag{4.10}$$

3. Find the optimal splitting point f along the feature axis j that minimizes the variance of the two new clusters C_1 ($< f$) and C_2 ($> f$):

$$\underset{f}{\operatorname{argmin}} \sum_{x_n \in \mathcal{C}_1 | x_{n,j} < f} \|x_{n,j} - \mu_{1,j}\| + \sum_{x_n \in \mathcal{C}_2 | x_{n,j} > f} \|x_{n,j} - \mu_{2,j}\| \tag{4.11}$$

4. Return to 1 until the desired K clusters have been found.

Figure 4.8 shows an example of a VAR-Part initialization method in practice. It can be seen that the feature space (in this case a population with 2 features is given, which allows for a standard ploting) is divided hierarchically until $K = 3$ clusters are found. The centroids are then returned as the mean data points within each of the regions. This splitting criterion shares some similarities with the hierarchical partitioning that Random Decision Trees do. PCA-Part method follows the same procedure with an additional PCA-projection step to hierarchically split the data along its rotated main axis. This is adds robustness but we didn't find it useful in practice due to its extra computational cost (applying PCA) for a minimal performance increase.

Last, we propose an additional sampling criterion for completeness that is based on the same premise as SLIC. On the first iteration, cluster centres are sampled uniformly across the image grid (as in SLIC and similar algorithms). From there on, each of the regions to be splitted will be optimized separately at every iteration, keeping track of the clusters lying in that region in the previous iteration. The modifications needed to add to the SMURFS algorithm to keep track of the regions are listed as follows:

1. Initialize a uniform grid of K 2nd-order super-regions across the image.
2. Extract k (with $k \simeq M/K$ and M the desired number of superpixels) cluster centres uniformly distributed in **each of the regions**.
3. Split each of the regions j with k_j cluster centres.
4. Merge resulting regions into new 2nd-order regions, keeping track of how many components have been merged (in order to form new k_j values for each region).
5. Go back to step 3 and repeat until convergence. Each region will now have a different number of cluster centres to be sampled.

The algorithm will now be extremely efficient, as centroids don't have to be sampled at every iteration, but the joint optimization from subsubsection 4.1.1.1 can't be applied since the number of centroids for each region will be different and thus, graphs have to be constructed for each of the regions separately.

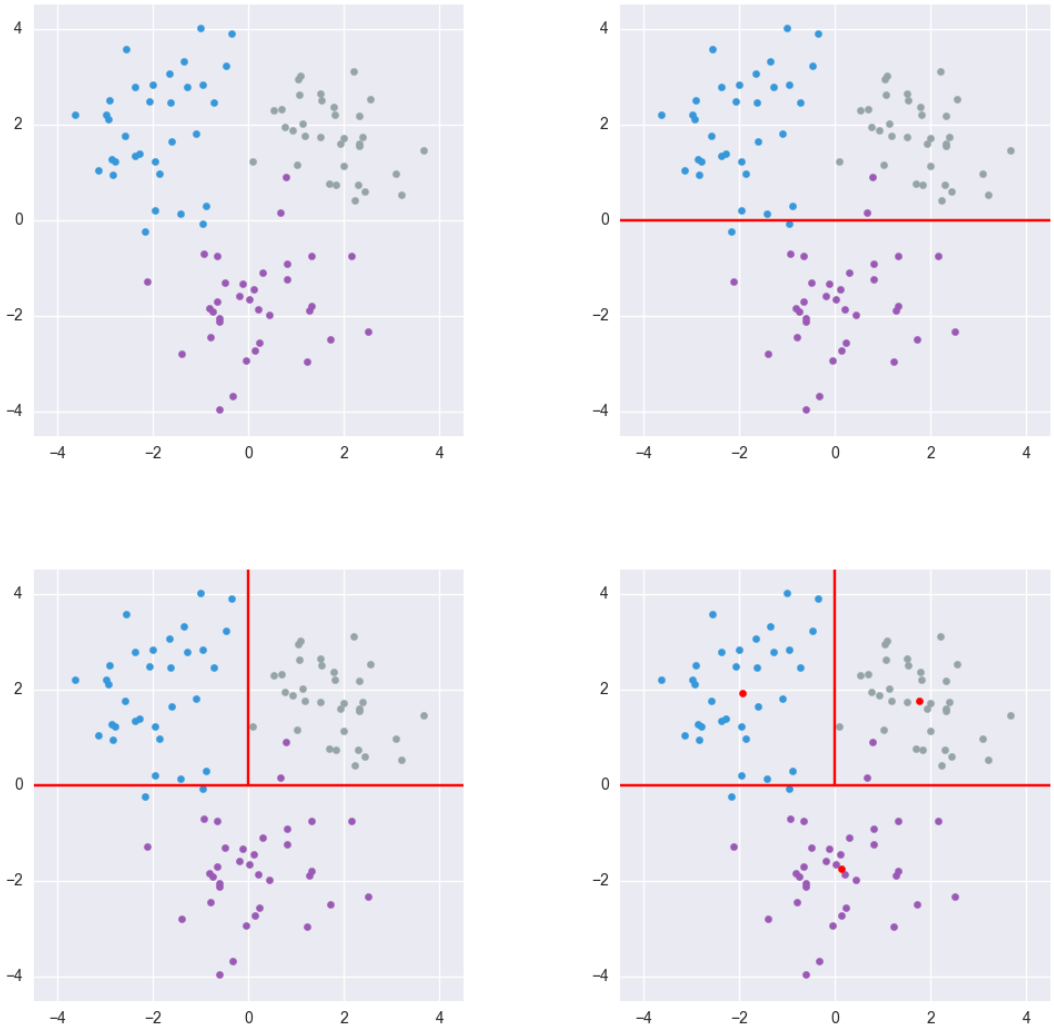


Figure 4.8: Example of PCA-Part initialization for k-means. For the initial data (already coloured with real cluster membership for easier visualization), the axis with largest variance is the y axis, which gets splitted along its optimal split point ($y = 0$). Of the two newly created clusters, the one with largest variance is the top cluster along the x axis, of which optimal split point is $x = 0$. The centroids (marked as red dots) are then calculated by averaging data points within the clusters.

4.4 Parameters of SMURFS

The main parameters of the algorithm are the number of desired superpixels M, the smoothness constant for the split step λ, the number of clusters to be sampled during the split step K and the scale μ that controls how similar regions have to be in appearance and size to be merged. Sensible defaults are listed in Table 4.2 for SMURFS:

Parameter	Description	SMURFS
λ	Controls strength of the denoising	1.0 [0.01, 5]
K	Number of centroids sampled per region	4 [2, 50]
μ	Ratio for the size of the regions to be merged (proportional to the average size N/M)	0.1 [0.01, 2]

Table 4.2: Default tunable parameters for both SMURFS. N indicates the total number of pixels in an image while M the desired number of superpixels. Brackets indicate expected parameter range.

The parameter K differs from HPCS (chapter 3) and SMURFS because regions are considerably smaller in SMURFS due to the whole image being considered as a single region in HPCS. Similarly, λ is 0.5 in HPCS as it has empirically proven to give better results due to HPCS considering colors from the whole image instead of just a region.

The following subsections will give insight into how the main parameters of the algorithm affect the result and how to interpret them to obtain the best result possible for different images.

4.4.1 Controlling the strength of the pairwise splitting term, λ

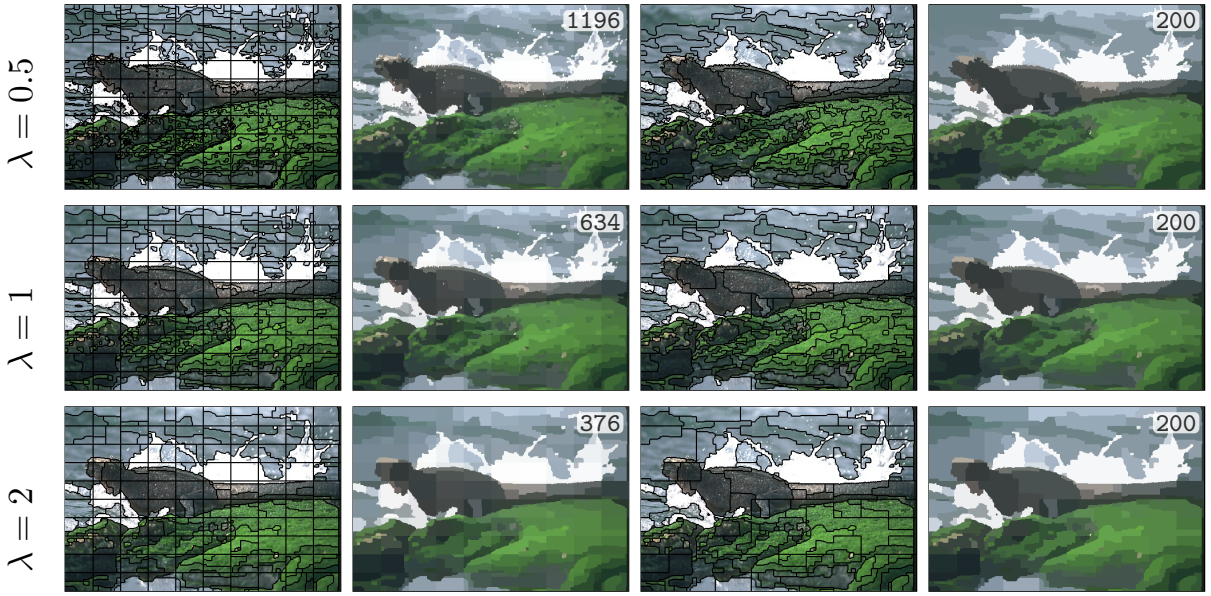


Figure 4.9: Result of applying SMURFS with different λ values to a textureless image. First two columns show the result after the initial split iteration while last 2 columns show final results. It can be seen that low λ values generate much detailed superpixel segmentation while preserving all the image boundaries. Large values, instead, tend to over-denoises the image too much, making it miss some important region boundaries.



Figure 4.10: Result of applying SMURFS with different λ values to a highly textured image. First two columns show the result after the initial split iteration while last 2 columns show final results. It can be seen that low λ values initially generate too many regions that are difficult to further reduce to just 200. Large values, instead, tend to remove textural noise from the image allowing for a much coarser segmentation while still retrieving image contours properly.

Similar results can be seen in Figure 4.9 Figure 4.10 for SMURFS than the one discussed above for HPCS. Lower values of λ in SMURFS tend to generate more regions, while larger values tend to remove texture or textural noise allowing for a much coarser segmentation results (even recovering windows in Figure 4.10 after surrounding textural noise has been ignored). Finding a balance between the two is subject to the subsequent application in which SMURFS is going to be used.

4.4.2 Controlling the number of centroids per region

SMURFS shows similar behaviour to K than HPCS, lower values of K giving more coarse results. However, since SMURFS does an iterative refinement, it is much less sensitive to the parameter K, working very well in practice even with the smallest $K = 2$. This is due to the fact that most mistakes made in a single splitting iteration are likely to be fixed in future iterations. It can be noted that on the first two columns of Figure 4.11 and Figure 4.12 that the K value has the most meaningful impact on the first splitting iteration, by the time the algorithm reaches its stopping criterion results are corrected. This also makes every iteration of SMURFS considerably quicker than a single iteration of HPCS, since



Figure 4.11: Result of applying SMURFS with different K values to an image. First two columns show initial results after partitioning the image, last two columns show final results after post-processing. It can be seen that the algorithm is much less sensitive to the K parameter, given good results for every value of K.

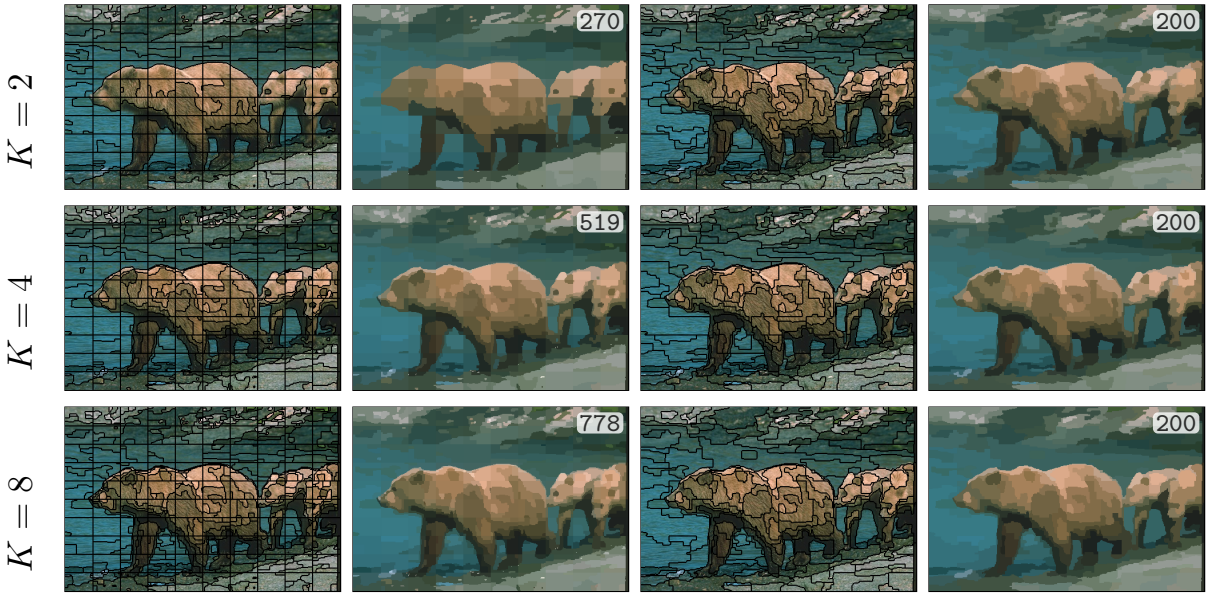


Figure 4.12: Result of applying SMURFS with different K values to an image. First two columns show initial results after partitioning the image, last two columns show final results after post-processing. It can be seen that the algorithm is much less sensitive to the K parameter, given good results for every value of K.

1. SMURFS only samples $K_{smurfs} < K_{hpcs}$ samples from each region, instead of the whole image as HPCS.

2. SMURFS disconnects edges of pixels not belonging to the same region, resulting in smaller graphs (see subsubsection 4.1.1.1).

3. Since $K_{smurfs} < K_{hpcs}$, obtaining the near optimal solution with FastPD is considerably faster.

4.5 Extensions of SMURFS

The previous section gave insight into parameter choosing for both HPCS and SMURFS. However, the mentioned parameters are only the main parameters of the algorithm that can be tuned to obtain different results, with the default values being tested to give quantitatively good results for a wide variety of image types. There are other parameters of the algorithm that affect the design and application choice of the superpixels and cannot be tuned on the fly. These parameters include: The similarity weights between

Parameter	**Description**	**HPCS & SMURFS**
D	Feature map in which the algorithm will operate.	Normalized $L \cdot a \cdot b$
Γ	Sampling function for the split step.	k-means initialized with VARPart
Q	Descriptor mapping for superregions.	Average feature D (Mean $L \cdot a \cdot b$ color) of all representative units inside each region).
W	(Split) Similarity between adjacent pixels.	Equation 4.12
W	(Merge) Difference between adjacent pixels.	Equation 4.15
$\phi_{p,q}$	Pairwise function of the labels of adjacent pixels	Potts or ℓ_1
τ	Threshold function for merging.	Proportional to μ, W

Table 4.3: Default design choices and mappings for HPCS and SMURFS. These are not tunnable on the fly and require programming. Different choices of these functions will result in different behaviours.

adjacent regions (pixels for standard SMURFS) are given as differences embedded in a Radial Basis Function (RBF) which is known to smooth similarities (see (Black et al., 1998) for different robust similarity functions), which is usually refereed as Gaussian Kernel in machine learning literature,

$$w_{p,q} = \exp(-\gamma \cdot \|D_p - D_q\|^2), \tag{4.12}$$

where γ can be manually tunned or approximated by the data

$$\gamma = \frac{1}{|\mathcal{E}|} \sum_{(p,q)\in\mathcal{E}} \|D_p - D_q\|^2 , \tag{4.13}$$

where a suitable γ is approximated from the data by averaging the differences of all the adjacent nodes of its representative graph (or RAG). It is to note that with a suitable γ fitted to the data the pairwise term ϕ could be replaced by a Potts model instead of the previously mentioned ℓ_1 potential,

$$\phi_{p,q}(f_p, f_q) = [f_p \equiv f_q], \tag{4.14}$$

where $[\cdot]$ is a logical operator that returns 1 if the predicate inside it is true or 0 otherwise.

Conversely, the weight W in the merging process refers to differences between adjacent regions and is measured as the squared Euclidean distance between their descriptors,

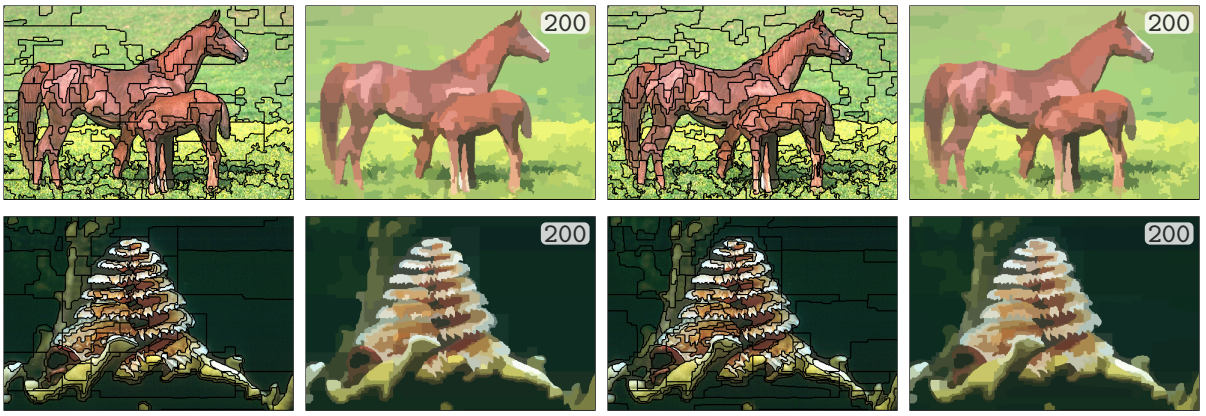
$$w_{p,q} = \|Q_p - Q_q\|^2 \tag{4.15}$$

where p and q are superpixels this time instead of pixels, and Q_p and Q_q are mean descriptors of all the pixels x (D_x) inside their corresponding superpixel.

The following subsections will discuss how to introduce prior information or sightly modify the algorithm to achieve different results best suited for particular applications. It is to note that these configurations will be given in the domain of SMURFS algorithm, but most of them also can be applied to HPCS.

4.5.1 Q-SMURFS: Quick SMURFS

The first obvious modification of SMURFS is to reduce its number of iterations to 1 to make it as quick as possible. This would reduce its accuracy but improve its computation time. Additionally, as noted in the previous section, the K parameter affects the quality of the initial regions after the first iteration of SMURFS, and although the parameter is not important for further iterations of SMURFS, it will improve the performance of SMURFS if applied to a single iteration. Thus, Quick SMURFS (Q-SMURFS) is defined as a single iteration of SMURFS with $K = 8$ as default, with results comparable to SMURFS as in Figure 4.13. At the expenses of losing some of homogeneity on the shape of superpixels, Quick SMURFS give relatively good reconstructions.

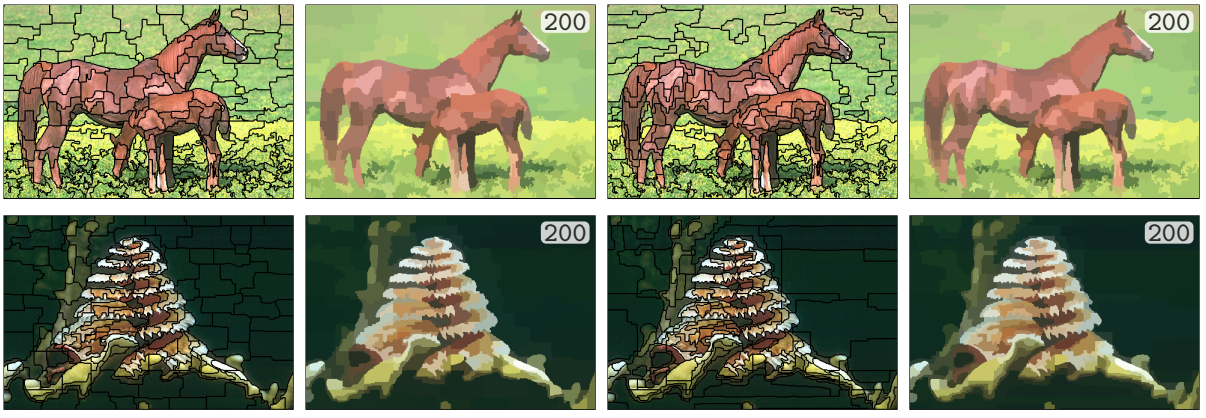


(a) Q-SMURFS (b) SMURFS

Figure 4.13: Q-SMURFS: A single iteration of Quick SMURFS is sometimes sufficient to achieve good results.

4.5.2 R-SMURFS: Regular SMURFS

Regular (or Compact) SMURFS is a result of adding spatial information to the feature mapping D (and thus, also to Q), resulting on superpixels of similar size and regular shape that gets clustered also in the spatial domain. Compactness is required superpixel feature in current state-of-the-art algorithms, and being able to generate superpixels of roughly similar size and shape is useful for certain applications (e.g. when descriptors need to be extracted from a superpixel, superpixels of regular shape often give better descriptors). To that end, and following a methodology similar to SLIC, regularity or compactness is added into SMURFS by means of defining the feature space as



(a) R-SMURFS (b) SMURFS

Figure 4.14: R-SMURFS: Regular SMURFS adds spatial information to the algorithm resulting in more compact (regular) superpixels in image areas with low textural information.

$$D = \{\gamma \cdot x, \gamma \cdot y, L, a, b\} \tag{4.16}$$

where x and y are normalized coordinates (in the $[-1, 1]$ range) of the pixels weighted by γ to control the relevant importance of the spatial versus the color information. $\gamma = 10$ as default encourages superpixels to form compact enough clusters where no texture is present, as shown in Figure 4.14. The compactness would be override by the subsequent merging operations. The merging criterion already takes into account spatial information by considering the position of each pixel (and thus, superpixels) in Q, which is derived by D that already has position encoded (see Equation 4.6). But, in order to fully take into account the superpixel regularity during merging, the merging criterion is modified to similarly take into account the distance between centres before merging by modifying the threshold function in Equation 4.8 as follows:

$$\tau(p, q) = W_{p,q} < \min(C_p, C_q) \quad \& \quad dist(p, q) < \frac{1}{\mu}\sqrt{\left(\frac{W}{\sqrt{M}}\right)^2 + \left(\frac{H}{\sqrt{M}}\right)^2}, \tag{4.17}$$

where W and H are image width and height respectively, M is the desired number of superpixels and $dist$ is the euclidean spatial distance between current components p and q. The second threshold expression accounts for the expected average distance between centroids given a uniformly distributed grid, which weighted by μ gives a tolerance threshold for two regions being merged. Thus, two regions are not likely to merge if their centres lay far from each other. The & states for a 'logical and' in the above equation.

4.5.3 B-SMURFS: SMURFS with boundary prior

Assuming that a good boundary map B scaled to $[0, 1]$ (1 representing edgeness and 0 the absence of an edge at a current pixel) and, preferably binarized and skeletonized, is given for the image to be segmented, the data weights in both splitting and merging can be modified to include the boundary prior. The weight in the split is replaced by

$$w_{p,q} = \exp(-\gamma \cdot \|D_p - D_q\|^2 \cdot \max(B_p, B_q)), \tag{4.18}$$

allowing the algorithm to decide whether two adjacent pixels should be together based on both their color differences and the edgeness. Because real world object don't usually have a pixel-wide boundary contour around them, pixels on both sides of the contour can be considered *boundary pixels*, however, (skeletonized) boundary maps represent boundaries by marking one of them. By weighting the maximum boundariness response B_p and B_q between every pair of adjacent pixels p and q with their color difference, it does give a good estimate of the likelihood of two adjacent pixels being in the same

object. This is, if p and q belong to the same object but q is marked as a boundary pixel, their small color difference (near 0) will cancel the boundary term, and thus have a big likelihood of being together ($\exp(0) = 1$).

Similarly, the weights in the merging step are adapted to measure the amount of boundariness between all the *boundary* pixels $bb(p, q)$ between two adjacent superpixels p and q,

$$w_{p,q} = (1 - \lambda_b) \cdot \|D_p - D_q\|^2 + \lambda_b \cdot \frac{1}{A} \sum_{(w,z) \in bb(p,q)} max(B_w, B_z). \tag{4.19}$$

here the first term is identical to the previously mentioned one, that is, the standard difference between colour descriptors of superpixels p and q. The second term introduces an additional term that measures the average response of the boundary map B along the boundaries of the two touching superpixels p and q. Every pair of w and z are essentially every possible pair of pixels laying in the boundaries between p and q (i.e. one laying in superpixel p and the other one on q). Last, λ_b is a parameter that controls how much do we trust the boundary map over the color difference of superpixel descriptors, set as 0.5 by default.

4.5.4 C-SMURFS: SMURFS with Contextual prior information

Similarly, contextual prior can be added to the superpixel construction in the same way if higher level objects are known in advance, this is, a higher level labelling $\boldsymbol{t}$ exists for the image or class probabilities $\boldsymbol{c}$, witch $\boldsymbol{c}$ a $N \times C$ matrix with N image pixels and C the number of classes, are available for the current image.

The priors are introduced similar to the boundary priors in B-SMURFS. In the first case, where a ground truth labelling is known (because it has already been segmented by automatic or manual approaches), super-regions can be hard-constrained not to go from one class label to another by modifying weights during the split as

$$w_{p,q} = \exp(-\gamma \cdot \|D_p - D_q\|^2) \cdot [t_p = t_q], \tag{4.20}$$

where the similarity will be truncated to 0 if the ground truth labels of pixels p and q do not match, and modifying the merging weight as

$$w_{p,q} = \|D_p - D_q\|^2 \cdot \left(\sum_{(w,z) \in bb(p,q)} [t_p \neq t_q] \right). \tag{4.21}$$

making the difference larger if boundary pixels are in different class labels, thus, preventing the merging.

In the second case where class probabilities are given it is sufficient to replace the binary function $[\cdot]$ with the intersection distance between ℓ_1 normalized vectors:

$$intersection(\boldsymbol{c}^p, \boldsymbol{c}^q) = \sum_i \min(c_i^p, c_i^q) \tag{4.22}$$

which returns 1 if and only if both histograms (class probability distributions in this case) are identical. During merging $1 - intersection(\dots)$ accounts for the priors in a similar manner.

4.6 Experimental Comparison of Superpixel Algorithms

4.6.1 Qualitative Evaluation

We compared SMURFS as a superpixel algorithm to other top performant algorithms of the literature Figure 4.15, Figure 4.16, Figure 4.17 and Figure 4.18 show comparisons of SMURFS to other algorithms. Figures Figure 4.15 and Figure 4.16 show comparisons as before to visualize the quality of SMURFS. However, to better appreciate the power of SMURFS to better represent data with less representative units we adopted the mean colour visualization in Figure 4.17 and Figure 4.18. Essentially, Figure 4.16 and Figure 4.17 are the same superpixels with different visualization, reconstructing the original image only using the mean colour of the generated superpixels. It can then better appreciate the power of SMURFS in Figure 4.17 and Figure 4.18 as with the same 200 superpixels more visual information is recovered by SMURFS. Compact superpixels are good at extracting high level objects, or when images are simple and have homogeneous background, however, in complex backgrounds they are not able to recover much information from the image, missing some important boundaries. LSC is supposed to be compact, however under complex scenes it behaves more adaptive and creates irregular superpixels, producing also good results. In images like the 3 cyclist, the air plane or the buildings it the superiority of SMURFS can be appreciated. This validates our premise that irregular superpixels are better than compact superpixels when the final aim is to use them for image segmentation.

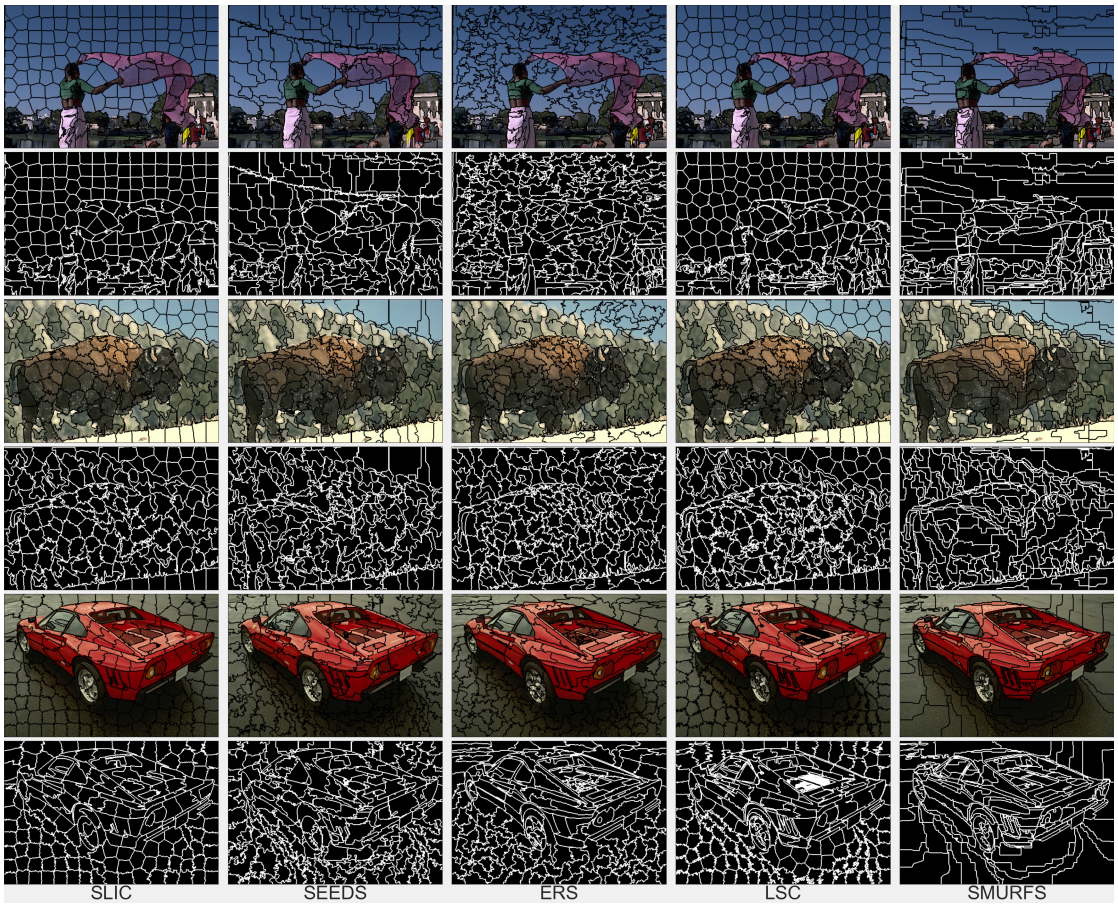


Figure 4.15: Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels.

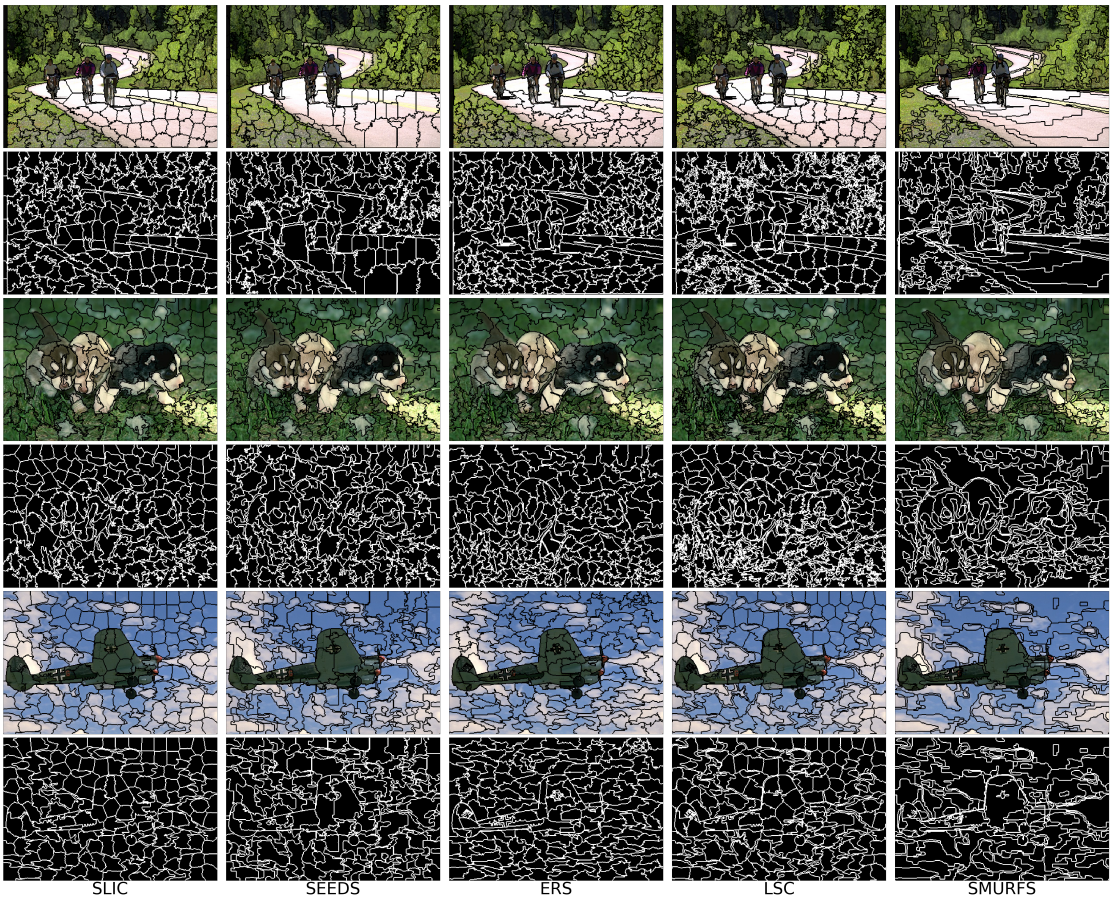


Figure 4.16: Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels.

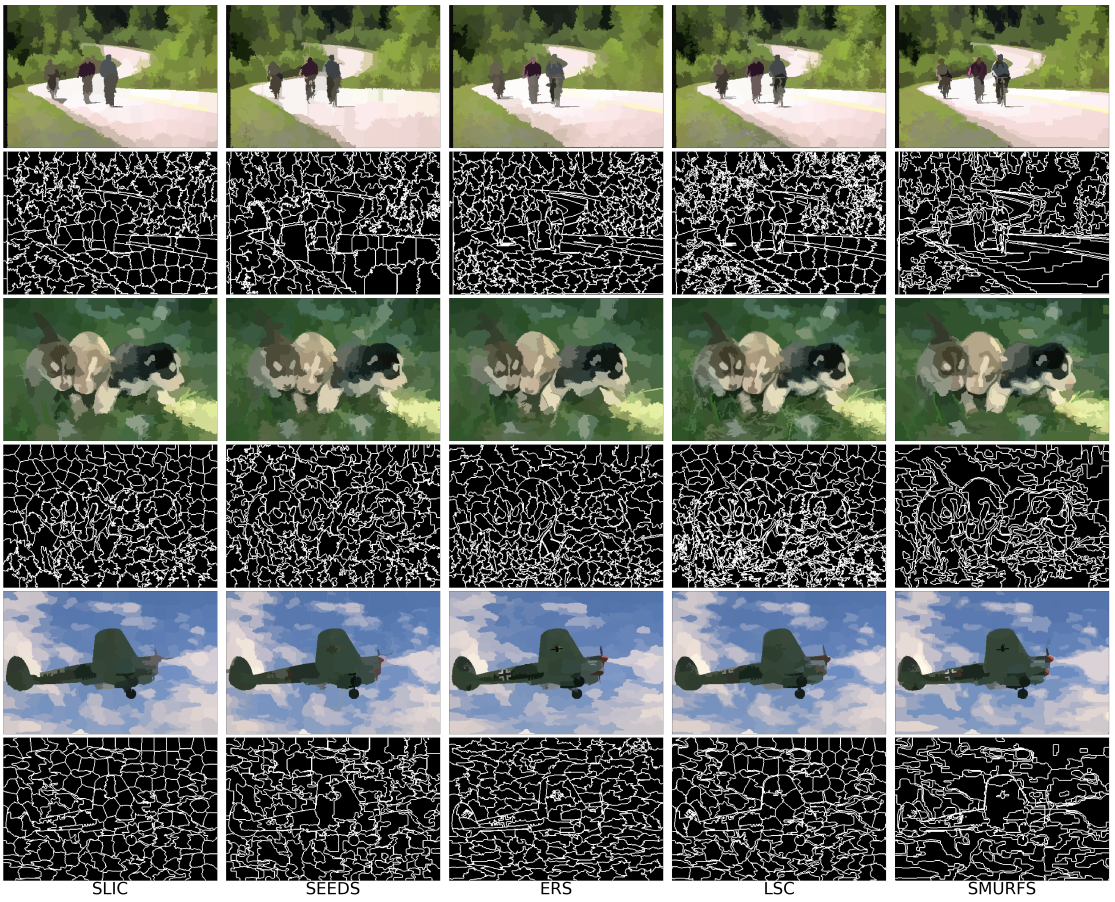


Figure 4.17: Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels (with boundary visualization).

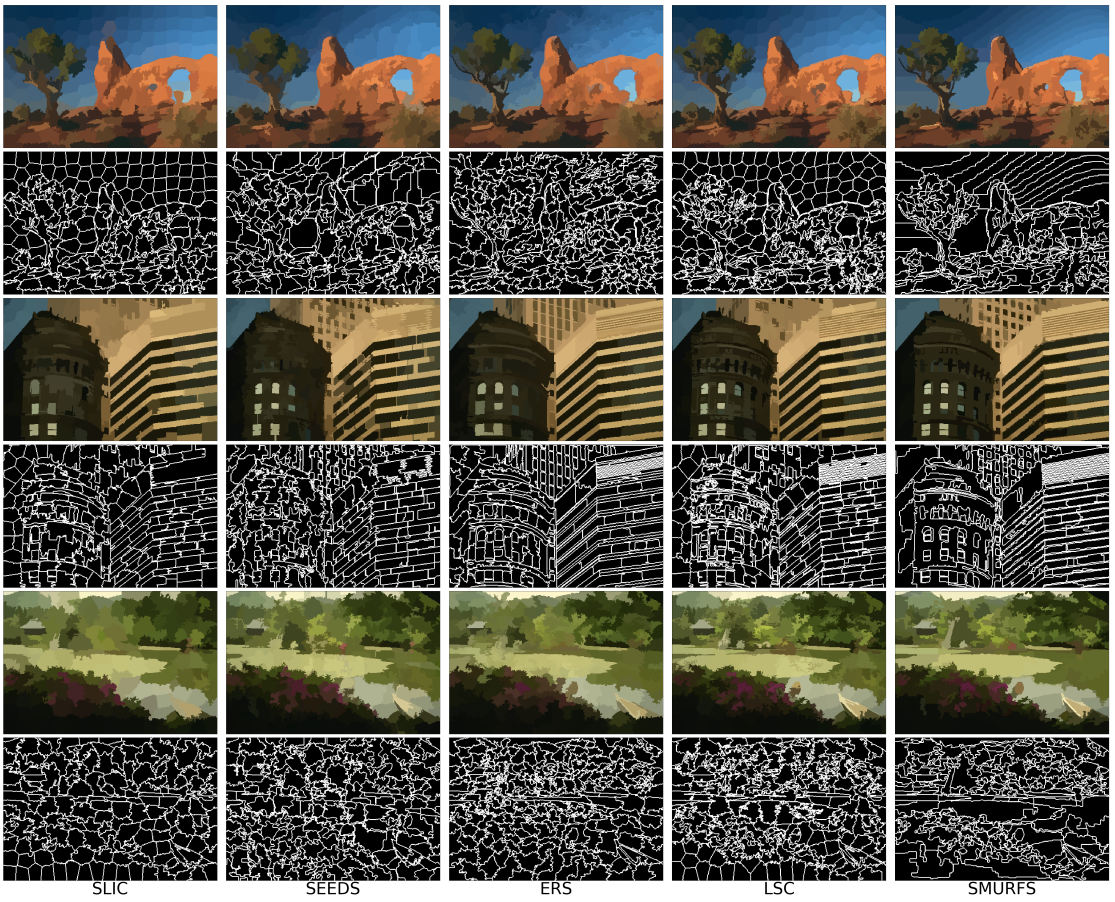


Figure 4.18: Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels (with mean color visualization).

Last, Figure 4.19 shows example comparison of boundaries and reconstructions of 7 different algorithms.

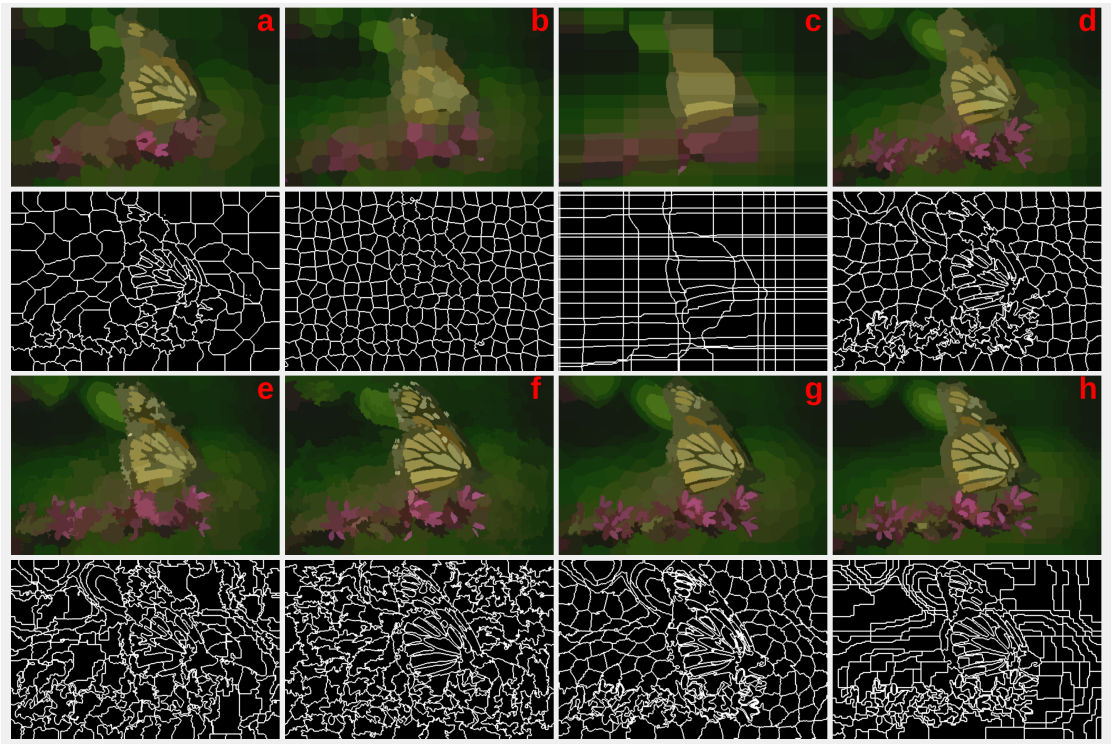


Figure 4.19: Overview and comparison of SMURFS superpixels to state of the art with $M = 200$ superpixels. (a) enteOpt, (b) Turbopixels, (c) Superpixel Lattices, (d) SLIC, (e) SEEDS, (f) ERS, (g) LSC, (h) SMURFS.

4.6.2 Quantitative Evaluation

Similar to HPCS, SMURFS was quantitatively evaluated in BSD500 dataset against top performant algorithms in the literature. Figure 4.20 shows benchmark results of SMURFS. SMURFS outperforms the state of the art in every metric, with the exception of LSC in boundary recall for less than $M = 200$ superpixels. This verifies our finding during the qualitative evaluation that SMURFS better captures image information. Achievable Segmentation Accuracy (ASA) represents how well the image would be segmented if using superpixels. It can then be understood that ASA and the mean-colour visualization of superpixels are very closely related. If a superpixel reconstruction visually shows better quality and texture detail, it is likely that its ASA will be higher. Similarly, CUE is inversely proportional to ASA, also verifies the above findings. We believe that the fact that the Boundary Recall (BR) is smaller than the one of LSC for small number of superpixels has to do with the variability of the annotations and the regularity of LSC. SMURFS tends to follow image boundaries too closely, sometimes better than human annotators do, which would have caused to fail in some images. On

TP	EneOpt	SLIC	SEEDS	ERS	LSC	*PCS*	*SMURFS*
> 5s	> 5s	0.33s	0.25s	1.60s	0.67s	1.10s ($K = 16$)	0.75s ($K = 2$)
						1.79 ($K = 32$)	1.51s ($K = 5$)

Table 4.4: Computational times for some superpixel algorithms.

the other hand, the grid-like regularity of LSC might have made it *touch* some impossible to detect boundaries just by being in aligned with a regular grid, boundaries that otherwise SMURFS would have ignored due to their weakness.

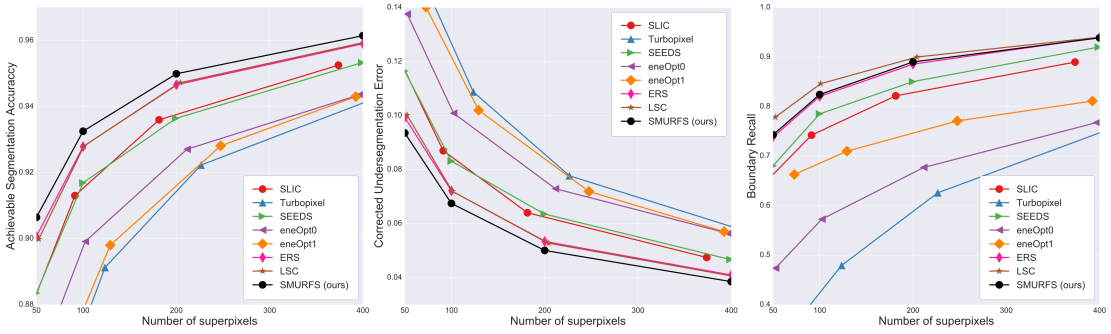


Figure 4.20: Quantitative evaluation of SMURFS.

On the computational aspects, both HPCS and SMURFS are computationally more demanding than SLIC, LSC or SEEDS, but have similar computational time than ERS and are orders of magnitude faster than other MRF approaches such as EneOpt1 and EneOpt2. Table 4.4 shows average computational times of superpixel algorithms for images in the BSD dataset:

4.7 Other applications and properties of Super-Regions

Here we will introduce few interesting applications for the Super-Regions with SMURFS. Some of them have already been discussed along the previous chapters, but some specific examples here will be presented.

4.7.1 Better reconstructions for better segmentations

Super-Regions generated by SMURFS can better represent the underlying image. Their adaptive boundary adherence is due to a multi-scale refinement, allowing the algorithm to model both long-range and local information. It allows to over-segment complex areas of the images with more superpixels while leaving homogeneous areas with fewer partitions. This allows the algorithm to recover more detail from the images with less

superpixels than other algorithms. This is, compact superpixels like SLIC, would have to create much more superpixels ($M > 1000$) for a 481×321 to recover the same level of detail than SMURFS with $M = 200$.

To illustrate how SMURFS can help better representing the image with less representative units, Figure 4.21 shows an example or an image from the BSD500 dataset reconstructed with increasing amount superpixels. It can be noted that as the number of superpixels increases, the images start to become undistinguishable from a distance.



Figure 4.21: Image reconstruction using SMURFS with $M = \{100, 200, 400, 800\}$. *It can be seen that as the number of superpixels increases, SMURFS recovers more detail from the images and these become difficult to distinguish from a distance.*

One of the goals of this thesis is to create super-regions that better represent the image so that segmentation algorithms working on the superpixel level can better segment the image without having to create huge amount of superpixels. To date most of the approaches using superpixels as source have been designed towards object detection or classification, where exact boundaries are not of particular interest. However, when the posterior task is segmentation, SMURFS should provide better boundary detail with less superpixels to make further algorithms not only faster, but also more accurate.

4.7.2 Super-Regions as post-processing for superpixel algorithms

SMURFS can be used as post-processing for other superpixel algorithms. HPCS could be used to underlay superpixel algorithms as the 1st-order super-regions, and the use HPCS on top of that to generate over-segmentations with fewer super-regions as 2nd-order super-regions. SMURFS can not only user other over-segmentations as input to post-process them, but can also use them as initialization. SMURFS is by default initialized with a regular grid, but providing the result of another superpixel algorithm as initialization is straight forward.

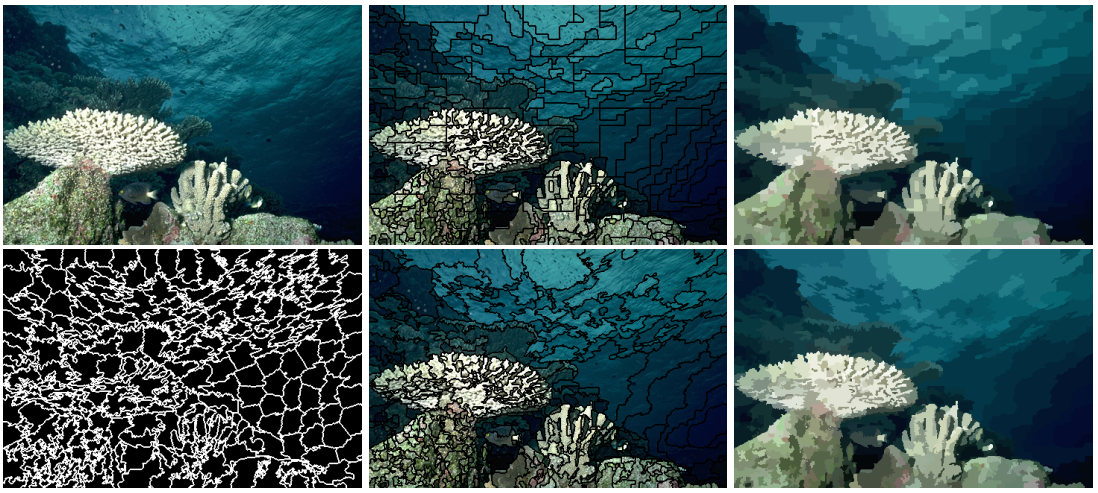


Figure 4.22: Image reconstruction using SMURFS ($M = 400$) with the over-segmentation of LSC ($M = 200$) as input. Top row shows from left to right: original image, boundaries of SMURFS applied to the original image, and reconstruction using SMURFS. Bottom row shows from left to right: boundaries from LSC over-segmentation used as an input and boundaries and reconstruction of a single iteration of SMURFS using LSC as initialization.

It can be seen from Figure 4.22 that by using the over-segmentation of LSC as initialization for SMUFS (bottom row), the SMURFS result gets biased and returns more accurate boundaries and reconstructions. In this case we used an over-segmentation with more superpixels $M = 200$ than desired $M = 400$ as initialization to improve the initial condition given by our grid initialization. It can be seen that using LSC helps SMURFS recover more detail from the image (although overall segmentation quality keeps the same). Similarly, SMURFS could be used as HPCS before to generate less superpixels from a given over-segmentation framework with no change at all to the algorithm.

4.7.3 Interactive refinement of Super-Regions with SMURFS

Another special usage of SMURFS is to interactively refine super-regions. It has been shown in subsection 4.2.2 that SMURFS works by refining super-region hierarchies iteratively. It has also be shown in subsubsection 4.1.1.1 that although SMURFS uses a single cut approach to partition all the regions at once, for large datasets solving each of the regions separately is much more efficient and can be done completely in parallel. Thus, it is straightforward to extend super-regions to an interactive framework for quality refinement. We implement this in a user driven interactive framework by using scribbles as commands for super-region updates:

1. At a given level, two regions can be merged together by annotating them with a scribble (Figure 4.23). This effect propagates HSplit (algorithm 9) from the given hierarchy level up to the top level.

2. A region can be selected to be further splitting by marking it with another scribble (Figure 4.24).

3. Further markers can be given to split a region with user-given points as sampled cluster centres, which would allow for better user-driven refinement.

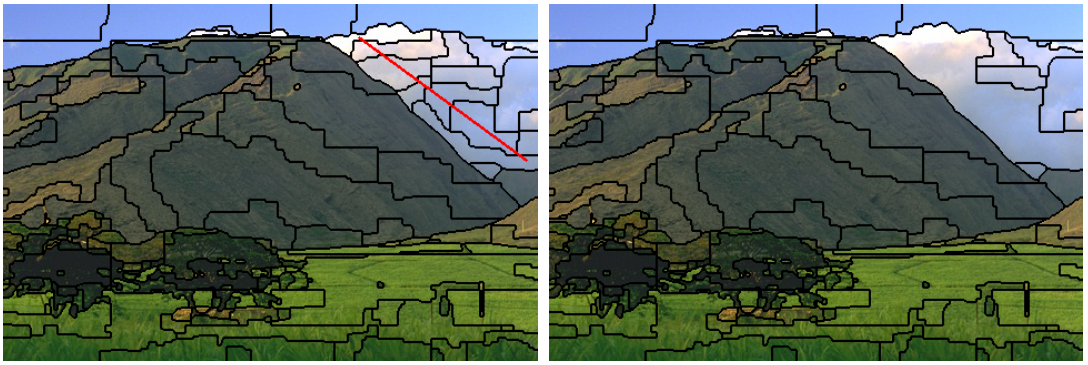


Figure 4.23: Interactive merging of super-regions. Red marker on the clouds of the left image indicates that the regions should be merged together. The merge operation is propagated through the hierarchy if needed, resulting on the right image with merged regions.

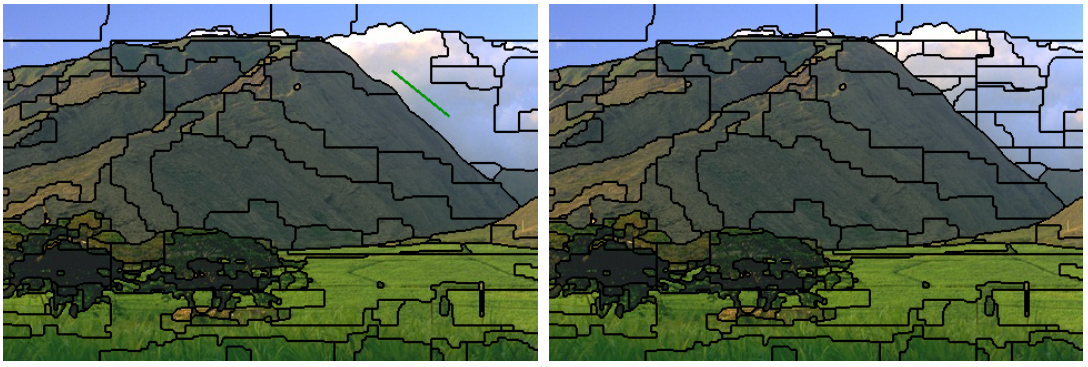


Figure 4.24: Interactive splitting of super-regions. Green marker on the clouds of the region on the left indicates that the region should be further splitted. The HSplit algorithm of SMURFS is then applied to that single region to further split it into new regions.

Figure 4.23 and Figure 4.24 show simple examples of region merging and splitting interactively. Though these examples are trivial and simple, interactive refinement of super-regions can be used as a powerful tool for fast interactive image segmentation. As

the boundaries of super-regions already adhere object boundaries, whole objects could be extracted with couple of clicks. Similarly, wrong boundaries of objects were a single superpixel lays both inside and outside of the object could easily be fixed by marking both outside and inside with different colours and using the points as cluster centres (Θ^j) for the split procedure of SMURFS.

4.7.4 Super-Regions for biological volume over-segmentation

Another interesting property of both our algorithms, HPCS and SMURFS is that their extension to 3D is trivial. As both the split and merge procedures are formulated in a graph-based framework, extension to 3D only changes the 4-connected pixel neighbourhood to 6-connected or 18-connected voxel neighbourhood. With this minor modification the algorithm is able to work with 3D data inherently. Extension to ND becomes then straightforward. The main limitation of the algorithms, though, is the computational complexity of the MRF optimization process. Solving MRF problems on the voxel grid of a large 3D dataset is orders of magnitude slower than on a small image grid. This is, for a $1000 \times 1000 \times 200$ volume, solving a MRF at the voxel level could take minutes to hours (depending on the number of K sampled for each region). This makes HPCS impractical for extracting supervoxels. However, once supervoxels are extracted (by another method), HPCS can be applied on top of the 1st-order super-regions (supervoxels) to extract further regions on the hierarchy (e.g. 2nd-order super-regions, or megavoxels as we call them).

SMURFS is different to HPCS, in the sense that the split procedure only requires local MRF optimizations. Thus, the solution to the splitting of all the regions in a volume can be separated in smaller sub-problems, one per region, and solved them separately. This makes a solution for SMURFS in 3D datasets still feasible as the computational complexity is bounded by the size of each of the regions. Thus, for a region of size $100 \times 100 \times 50$ as an example of the size that individual regions could have, the MRF solution is very efficient, taking under a second to be computed. We add an additional constraint to the SMURFS' merge operation to stop merging regions if their size is larger than a predefined value, to prevent SMURFS from gathering VERY large homogeneous areas in just one region, which would make the split operation very computationally expensive.

Figure 4.25 shows a 3D dataset (top row) and its denoised version using Total Variation (more details on Appendix B). Figure 4.26 shows an example of SMURFS applied to a 3D dataset of shape $946 \times 946 \times 100$. For visualization purposes 3 different slides

are shown at slice intervals $25, 50 and 75$, but supervoxels created this way maintain 3D shape.

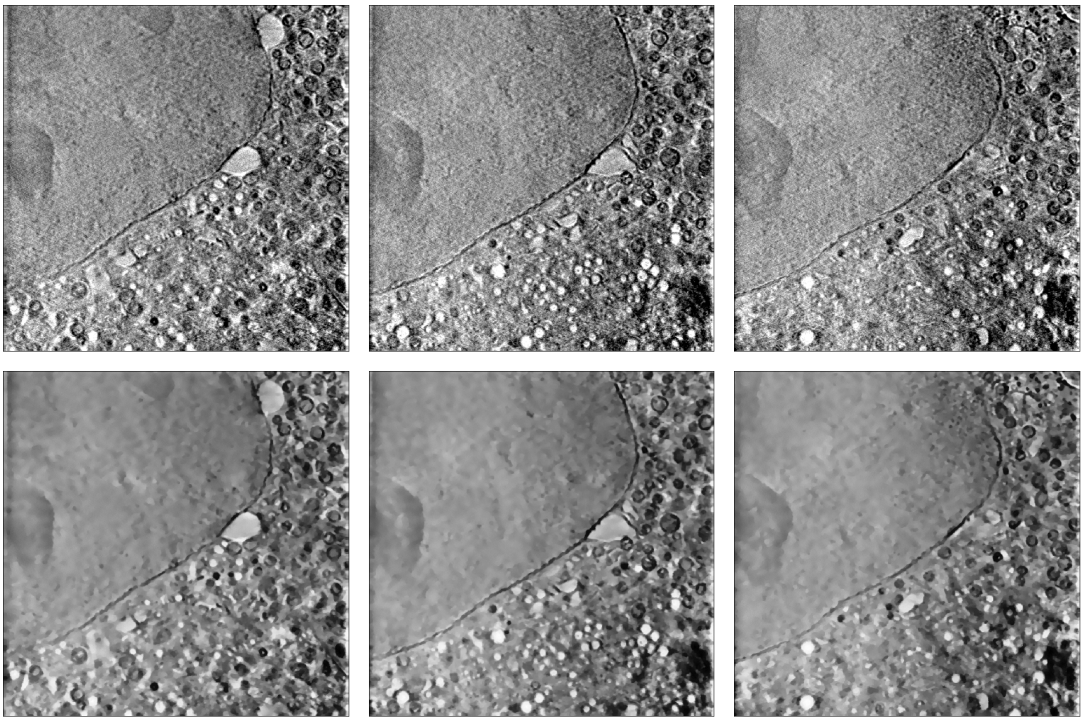


Figure 4.25: Top row shows slices 25, 50 and 75 respectively of a ROI of size $946 \times 946 \times 100$ *of a cryo-SXT dataset. Bottom row shows the denoised images using Total Variation.*

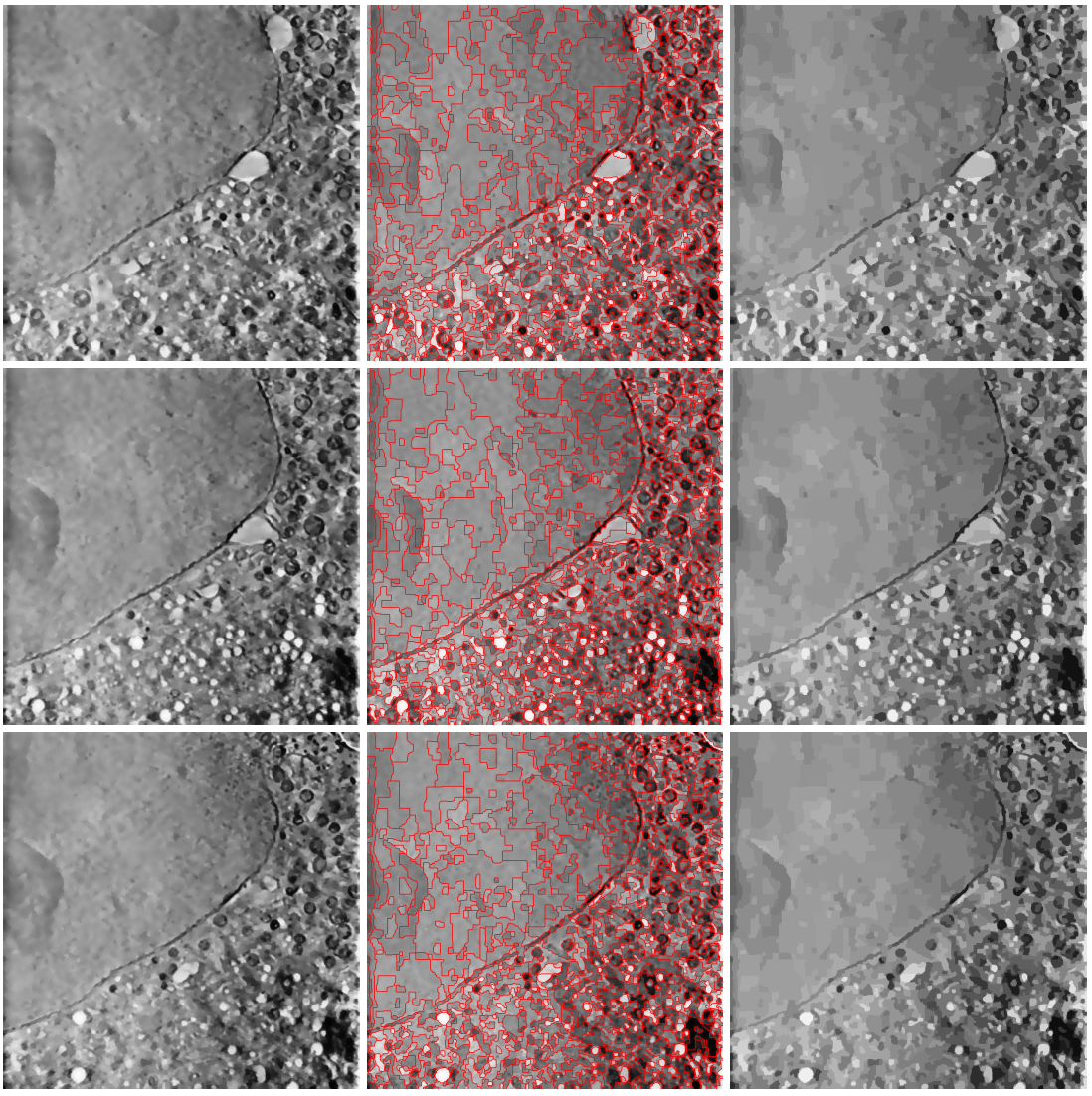


Figure 4.26: SMURFS supervoxels applied to a 3D dataset. Rows show slices 25, 50 and 75 respectively of a ROI of size $946 \times 946 \times 100$ *of a cryo-SXT dataset.*

SMURFS has been applied directly to the denoised TV images that better represent data while removing noise. To handle biological datasets with SMURFS, SMURFS was set to extract supervoxels of approximate shape $30 \times 30 \times 30$, with initial and maximum size regions set to $80 \times 80 \times 50$ to keep the computation efficient. The computational time for 3 iterations of SMURFS was about 9 minutes.

For comparison purposes we computed SLIC supervoxels also on the same data, which CPU code took about 3 minutes to converge. This shows that although SLIC is still faster than SMURFS, SMURFS' sub-problem formulation makes its segmentation efficient and comparable computationally to the one on SLIC, as it is still in the same order of magnitude, as opposed to the full MRF formulation of HPCS which would have taken

up to few hours to converge. Figure 4.27 shows a comparison of SMURFS 3D with SLIC 3D. We show different variants of SLIC set to extract $30 \times 30 \times 30$, $20 \times 20 \times 20$ and $10 \times 10 \times 10$ supervoxels respectively. It is to note that for SLIC to achieve the same representation detail than SMURFS $10 \times 10 \times 10$ supervoxels are needed, which makes for up to $\frac{946 \times 946 \times 100}{10 \times 10 \times 10} = 89491$ supervoxels, while SMURFS only needed approximately $\frac{946 \times 946 \times 100}{30 \times 30 \times 30} = 3314$.

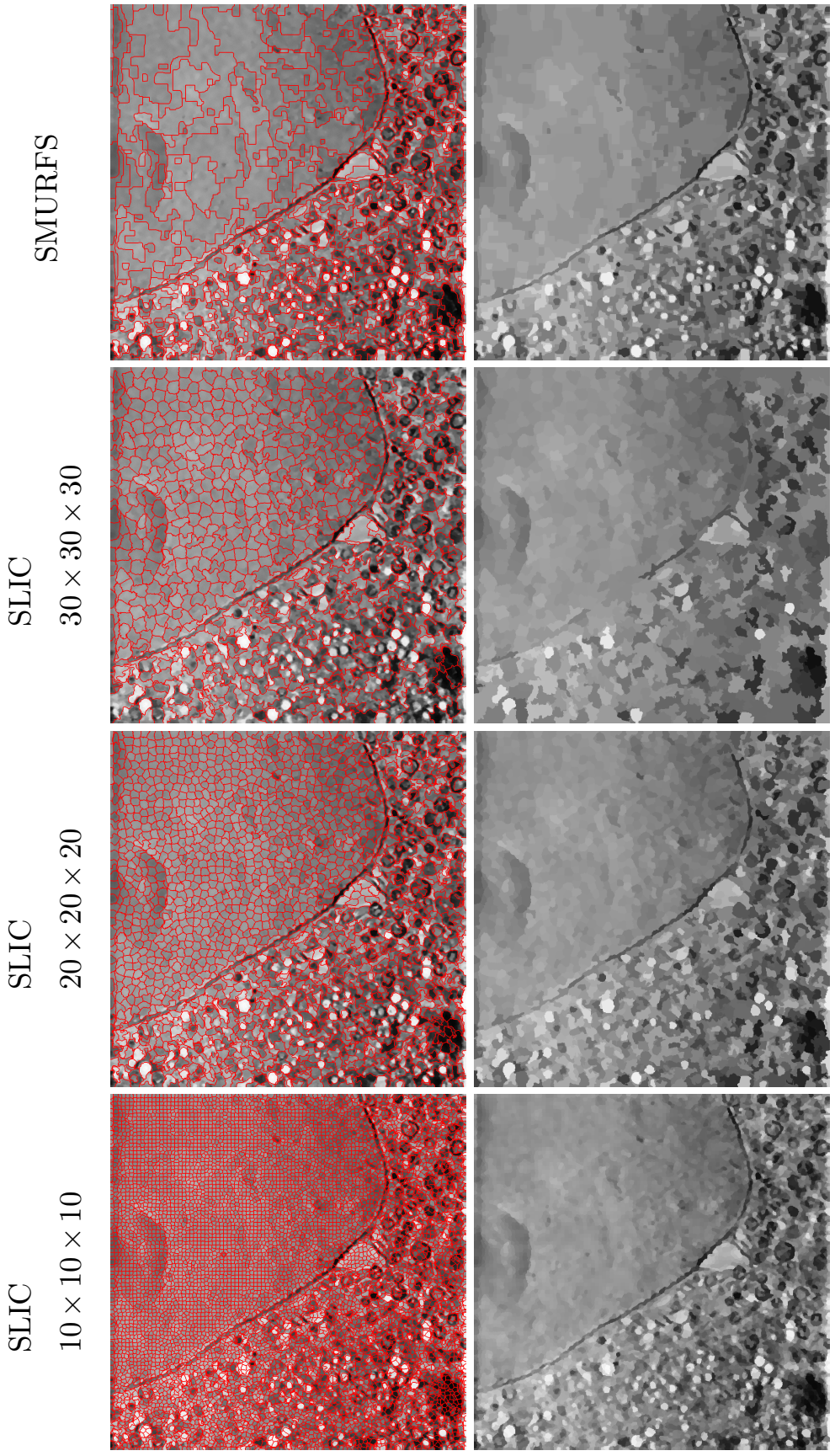


Figure 4.27: Comparison of SMURFS supervoxels with SLIC supervoxels. First row shows SMURFS supervoxels, boundaries and reconstructions. Next three rows show the result of SLIC.

4.8 Discussion

In this section we refined and extended the concept of Super-Region Hierarchy, first introduced in the previous chapter, formulated as over-segmentation layers where all the elements of a hierarchy are formed of similar elements from the previous level. We proposed two algorithms, HPCS and SMURFS which use a set of proposed split and merge operations to extract the so-called Super-Regions. Both algorithms are able to extract superpixels of good quality but are also generalized to further levels in the hierarchy.

Here we have presented the theoretical grounds of our framework, which aims to dramatically reduce the number of representative elements needed to represent an image without too much loss in detail. The next chapter will add many more qualitative and quantitative evaluations and comparison to better understand the strength of these super-regions, as well as discussing some other applications aside from just extracting superpixels.

As a recap of both super-region algorithms, before continuing to the next part of the thesis:

Then extensions of both HPCS and SMURFS were proposed. We have shown how the main parameters of the algorithms affect the output result. The two algorithm proposed show state-of-the-art superpixel quality, both quantitative and qualitatively. SMURFS have been shown to be superior in performance to other superpixel methods, at the expenses of computing less regular superpixels. However, as we have seen with the reconstruction experiments, regular superpixels are not always ideal, and for image segmentation task irregular superpixels better adhere the object boundaries.

We have also shown that our 2 superpixel algorithms are much more than *just* superpixel algorithms. Their graph-based formulation makes them very easily generalizable to hierarchies, which introduced hierarchical super-regions. We have shown that super-regions can be used not only to compute superpixels, but also to post-process superpixels and allow to represent images with less representative units while maintaining accuracy.

A novel idea of interactive super-region refinement has also been proposed. By refining hierarchical over-segmentations whole objects could be segmented by drawing scribbles to join or refine current regions. Based on this idea, SuRVoS (chapter 7) allows an user to interactively segment biological volumes by using a two-level super-region hierarchy. SuRVoS does not recompute super-regions based on user input, but it does maintain a single labelling at the voxel level, and then allows the user to easily annotate hundreds or thousand of voxels with a single click by using the underlying supervoxels and

megavoxels (2nd-order super-regions), similar to the super-region refinement presented in subsection 4.7.3.

Last, we presented a very straight forward 3D extension of SMURFS, with the only modification of adding a 3D neighbourhood system (6/18N) to the initial layer and a maximum size constrain for a region of $80 \times 80 \times 50$ voxels, thus keeping the worst case scenario of the MRF optimization for each of the regions very efficient.

Overall, super-regions seem to be very promising and could be used in multiple applications, from hierarchical segmentation to interactive segmentation (through region refinement). A particular work very directly related to hierarchical regions is Associative Hierarchical Random Fields (Ladickỳ et al., 2014), recent work on hierarchical MRF models for semantic segmentation that could benefit from super-regions extracted from SMURFS instead of their stacked Meanshift Segmentations.

Last, SMURFS and HPCS have only been evaluated as superpixel algorithms, by computing 1st and 2nd order super-regions and jointly refining them (in the case of SMURFS). However, we have shown that theoretically arbitrary level or hierarchies can be computed, each one with increasing level of abstraction. There is currently no available dataset to evaluate the quality of such hierarchical partitioning, as the proposed approach is novel and unique in that sense, thus it was out of the scope of this chapter to quantitatively evaluate it.

Following sections will show how biological datasets are naturally hierarchical, and how they can benefit from such approaches. We think that hierarchical parcellation of datasets, however, has many more applications than listed here, and are possible extensions to the work presented here, such as using the user input to merge multi-scale hierarchies into single segmentations interactively.

Part II

Applications of Super-Regions to Biological Volume Segmentation

5 Introduction to the Segmentation of Biological Volumes

This chapter will try to serve as a smooth transition from standard 2D images to the intriguing and complex world of sub-cellular biological imaging (briefly introduced in section 1.1). Biological imaging is a very broad research area covering everything from 2D images of animals, medical images of human patients or brain maps, to the analysis of sub-cellular organelles and particles. In the following chapters we will focus on the specific area of sub-cellular imaging, where biologist study whole cells, sections of them, organelles, or even smaller structures such as proteins and macromolecules trying to understand and characterize different processes or structures taking place in cell functions, often under the presence of external pathogens or bacteria making it function abnormally. Figure 5.1 shows an image of different imaging resolutions and elements being studied in the field of structural biology. It can be seen that even within this field, structural biology works at very different scales, under very different imaging modalities to answer very different questions.

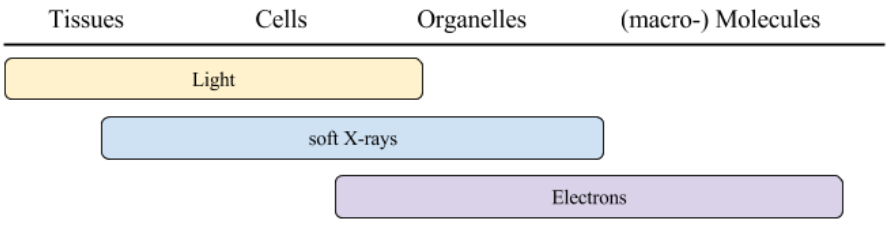


Figure 5.1: Broad overview of the resolution of different imaging techniques. Exceptions might apply using special techniques such as super-resolution light microscopy.

Excluding the higher resolution (smaller scale) fields (protein, molecule or crystal characterization), often refereed to as single-particle imaging where the aim is to reconstruct a 3D model of very small particles as accurately as possible to unveil their structure and properties, segmentation is a crucial step to extract quantitative data in every other scale of structural biology. However, the segmentation process is not very well defined and varies from experiment to experiment. Focusing on cells, either whole cells or sections of cells (referred to as samples from now on), there is a wide variety of imaging modalities and sample preparation techniques that generate very different types of images at lower or higher resolution. Depending on the biological question trying to answer, the sample will be imaged with different techniques, each of them with their own pros and cons as detailed in section 1.1. Each of the techniques will then generate a volume at different resolutions, affecting not only the size of it, but also the contrast, noise, visibility and appearance of the organelles present within it. For example:

- Cryo-soft X-ray Tomography (Cryo-SXT) will be able to image whole cells, at the expense of having worse resolution than EM techniques, making structures such as micro-tubules or endoplasmic reticulum no longer visible. Additionally, the cell has to be plunge frozen prior to imaging, to preserve it in a near-native state, to fix it and prevent it from moving, and to reduce radiation damage.
- Scanning Electron Microscopy (SEM) will scan the surface of samples, giving high resolution structural information, but in order to scan a section of the cell fully and extract a 3D volume from it, milling (or sectioning) is needed. Milling is the action of removing layers from the cell in order to image the surface of the next layer, until the whole section of the cell is imaged. In order to image it, it has to be previously fixed, either with cryogenic or chemical fixation. Each of them has different benefits and will affect the contrast and the structure of the imaged sample. Additionally, different milling techniques, such as Serial Block Face (SBF) or Focused Ion Beam (FIB) variants, will also affect the quality and size of the recovered volume and include different damage artefacts.
- Transmission Electron Microscopy (TEM) can be used to extract a tilt series of a very thin sample (under $100nm$) and reconstruct it to a 3D volume. Because of the quick absorption of the electrons the sample is limited to be very thin, but allows for the higher resolution of the three techniques mentioned. In order to collect a TEM volume, the sample has to be fixed and milled, as with SEM, introducing the same limitations, variations and artefacts.

There are even more factors in play that alter not only the resolution of the volume (e.g. the type of microscopy used, the resolution of the zone plate or the size of the detector

within the microscopy), but also structure and appearance of the sample (e.g. different fixation methods and contrast agents used) and the artefacts introduced while imaging it (e.g. radiation or milling).

From the above, the huge challenge that is the segmentation of sub-cellular volumes should be clear. The biological aspect makes the task very challenging, but the computational aspects of it make it raise the challenge further, as the recovered volumes are often in the order of Gigabytes (it can go up to Terabytes, but those cases wont be contemplated in this thesis). Because of the underlying complexity of the data (both biologically and computationally), and the methods used to capture it, segmentation plays a crucial role in quantifying the information just captured, yet it is mostly used as a tool to process the data rather than an inherent part of the imaging process. The data is so challenging and big, that it is often imaged with a very concrete purpose in mind, such as to study the shape and size of the microtubules, the structure or curvature of different membranes, or the shape and size of mitochondria. This usually involves (semi-)manual annotation of the objects of interest exclusively, which takes a considerable amount of time (from days to weeks, or even years in particular longitudinal studies where whole brain cells are mapped (Markram, 2012)), leaving the rest of the volume unprocessed and unanalysed. The amount of data generated by current bioimaging techniques is currently overtaking our ability to efficiently analyse it. In order to analyse a biomedical volume, expert knowledge is required to visually explore the volume and manually tag different areas/sections of it for measurement. This manual segmentation labour is tedious, error prone and time consuming. Due to the complexity and variability of the underlying biological data, different approaches towards quantification of the datasets are taken. The importance and cost of the biological data and the very selective (and minimal) segmentation performed to extract quantitative information prevent volume segmentations to be publicly available. At the same time, the lack of public segmentations force biologists to perform new segmentations from scratch, for every single volume, as very rarely the same type of volume is analysed twice with the same goal on mind (by the same biological team). Thus, this results in tons of data and their respective segmentations end up wasted and abandoned, preventing machine learning approaches to make their way into the sub-cellular biological world. The following chapters will try to make a step towards a new direction, introducing novel techniques for quick segmentation of volumes, with the aim and hope of introducing procedures towards whole volume segmentation, that would pave the way for more publicly available volume segmentations that would allow modern and complex machine learning processes to better study and understand sub-cellular processes.

Most of the work over the last years on biomedical imaging has been towards (semi-) automatic segmentation of concrete sub-problems, as discussed above. Based on that,

we categorize the segmentation processes as follows:

Object segmentation: here the aim is to segment single objects, often from user seed points. A human user inputs some seeds inside the object of interest (and optionally extra seeds to identify the background) and the object is extracted by using Region Growing (Pohle and Toennies, 2001; Chen and Shen, 2010), Watersheds (Sijbers et al., 1997), Active Contours (Chan and Vese, 2001; Li et al., 2005, 2010), Graph Cut (Boykov and Funka-Lea, 2006) or Random Walks (Grady et al., 2005; Top et al., 2011), for example. Object segmentation is more widely used in medical imaging, where the aim is to extract few organs and the user has to provide minimal input to extract each of them manually. It rapidly becomes expensive (both for computationally and time consuming for the user) when applied to biological imaging, both due to the hundreds of organelles present and because of the low signal to noise ratio, making these algorithms that rely on local structure fail to segment objects with missing boundaries. These methods are often quite computationally demanding, and applying them to a 3D volume with multiple objects of interest is often infeasible, as multiple interactions would be needed for each object and the computational cost of the algorithms wouldn't allow for real-time interactions. Additionally, some datasets with anisotropic sampling would lose boundaries on the Z direction, and without models having that into account approaches like region growing or watershed would fail to segment the objects.

Instance segmentation: where all the instances of a single object (or all objects of the same class) are aimed to be segmented individually. An example of applying superpixels to the instance segmentation process for 2D images of plants is available in Appendix C. This can be seen as a generalization of Object Segmentation, and the trivial example would be to segment all the objects of interest following an Object Segmentation approach iteratively, extracting every object on the image one by one by one of the methods described above. This however becomes infeasible and very expensive (both on user time and computationally) when the number of objects of interest is in the order of tens or hundreds. Other approaches include the use of template matching or object detection methods to first locate the objects of interest and then extract automatic segmentations from the found points. Though finding approaches that jointly detect and segment objects in images is not very common, there are some approaches working with light microscopy to locate and segment cells (Al-Kofahi et al., 2010; Wienert et al., 2012; Zhang et al., 2014) simultaneously. However, such approaches have been mainly applied to 2D images with homogeneous background and objects of interest with similar appearance, size and well defined boundaries. They use heuristics or assumptions about datasets

for detection of objects of interest (as there is nothing else in the images but object and background) and are able to obtain fully automatic results. Training based approaches such as the work of Kainz et al. (2015) use Random regression Forests to detect cells in images, which yield better precision on the detections and could be used to improve the segmentations of the aforementioned approaches. However, it relies on available training data in the form of distance maps to the centre of the cells, which as discussed above, is not always available. Recent approaches such as the jointly trained CNN (for both detection and segmentation) from Hariharan et al. (2014) could be used to further improve this methods, but its applicability to biological datasets is more complicated due to the lack of training data.

Semantic segmentation: probably the most complicated and least studied subject. It is very well known in computer vision with 2D images, where the whole image is parsed into objects with their corresponding classes. Semantic segmentation is sometimes also refereed as scene parsing, and has been the focus of the deep CNN work lately (Lin et al., 2016; Luc et al., 2016). For 3D volumes, however, these techniques are still far away since the main limitation is the lack of training data and the huge variability of datasets arising from biological imaging modalities. There are few approaches trying to merge interactive segmentation (or Object Segmentation) with machine learning in order to model the appearance of the objects of interest and extend it to a global optimization problem for the whole volume. ilastik (Sommer et al., 2011) was the pioneer in this field and the inspiration for most of the work presented in the following two chapters (chapter 6 and chapter 7), resulting in, among other publications, a tool named Super-Region Volume Segmentation (SuRVoS) (chapter 7) (Luengo et al., 2017b) to assist with interactive segmentation of challenging, anisotropic, biological volumes. Over the past two years Deep Convolutional Neural Networks (Deep CNN) approaches have started to be applied to this field. There are few works recently, with U-Net (Ronneberger et al., 2015) as the main pioneer and its 3D extension (Çiçek et al., 2016), which was also applied with few annotations. However, the U-Net with limited training data has only been applied to confocal images, which are multi-channel volumes with much lower resolution and much better boundaries. Although very promising, its applicability to higher resolution single-channel and much noisier datasets is still to come, due to the lack of sufficient training data to properly model the underlying complex structures.

Segmentation fusion: there are particular problems that arise when several of the above approaches are used together to solve a different global goals. For example, there is Class segmentation, where all the objects (or networks of objects) are

aimed to be segmented. This is, all voxels belonging to a class (e.g. membrane, mitochondria) are intended to be segmented simultaneously, without differentiating between individual instances. This is a special case usually used as an intermediate step for further processing, and thus we don't include as a particular segmentation strategy. A very common instance of this problem is the membrane detection in connectomics, an emerging branch of neuroscience that aims to map the physiology of the brain. The goal is to extract all the regions of on an image/volume, for which approaches such as (Vazquez-Reina et al., 2011) rely on an accurate detection of the membranes separating them. Deep Neural Networks (Ciresan et al., 2012) have been successfully applied to this particular problem as there is abundant training data for the membranes, and are able to achieve better accuracy than previous approaches. As the recent result from Beier et al. (2017) and Zeng et al. (2017) shows (both published couple of weeks before the submission of this thesis), the segmentation of connectomics is almost a solved problem, as deep CNN solutions already achieve near-human results. Thus, the field of connectomics can be seen as a particular case of Instance Segmentation, that uses Class Segmentation as intermediate step to find boundaries of objects and then segments all the instances of those objects by extracting different regions from those boundaries.

Before delving into the segmentation problem and how to interactively apply it to large volumes, the section 5.1 will give more insight on this particular dataset type to give a better understanding of the problem being addressed. Then, section 5.2 will give an in-depth introduction about Decision Forests (e.g. Random Forests (Breiman, 2001)) for classification of datasets. Decision Forest models are particularly well suited for classification of sub-cellular datasets since they don't require huge amount of annotated data (unlike Deep CNN (Ronneberger et al., 2015; Çiçek et al., 2016)) and they are inherently multi-class and multi-label (unlike SVM (Cortes and Vapnik, 1995), although there are tricks to make them work for multi-class problems, such as the One-versus-Rest (Weston et al., 1999; Weston and Watkins, 1998)), therefore, extensive use of Decision Forests and extensions to them will be introduced in the next two chapters (chapter 6 and chapter 7) and this section should serve as a broad introduction to them. Then, section 5.3 will highlight recent related decision forest models applied to solve different biomedical tasks, and finally related works in the literature that address similar problems or which are of particular interest to this thesis will be highlighted.

It is important to note that, to date, there are very few works that directly address the problem of sub-cellular semantic volume segmentation with no prior data available, and therefore related works from very different fields have been presented, with the aim

of gathering information and knowledge from different computer vision related fields to give insight into this very challenging one.

5.1 Understanding Cryo-Soft X-ray Tomography data

The scientific effort from several different areas of the scientific community needed to capture biological volumes is sometimes taken for granted. In sub-cellular biology, where the imaging modality is predominated by a variety of Electron Microscopes (EM), the aim is to study biological specimens at an unprecedented level. Standard Light Microscopes (LM) have a limited resolution of up to $0.2\mu m$, which is sufficient for most lab applications, but not enough to study structural and morphological changes inside cells. Recent advances in super-resolution try to over-come this limitation by achieving lower resolution by reducing illumination noise while reconstructing the data (Lukinavičius et al., 2013). Electron Microscopes, on the other hand, use electrons as a source of illumination and capture the scatter pattern or the direction of the electron beam, images of much smaller resolution can be captured. There are two main (common) types of Electron Microscopes, Transmission Electron Microscopy (TEM) or Scanning Electron Microscopy (SEM), each with their corresponding variations. TEM works similar to the standard Light Microscopy, but sending an electron beam instead of light though a very thin slice of the specimen, which achieves a resolution of $0.05nm$. The limitations of this technique are sample size, which has to be a maximum of few hundreds nanometers (depending on the model). SEM, as the name suggests works by scanning the sample's surface using an electron beam that interacts with the atoms, which produces various signals that are then analysed to understand the composition and topology of the samples. The most common SEM is the one that captures secondary electrons emitted when atoms gets excited. SEM achieves a natural resolution of $0.4nm$, but it is limited to study only the surface of the samples. Additionally, contrast is one of the main problems of these techniques, which has to be generated by embedding the sample into resin and adding heavy metals during sample preparation to scatter the electrons when the beam passes through.

Recent imaging techniques introduce Cryo-TEM, which replaces the resin with vitreous-ice to preserve the cells as close to the living state as possible. The depth of the samples is severely limited, to have a maximum depth of $750nm$ and the contrast in these generated images is very poor, as it is generated by a phase difference between biological material and surrounding ice. This causes multiple acquisitions from the same area to be required to later perform averaging and reduce the background noise.

Here is where Cryo-soft X-ray Tomography (cryo-SXT) has recently taken ground to fill the resolution gap between light and electron microscopes. Using soft X-rays in the so called water-window (284-500eV), which have enough energy to achieve up to $10\mu m$ penetration depth (an average cell is $3 - 15\mu m$ thick) and yet achieve a resolution of $40nm$ (which is currently limited by the zone-plate and is likely to improve in the future). Samples are vitrified prior to imaging, but no cutting/slicing is necessary, preserving the cell in its near native state. So, for the first time, whole cells can be imaged without damaging them prior to being imaged.

As with light microscopy, TEM or SEM, a single projection is not always sufficient to understand the structure or morphology changes inside a sample. Only 2-dimensional projections are not sufficient to understand the underlying configuration of the specimens, as with limited knowledge of its 3D state and the penetration of the X-rays and electrons, accurate measures are not possible. In many cases, going further is not necessary as the information given by a 2D projection gives enough information to understand what is in the interior of an specimen, but when the structure of its internal components is the true objective of the study, then understanding and studying the third dimension becomes essential.

The simplest way to studying the third spatial dimension is to setup a stereo system, and imaging the cell from two different angles to reconstruct 3D information. Stereo has been a very successful approach with natural images (images of real world recorded with a camera), however, light bounces back on objects to create an image, while electrons and X-rays penetrate them and are captured after the specimen has been traversed. This makes an standard stereo system unreliable, as the beam gives too much information to be reconstructed in a 3D view with just two images. Although, as previously mentioned, milling is an option and widely used in conjunction with Scanning Electron Microscopy (SEM) to scan whole specimens by scanning the surface of them layer by layer, the next option to study the 3D information of specimens, without damaging them, is tomography.

Tomography, as presented in the introduction (section 1.1, Figure 1.2), aims to scan different projections of the data at different angle intervals and reconstruct them in a full 3D volume. These projections themselves are too noisy, but when combined and averaged through reconstruction algorithms the true nature of the sample can be recovered. Tomographic reconstruction is a whole different and wide research field by itself. Hundreds to thousand of 2D projections (also known as *tilt series*) are captured per sample to later reconstruct a full 3D view of it. This 3D view results in a large 3D matrix which can range from about 500MB-1GB (standard size) up to several GB or even TB in size.

In cryo-SXT usual reconstructions are $1000 \times 1000 \times 500$ to $2000 \times 2000 \times 1000$, thus, 0.5-4GB. Reconstruction algorithms then present a huge computational problem, with their complexity bounded by memory and processor limitations. Lately, iterative algorithms (Palenstijn et al., 2011) have been made popular due to recent breakthroughs in GPU technologies, making iterative reconstructions have replaced traditional mathematical methods based on filtered back-projection (FDK/FBP), that although being mathematically correct, doesn't work well with noisy input from electron or X-ray projections. These iterative reconstruction algorithms takes few hours to reconstruct a 3D volume with modern GPU(s), which is sometimes longer than the acquisition time of complex sub-cellular volumes, but is still much less time than the time needed to analyse it.

Even in an ideal case, tomography reconstructions are not perfect for a variety of practicals reasons, but the most common in cryo-TEM and cryo-SXT tomography are the limited tilt angle of the microscopy. In the ideal case, a tilt series (series of 2D projections at different angle intervals) is collected with full view (360^o) around the sample to be imaged. As in the example in the introduction (Figure 1.2), assume that the sample being imaged is a cylinder. In order to get a full 3D reconstruction of the surface of the cylinder, only -90 to $+90$ rotation interval is needed to get an accurate 3D representation of the cylinder. However, if the cylinder was semi-transparent and we were interested in small blobs-like objects floating in the cylinder, a full 360 degree rotation is necessary to account for the opacity and occluding blobs. Any missing perspective (2D projection or tomogram) is information we lose that will introduce *artefacts* into the 3D reconstruction. An *artefact* are areas of the image with heavy deformation, the creation of false or erroneous data where it shouldn't be any (due to strong noise), the reduction of contrast due to extremal values and the a poor signal to noise ratio.

When imaging a cell with cryo-SXT, a population of cells are grown in a $3mm$ gold grid (sample grid) prior to being imaged. These grids are then vitrified and introduced into the microscopy to image them. First, the grid is scanned with a fluorescence microscopy to find in the grid an area with low density cells, in order to find an isolated cell to be imaged. This region of interest is then selected and imaged with cryo-soft X-rays, capturing a tilt series of the region of interest. This introduces a higher level of complexity to the reconstruction, as one could have noted, we are no longer trying to create a tilt series of a whole object by rotating around it, but we are trying to image a sub region of interest of such object, and we can no longer rotate a full 360^o as the region of interest wouldn't be visible. Current specialized microscopes, installed at B24 (beam line at Diamond Light Source), don't support full rotation either as the X-rays wouldn't have enough energy to penetrate the whole grid rotated more than 70 degrees, which limits the sample rotation from -70 to $+70$ degrees. With the addition of the sample being a sub-region that could be in an arbitrary position in the sample grid, the

centre of rotation is also affected, aggravating the artefacts generated during and after the reconstruction.

In short, in addition to poor contrast and low signal to noise ratio, which is expected in complex Tomography Reconstructions, the cryo-SXT system also introduces limited angle problem and an out of focus problem due to the centre of rotation changing on different tilts. In order to properly recover 3D information, fiducial (small gold particles that highly absorb X-rays) are introduced during sample preparation. These fiducial absorb all the X-rays resulting in small very very dark blobs in the 2D tomograms. Feducials are then used to perform tomogram alignment prior to imaging, which lessens the effect of the center of rotation changing every iteration. On the contrary, feducials make contrast of the image even worse, as they have a very high absorption coefficient compared to the organelles or water of a living cell. They create very dark blobs in the reconstructed image, with very bright halo-like artefacts surrounding the feducials due to the limited angle problem. Thus, while very much needed to properly perform image alignment, feducials highly alter the contrast of the image by introducing very dark blobs with very bright halo-like noise, and make the data of the nearby area somewhat unreliable. On addition to contrast and noise problems, limited angle is known to deform objects and blur (or even remove) image boundaries, specially in the slices far from the middle of the volume.

As a summary, segmenting 3D cryo-SXT volumes (or equivalently 3D cryo-EM volumes, which have even lower signal to noise ratio) are extremely challenging due to:

- Poor contrast due to feducials introduced to align the volume.
- Poor signal to noise ratio due to feducials, limited angle projections and reconstruction process.
- Several artefacts within the volume making it difficult to properly recognize or measure some areas of the volume.
- Elongated organelles due to the missing wedge (limited angle) reconstruction and anisotropic sampling.
- Highly anisotropic datasets, as data is likely to be more reliable in X-Y plane than X-Z or Y-Z (this is, data across Z is not that reliable due to noise and limited angel).
- Missing boundaries make it difficult to recognize where an organelle starts and ends (specially in depth or Z direction).

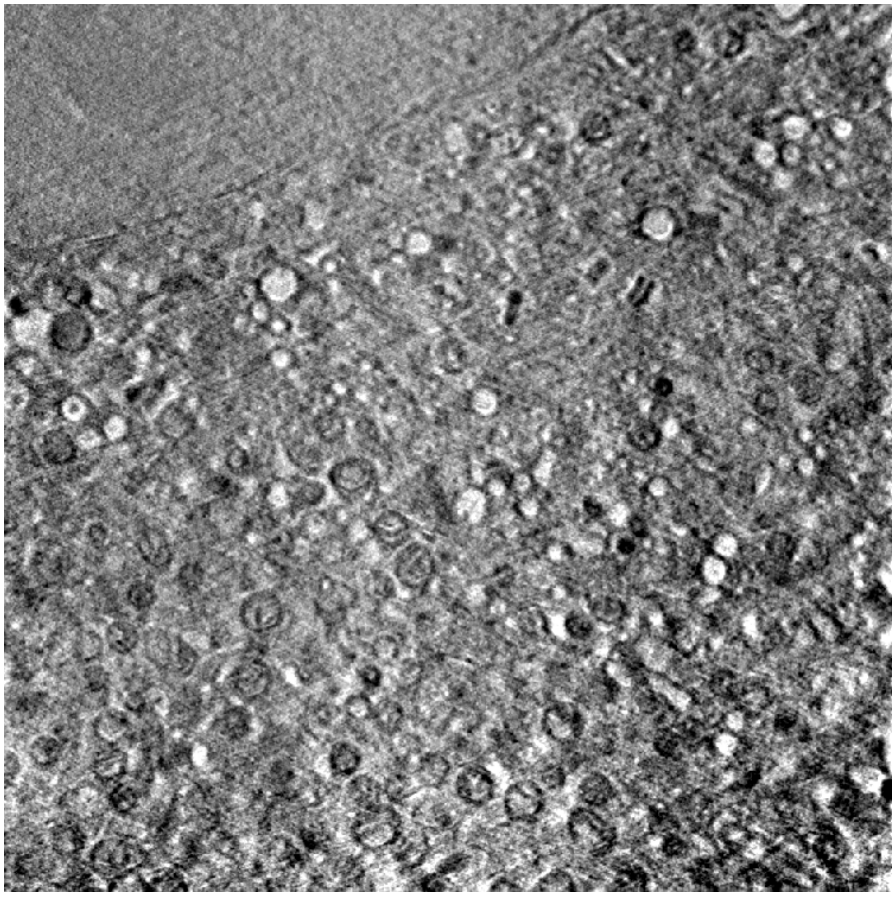


Figure 5.2: Example of cryo-SXT biological datasets, without any filter or pre-processing.

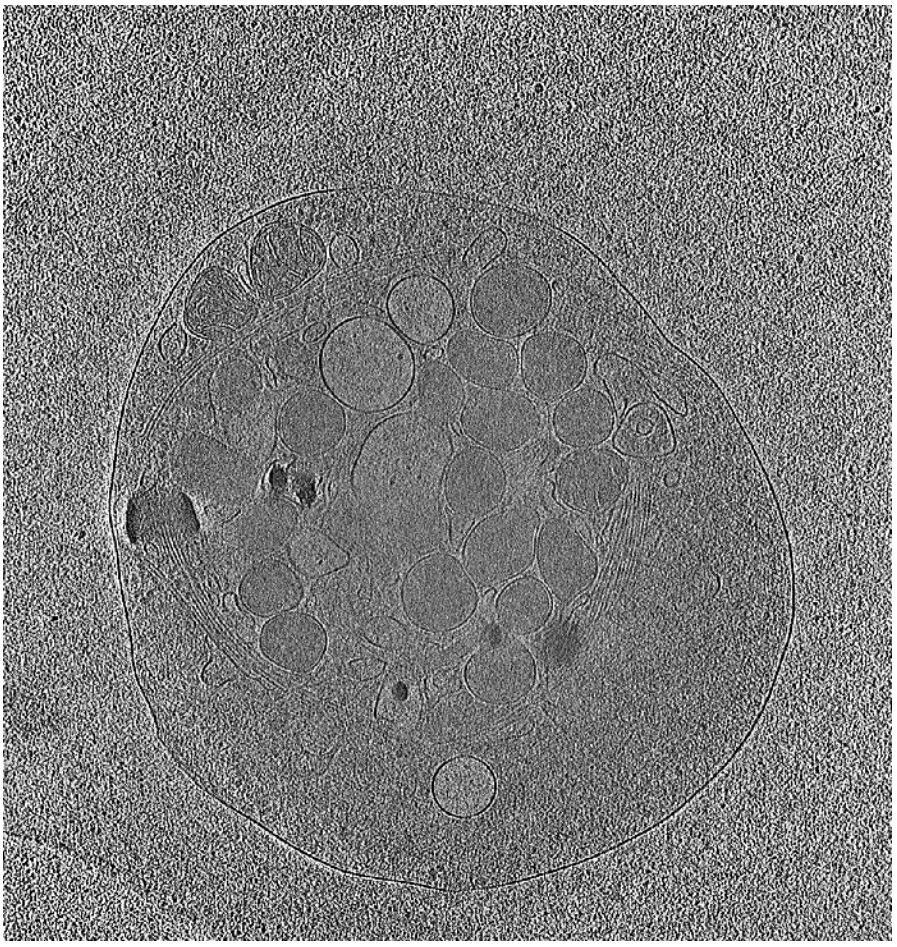


Figure 5.3: Example of cryo-EM biological datasets, without any filter or pre-processing.

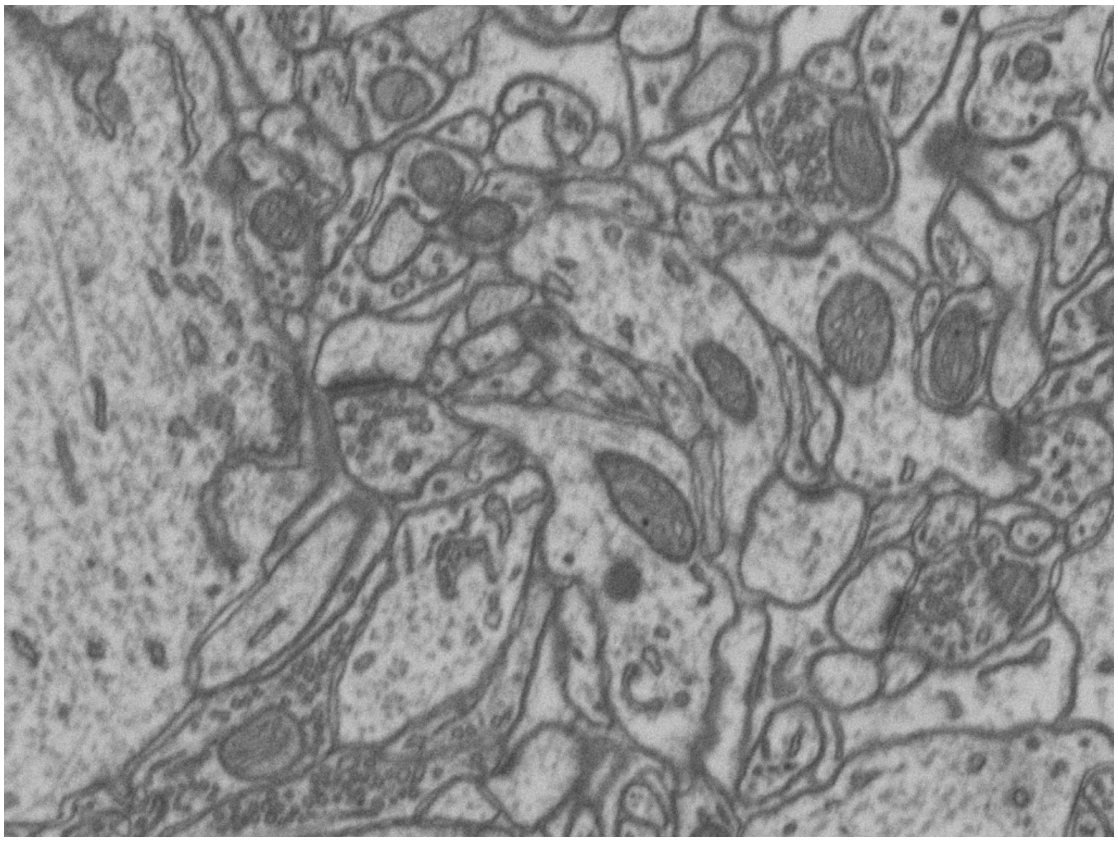


Figure 5.4: Example of resin embedded FIB-SEM biological datasets, without any filter or preprocessing.

5.2 Decision Forests for Classification

Making a temporary jump from cellular biology, this section aims to give a bit of an overview about Decision Forests (Criminisi et al., 2012). Decision Forests will be the key component to understand both related works and the novel work presented later in chapter 6. In order to keep this thesis self-contained a brief introduction to Random Forest and Extremely Randomized Trees (also known as Extremely Random Forests) will be given in this section. We follow notation and summarize the key concepts from the thorough work of Criminisi et al. (2012), and we refer the reader to it for further details.

Random Forests where introduced by Breiman (2001) as a result of merging Decision Trees (Breiman et al., 1984), largely studied in the literature but rarely applied to real problems due to their poor results, and ensembles, or in other words, how to combine a set of weak learners to build a strong model and approach complex problems. Random Forests are defined as a set of Decision Trees, each of them trained in a different subset of the whole data (process called bagging) which makes each tree learn different features and combines them to reduce their variance.

5.2.1 Decision Trees

A Decision Tree, denoted as $\mathcal{T}$, is defined as a hierarchical binary tree like structure, where nodes perform a binary test over the input samples $\mathcal{S}$ and separate them in two sets $\mathcal{S}^L$ and $\mathcal{S}^R$ that are passed to the left and right nodes respectively. The samples that satisfy the binary test are grouped into $\mathcal{S}^L$ while the nodes that don't pass it form $\mathcal{S}^R$. This process iterates until all the samples arrives at a leaf node for which a Decision Tree makes a prediction based on the distribution of all the samples that arrived at that same leaf in the past.

During training, a Decision Tree is initially composed of a single empty node (the *root* node) and receives a set of N_T data points $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_{N_T}\}$ composing the initial set $\mathcal{S}$. Each data point is a d-dimensional feature vector $\boldsymbol{x}_i = \{x_1, x_d\}$. The goal during training is then to find the optimal split according that maximizes the Information Gain I_j, which depends on the problem being solved (e.g. classification, regression, density estimation...). This is, being $j \in \mathcal{T}$ a node in a tree, $h(\mathbf{x}_i; \boldsymbol{\theta}_j) \rightarrow \{0, 1\}$ is defined as the split function of parameters θ_j applied to an input data sample $\boldsymbol{x}_i$, and the train procedure tries to find the optimal parameters:

$$\boldsymbol{\theta}_j^* = \underset{\boldsymbol{\theta}_j}{\operatorname{argmax}} \quad I_j \tag{5.1}$$

During testing, instead, the learnt split function is applied to the input to route it through the tree until it reaches a leaf node:

$$r = h(\boldsymbol{x}_i; \boldsymbol{\theta}_j^*) \tag{5.2}$$

where $r \in \{0, 1\}$ is a boolean flag that sends the sample to the left if it is 0 or to the right otherwise.

5.2.2 Offline Training of a Decision Tree

The split functions stored at each nodes of the trees play a crucial role in routing a sample through the tree until it reaches a leaf node containing its prediction. Initially these split functions where manually tuned based on a human expert's knowledge (manually crafted rules). However, the applicability of these systems was limited to toy examples as the trees were extremely biased from our poor ability to describe the real world based on rules (Breiman et al., 1984). Learning rules from data becomes essential in order to make the decision tree generalize better to newly unseen data. Learning the optimal split criterion for a single node is NP-Hard and with limited data is nearly impossible to find the optimal split within the unbounded feature space of the data. Thus, the split functions are learning in a greedy fashion, where each node j will learn the best split for the incoming data samples $\mathcal{S}_j$ into two disjoint sets $\mathcal{S}_j^L$ and $\mathcal{S}_j^R$ respectively. The problem as stated before is formulated as the maximization of the information gain I_j for parameters $\boldsymbol{\theta}_j$,

$$\boldsymbol{\theta}_j^* = \underset{\boldsymbol{\theta}_j}{\operatorname{argmax}} \quad I_j \tag{5.3}$$

with

$$\begin{aligned} I_j &= I(\mathcal{S}_j, \mathcal{S}_j^L, \mathcal{S}_j^R, \boldsymbol{\theta}_j) \\ \mathcal{S}_j^L &= \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{S}_j \,|\, h(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = 0\} \\ \mathcal{S}_j^R &= \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{S}_j \,|\, h(\boldsymbol{x}_i, \boldsymbol{\theta}_j) = 1\} \end{aligned} \tag{5.4}$$

Here $\boldsymbol{y}_i$ represents the ground truth of sample $\boldsymbol{x}_i$. The precise definition of the Information Gain I depends on the problem being solved. For instance for binary classification the best split would be the one that separates all the samples $\boldsymbol{y}_i = 0 \forall i$ into $\mathcal{S}_j^L$ and all samples with $\boldsymbol{y}_i = 1$ into $\mathcal{S}_j^R$.

5.2.3 Split functions

Split functions can be defined as any arbitrary function operating over the feature space of the samples, e.g. they could be made of hyperplane separation in linear SVM or a

decision boundary in a kernelized space. However, it is shown in (Criminisi et al., 2012) that the degrees of freedom of the split function has a huge impact on the performance on the tree (both training time and prediction accuracy) as node tests are optimized in a very reduced subset of samples $\mathcal{S}_j$ that arrive to node j and thus, a complex model would not only take much longer to train, but would also over fit the data.

In general, as the aim is to create an Ensemble of weak predictors, split functions are choose to be simple split threshold criterion over a reduced subset of the feature space ϕ, being the most used ones axis-aligned hyperplanes or oblique hyperplanes (Menze et al., 2011; Norouzi et al., 2015):

Axis-aligned hyperplane: a threshold applied to a single feature $\phi(\boldsymbol{x}) = x_k$.

$$h(\boldsymbol{x}_i; \boldsymbol{\theta}_j) = [\phi(\boldsymbol{x}_i) > \tau] \tag{5.5}$$

A special case of trees using this rule are popular in boosting after (Viola et al., 2003) and are decision trees formed of a single splitting node.

Linear separation: by creating an hyperplane over a reduced subset of features (e.g. one or two) $\phi(\boldsymbol{x}) = \{x_{r_1}, x_{r_2}\}$

$$h(\boldsymbol{x}_i; \boldsymbol{\theta}_j) = [\tau_1 > \phi(\boldsymbol{x}_i) \cdot \psi > \tau_2] \tag{5.6}$$

where ψ denotes a generic line. Note that by setting $\tau_1 = \infty$ or $\tau_2 = -\infty$ we recover a single threshold.

Non-linear separation: more complex splitting criterion can be created by generalizing the hyperplane to an arbitrary surface, e.g. with $\phi(\boldsymbol{x}) = \{x_{r_1}, x_{r_2}\}$

$$h(\boldsymbol{x}_i; \boldsymbol{\theta}_j) = [\tau_1 > \phi^T(\boldsymbol{x}_i)\, \psi\, \phi(\boldsymbol{x}_i) > \tau_2] \tag{5.7}$$

where ψ is this time a 2×2 matrix (or in general $k \times k$ where $k = |\phi(\boldsymbol{x})|$).

The parameters of the splitting function in the above cases are the thresholds τ_1 and tau_2, the mapping function ϕ (which features r_1 and r_2 to choose) and the projection vector/matrix ψ.

5.2.4 Introducing randomness in the training

The strength of the Random Forests comes from several non-correlated (or at worst, weakly correlated) classifiers. Randomness is introduced during the training process to

prevent all the weak classifiers from converging to the same one. Randomness is then introduced in different manners in the literature:

Tree Bagging: each of the trees is trained with a bootstrapped dataset. For a dataset with N samples $\mathcal{S}$, each tree is trained over a random sample of N samples (with replacement). It is to note that by doing this, samples will be repeated multiple times whereas other may not appear. This allows for each classifier to be trained in a different subset of the data and learn different properties.

Randomness injection at the node level: by reducing the parameter search for the split criterion from the whole feature space to reduced subset of randomly sampled feature candidates. By doing this, again, each of the nodes will be different and learn different things.

Several others exist, but in this thesis we will be mainly focusing in the second one. That allows to randomize the parameter selection to grow trees that are usually deeper (as the split functions are not as robust) but generalize better in practice and cope very well with over-fitting.

5.2.5 Training Decision Trees for classification

An ensemble (e.g. Random Forest) is composed of a series of T Decision Trees, each of them trained on a different randomly sampled dataset. Furthermore, the structure of the trees is unknown a priori, and thus has to be learned as well. Each of the trees starts with a single decision node, the root node. Nodes in a Random Forest try to optimize each of the Gini Criterion:

$$Gini(\mathcal{S}_j) = 1 - \sum_{m}^{M} p_m^2 \tag{5.8}$$

where M is the number of ground truth labels, and p_m is the probability of occurrence of class m within $\mathcal{S}_j$, tha is, the ℓ_1 normalized histogram of class distributions of nodes $\mathcal{S}_j$ that arrived at node j. Gini criterion measures the shape of the class distribution for a given node, for a leaf node with all the samples in it being of the same class, the summation will return 1 and thus, gini will be 0. As the class distribution gets more uniform, Gini will tend towards 1. The information criterion is then defined as the difference between the information on the current node and the leaf nodes for a given split parameters $\boldsymbol{\lambda}_j$

$$I_G = Gini(\mathcal{S}_j) - \frac{N^L}{N} Gini(\mathcal{S}_j^L) - \frac{N^R}{N} Gini(\mathcal{S}_j^R) \tag{5.9}$$

It can be seen from the equations above that for a binary classification problem, the split parameters $\boldsymbol{\theta}_j$ that maximize the Gini Information Gain I_G are the ones that put all the negative examples on one side (left or right) and the positives on the other one. This would make the class distributions on the children singular which would result on a *Gini* impurity of 0 for both left and right branches and thus maximize the information gain. Such split is however not always feasible to compute because the parameter search space for $\boldsymbol{\theta}_j$ would be too large, but it is also not desirable to find such split since the reduced amount of samples $\mathcal{S}_j$ at the current node might incur over-fitting.

Here we introduce two new concepts: when to grow the tree and split the current node into two new ones and how to find the optimal split. There are several stopping criterion for preventing a tree from growing indefinitively, the three most commons ones and the ones that we will be using along this thesis are:

Zero Impurity: when the Gini Impurity of the current node is smaller than machine epsilon, that is, close enough to zero meaning the class distribution of the current node is almost entirely focused on one label.

Minimum Number of Samples: a node has to contain a minimum number of samples in order to be a candidate to be split. This prevents from growing a branch that contains singleton nodes (nodes with single samples).

Maximum depth: in order to avoid over-fitting the maximum depth of a tree can be clamped

With methods to measure the quality of a split and how to grow a tree from a single root node, the only missing part is how to find the optimal split. It has been mentioned that searching for the whole parameter space is sometimes not feasible, but also not desirable, since it could over-fit the data, and it could also make all the trees to grow exactly in the same way.

For the simplest case where the split function is an axis-aligned hyperplane (the split criterion used in (Breiman, 2001) and (Geurts et al., 2006)) two new parameters k and ρ that control how to sample parameters $\boldsymbol{\theta}_j$ are introduced. The parameter k, with $1 \leq k \leq d$ controls how many features are going to be considered for splitting the current node (with $k = \sqrt{d}$ by default). If $k = d$ then all features will be candidates to be selected for the split criterion, resulting in all the nodes with the same input learning the exactly same optimal criterion, which is not desirable. With $k = 1$ on the other hand a single feature will be sampled at random, which is the opposite completely random case. On the other hand, the parameter ρ will control how many thresholds τ will be sampled for evaluation. To reduce the whole continuous feature space to a

more tractable space, a single feature p is first selected and then the discrete space of $\mathcal{S}_j$ (essentially, $x_{i,p} \forall \boldsymbol{x}_i \in \mathcal{S}_j$). Using the current data points as candidate axis-aligned hyperplanes will dramatically reduce the optimization problem of finding the optimal split, while still making it adequate. The parameter ρ then controls how many samples to use as candidate hyperplanes. If $\rho = N = |\mathcal{S}_j|$ then all samples will be used, on the other hand for $\rho = 1$ a single hyperplane will be drawn at random from the current sample pool as a candidate split for the current feature.

With this, each of the nodes that is a candidate to be split according to the termination criterion above will sample a total of $k \times \rho$ hyperplane candidates. By measuring the Information Gain of each of them the optimal split can be selected. It is to note that different from SVM or other standard classifiers, there is no need nor aim to find the global optimal split at every node, but to grow a set of different uncorrelated trees whose nodes will route samples differently until they arrive to a leaf node.

5.2.6 Class prediction with Decision Trees

Predicting the class of a sample is straight forward. The sample is routed through the root node to a leaf node by evaluating the learned split functions on it and deciding whether to send it to the left or right branch.

Once it arrives to a leave node (or a maximum depth requested) the target node can output either a class probability (as the maximum a posteriori of its class distribution) or a probabilistic class prediction as the ℓ_1 normalized class distribution of the node.

5.2.7 Ensemble models

In the previous sections the process of training a set of decision trees has been discussed. From learning the structure from a single starting root node, to how to sample and optimize the split parameters at each node.

A Random Forest(Breiman, 2001) for classification is then a set of T decision trees trained each on a bagged dataset. Each of the nodes of each of the trees will sample $k = \sqrt{d}$ feature candidates and will evaluate all the nodes $\rho = |\mathcal{S}_j|$ as hyperplane candidates. The trees will stop growing when they contain two or less samples or arrive to a maximum specified depth.

An Extremely Random Forest(Geurts et al., 2006) instead, will train each of the trees with the whole dataset (no bagging nor sampling needed) but will introduce a higher

level of randomness at the node level to achieve similar generalization. This is done by setting $\rho = 1$, which will sample a single random threshold for each feature candidate.

Extremely Random Forest are usually faster to train and usually grow much deeper (due to a less robust split optimization), but achieves in practice similar accuracy.

During testing, a sample is predicted by all the trees and then the forest outputs the maximum vote response for the class of the label. It is interesting to note that when the output of the classifier is not used as a definitive classification, the probabilistical interpretation of the whole forest for a given sample can be obtained as,

$$p(c \,|\, \boldsymbol{x}_i) = \frac{1}{Z} \sum_{t}^{T} p_t(c \,|\, \boldsymbol{x}_i) \tag{5.10}$$

where c is a given class, p_t the probabilistic prediction of the tree $t \in T$ and Z a normalization factor.

5.2.8 Other Ensemble Models

Other similar Ensemble models include Random Ferns (Ozuysal et al., 2010; Pauly et al., 2011), a constrained Random Forest where the split parameters $\boldsymbol{\theta}$ are the same for all the nodes in the same depth level. Random Ferns have been extensively used in the literature for object detection and localization. Decision trees, due to their efficiency on training and generalization properties have also been used as core weak learners for boosting techniques such as AdaBoost (Freund and Schapire, 1995; Zhu et al., 2009) or GradientBoosting(Friedman, 2001; Friedman et al., 2001), where decision trees are trained sequentially in a chain fashion and the miss-classified samples from one tree have a higher weight (or importance) in the next tree. That is, the output predictions of a tree are used as a weight for the samples of the next tree, creating trees that learn from the mistakes of the previous ones and improving the overall accuracy.

5.2.9 Why to use (Extremely) Random Forests

As mentioned before, (extremely) Random Forests will be extensively used later in chapter 6 and chapter 7. Other alternatives are available and have been widely used in the literature, such as the Support Vector Machines (SVM), which kernelized version often outperforms Random Forests, or Convolutional Neural Networks (CNN, such us U-Net (Çiçek et al., 2016)) that have been shown to outperform other methods when enough training data is available. However, in this thesis we will be focusing on a very particular problem in biological imaging: the interactive segmentation of high resolution

sub-cellular volumes with no previous training data available. This is, U-Net and some other previously mentioned methods have shown that with enough annotated training data, Deep CNN models are able to achieve excellent results, however, several areas of the volume need to be **fully** segmented prior to that. And this is where the rest of the thesis will be focused, introducing pipelines towards interactive segmentation of whole volumes or region of interests of them, with the aim of easing the annotation process so that other segmentation models can benefit and learn from the annotated data. To that end,

- Decision Forests will help collecting annotation data from minimal user input (in the form of scribbles) and yet be able to generalize very well to the same dataset without large amounts of annotated data (Criminisi et al., 2012).
- Both Decision Forests and SVMs require in general much less training data than CNN models, however, this is particularly true for Random Forests and linear-SVMs. As soon as more complex kernels (such as Radial Basis Function RBF) are chosen to fit the non-linear SVM, the amount of training data needed to infer the correct parameters grows rapidly.
- Random Forest classifiers and CNN segmentation models are inherently multi-label (and also multi-class), which make them very suitable to arbitrary segmentation problems, where the number of labels of different objects of interests is unknown a priory. Tricks such as One-versus-Rest (Weston and Watkins, 1998) allow SVM to perform multi-class classification, however, it requires to fit a separate SVM model for every label in the dataset.
- Random Forests are extremely efficient and can be trained (and used for testing) in parallel, as they are formed by a set of T decision trees independent to each other. Different from SVM, which requires sequential mathematical fitting, either by gradient descent or by solving a dual problem with better results but higher computational costs. On the other hand, CNN models such as U-Net require much larger amount of training data and the training times are reported to be at least 8-10 hours (Çiçek et al., 2016), which is infeasible for our problem as we want near real-time updates for interactive segmentation.
- Unlike SVM, Random Forests and CNN models inherently provide probabilistic interpretation of their classification results. Probabilistic outputs can also be extracted from SVM, such as the inverse distance to the classification hyper-plane when using linear-SVM, but its probabilistic interpretation is dependent on the testing set and requires additional computational resources to calculate them.

5.3 Decision Forests in Biomedical Imaging

There has been a trend over the last recent years to use Decision Forests to help segmenting biomedical volumes, in particular coming from medical imaging field. In most of the cases classification decision forests have been used to classify voxels or regions of the volume, but few exceptions arise, such as the work of (Criminisi et al., 2009, 2010), where regression forests where used to localize and extract the bounding box of several organs of CT scans. In their formulation, an ensemble of regression trees was trained to predict the displacement vector and the size of a bounding box from a given voxel.

Of particular interest for this thesis are (Lempitsky et al., 2009), (Geremia et al., 2010) and (Montillo et al., 2011), with the main difference that both were designed to deal with multiple medical datasets and learning from fully annotated inputs with very large runtime. The early work of (Lempitsky et al., 2009) used Random Forest to classify each of the voxel of a 3D volume into myocardium or non-myocardium. The interesting feature present in their work is the introduction of feature computation on-the-fly inside the Random Forest. Features for each voxel were computed by adding the intensities of all the voxels in a neighbourhood of radius $r = \{r_x, r_y, r_x\}$, where r was sampled randomly for each node. In (Geremia et al., 2010) they introduced Spatial Decision Forests, an extension of (Lempitsky et al., 2009) extends the input channels (gray scale or multichannel volumes) with features sampled from patches of random size at random displacements for every voxel, that is, the sum of intensities in the given patch computed with integral images. They showed that by adding these long range contextual-rich features the classification results were improved, as the Random Forest learned long-range contextual information. In (Montillo et al., 2011) the authors extended this idea by adding two different visual features, as the difference in intensity between two distant patches. In addition, they added entanglement features (referencing quantum mechanics) as they added contextual priors during training and testing, as for a given voxel at a given node in the tree, the confidence of itself and its neighbouring voxels is used to improve its classification. This has similar impact than auto-context (Tu and Bai, 2010), where the predictions of a classifier are used to train another classifier with the same training data, and thus, making it generate better predictions.

Other works of interest include Geodesic Forests (Kontschieder et al., 2013) which introduced a layer of class smoothing (similar to the application of an MRF/CRF over the voxel grid to refine labels) at fixed intervals of the Random Forest during training. That is, at every x levels of the Random Forest, they introduced a layer of class posterior smoothing (using geodesic distances between each of the nodes and each of the class

regions) that is then used as additional feature inputs for the next levels. This extension over entangled forests served as within-tree auto-context.

For practical purposes, however, the problem this thesis tries to solve is segmentation with limited annotation, and thus, no within-tree contextual features can be learnt of the form of (Montillo et al., 2011) or (Kontschieder et al., 2013), since whole volumes would have to be used during training to learn this context. However, short and long range features from (Lempitsky et al., 2009) and (Geremia et al., 2010) will later be extended in chapter 6, as they provide quick contextual information to improve the often noisy voxel segmentations.

5.4 Discussion

As mentioned above, some biological areas have already started to see the benefit from the Convolutional Neural Networks applied to biological problems. This is especially true for Medical Imaging or connectomics as they study datasets (usually with isotropic voxel sampling) for which plenty of already segmented datasets already exist, and which contain well defined boundaries visible in the main three perspectives.

On the other hand, most of other sub-cellular biological fields lack the homogeneity on datasets and available training data, which is where the rest of this thesis will be focus. In these fields, biologists collect large dataset of sub-cellular resolution much faster that they can be analysed. These dataset are usually volumes up to $2000 \times 2000 \times 2000$, with varying resolution depending on the imaging modality utilized to collect the data. Different cell samples with different morphology are imaged frequently and the lack of previous annotations, the low signal to noise ratio due to the imaging resolution, and limited research focused in such a difficult field, make automated techniques impossible to succeed.

The aim in this sub-cellular resolution field is to study the behaviour and morphology of sub-cellular samples under different circumstances, e.g. the presence or absence of a pathogen or bacteria. In order to study structures at this microscopic levels, both control and infected samples need to be imaged under special microscopes. At the same time, this microscopes often require special facilities to run. Because of the complexity of the studies and the time needed to collect and analyse these samples, these studies are often based on a few samples (< 10).

Thus, with all the challenges mentioned above, it is a fact that general computer vision and machine learning and segmentation techniques would fail on these datasets due to

the lack or very scarce of amount of previously annotated data available for training. The reasons for previous data not being available are summarized as follows:

- New projects are built onto specific new samples obtained with a cutting-edge imaging technique, for a very specific purpose.
- Each of the imaging techniques generates completely different image representations of the same data.
- Even the same imaging technique would generate different images for the same sample in different microscopies, due to the extremely complicated setup and calibration of them.
- Projects usually contain very sensitive data that for privacy reasons is not shared.
- Segmentations done for these projects are usually manual (very time consuming) and minimal in order to extract solely the object(s) of interest, resulting on very valuable segmentations that only target minimal areas and are often not interesting for general purpose algorithms.

During the next two chapters, algorithms and implementations will be tested and focused in Cryo-Soft X-ray Tomography (Cryo-SXT) datasets collected at B24 beam line at Diamond Light Source (B24), along with some other publicly available FIB-SEM datasets online (Lucchi et al., 2012a). Cryo-SXT datasets serve as a very challenging samples, with good contrast but low signal to noise ratio. Due to the limitations during the data collection, only a limited number of projections can be used to reconstruct a 3D volume of the data. This is known as a limited angle problem, and introduces several artefacts in the reconstructed volume such as elongated organelles, missing boundaries and unclosed contours. With no previously available annotations for this kind of datasets, as the imaging technique is rather new, this imaging modality poses as a very challenging target. As the aim of this thesis is to introduce novel and generalizable interactive segmentation workflows towards segmentation of arbitrary volumes, using one of the most challenging imaging modalities as target data will give insights and understanding of why this is such a difficult problem. It is to note, however, that all the methods discussed on the following chapters use no prior knowledge about the dataset and are generalizable to arbitrary volumes.

As a final remark and for clarification, the rest of the thesis will be focused on the semantic segmentation of sub-cellular datasets with no previous training data available, an extremely difficult problem due to various reasons already discussed in this chapter.

6 Super-Regions for Interactive Segmentation of Biological Volumes

We have already introduced all the challenges arising from cryo-SXT (or cryo-TEM) datasets, these dataset contains artefacts, low signal-to-noise ratio and hundreds of organelles to be segmented. To date, most of published biological studies in these kind of datasets use manual or semi-manual segmentation. Manual segmentation involves purely an expert user annotating the data for long and tedious hours. Semi-automatic annotation usually include boundary tracking algorithms such as Active Contours (Snakes, Level sets and variants) or Live-wire (in 2D or 3D). Such models are provide very accurate segmentation of the objects of interest, either by volume registration or contour optimization. However, biological datasets (i.e. TEM, SEM or cryo-TEM datasets) remain very challenging as each dataset corresponds to a different sample, which has its own subtleties in terms of image properties. This sample is usually at the cell-scale with low signal to noise ratio, with potentially tens to hundreds of sub-cellular features such as organelles which need labelling. Here, as the internal organization of a cell is not likely to match any other one, template models cannot be learnt. Boundary tracking models could still provide a sufficient segmentation of different organelles, but it would involve too much user interaction as an initial contour would be needed for each of many organelles. Recent works on contour evolution for interactive segmentation improve the need for near-contour initialization, but these methods still have slow convergence (even when running on the GPU) and therefore are not feasible for datasets with hundreds of organelles.

To handle the lack of structure in the target datasets, machine learning approaches have been adopted in recent years. Here, using available training data, local voxel descriptors

are extracted and a classifier is trained to learn to classify a voxel into different classes. This approach has previously been used to successfully segment mitochondria Lucchi et al. (2012a) or cell boundaries and organelles Vazquez-Reina et al. (2011). In fields such as connectomics, that analyses human brain cells, and other medical imaging fields where samples are likely to match previously seen ones, these methods are starting to have a huge impact. With previously segmented volumes and large challenges and public datasets being released, algorithms that learn from training data and generalize well to test data are currently dominating the late state of the art for segmenting membranes or other particular organelles. These approaches are yet to have an impact in biological cell imaging due to the lack of training data and very different cell samples. This section will be focused on building a framework in that direction.

To deal with the lack of training data, we will build a framework as an interactive segmentation tool. Recently ilastik (Sommer et al., 2011) was introduced as a ground breaking framework for interactive segmentation of volumetric datasets in the field of biological imaging. It introduces machine learning as a tool to learn to segment volumes from user annotations. First, user defined features are extracted from the data (more information about features can be found in *ilastik* web-page [1], but they will also be covered later in subsection 6.1.2). A user annotates different areas of interest (organelles, background, cell areas, etc) with scribbles (associated with one color for each label). An online Random Forest is then learnt from these annotations and the label for each pixel in the dataset is predicted by the classifier. The main drawback of this framework is the expensive computational cost of training a predicting labels for about 500M pixels in an standard biological 3D dataset. Learning from a few thousand pixels can take a few minutes, but predicting a label for all the pixels can take a few hours. ilastik solves the problem by implementing a lazy flow that only computes features, trains and predicts on the slices that the user is currently visualizing. However, visualizing the result across the volume decreases its effectiveness.

Here we build on the ideas of ilastik and the supervoxel segmentation of (Lucchi et al., 2012a) and propose an interactive segmentation pipeline that combines the best characteristics of both worlds. The interactive classification and features of the former will be extended to the supervoxel representation of the later. We will explore different data preprocessing and features in order to extract interesting information from the volume that will later be used in a classifier. First, local voxel features will be combined into more robust and discriminative supervoxel descriptors. Different from (Lucchi et al., 2012a), our descriptors will be formed from voxel features and will yield more discriminative representations for the supervoxels, while being much more computationally efficient. These features will then be used to train an online classifier (Online Random Forest to

[1] http://ilastik.org/

be precise) in order to learn from user annotations and predict the rest of the volume. Additionally, we will be targeting the full volume segmentation problem, this is, the aim of our work here is to fully segment all the objects, organelles, cell areas and other structures present in a 3D volumetric dataset. This makes our segmentation problem even more challenging, as we are no longer focusing in a single feature (e.g. mitochondria or cell membranes). Because single user annotations don't contain sufficient information to discriminate between all the possible classes present in a volume, we will introduce Hierarchical Segmentation Layers, where the problem of fully segmenting a volume will be split in smaller semantic segmentation sub-problems, where the information will flow from the top layers to add contextual information and help segmenting the more challenging bottom layers. Figure 6.1 shows an example of our hierarchical segmentation layers used to split the segmentation process in easier problems.

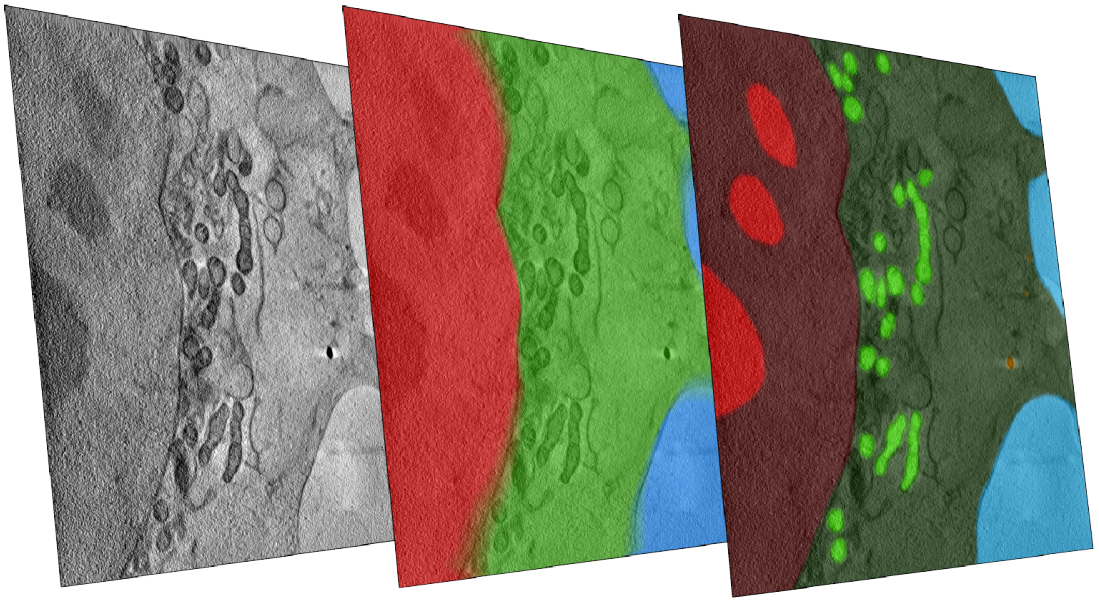


Figure 6.1: Hierarchical Segmentation Layers: instead of segmenting the whole volume at once, which could be difficult due to textural similarities of different regions of the dataset, segmenting the data hierarchically, by segmenting the data in meaningful regions and then inside those regions, makes the segmentation task much easier and intuitive.

The use of supervoxels as representative units decreases the problem complexity several orders of magnitude. Instead of applying our segmentation framework on the standard volume grid (of millions of voxels), we will show that extracting a Region Adjacency Graph (RAG) of supervoxels will make MRF models converge very efficiently in a few seconds, compared to the few hours that would take for a MRF on the voxel grid. Figure 6.2 shows how a RAG, a graph of adjacent supervoxels connected together represents the data with much less representative units.

The rest of this chapter will be organized as follows:

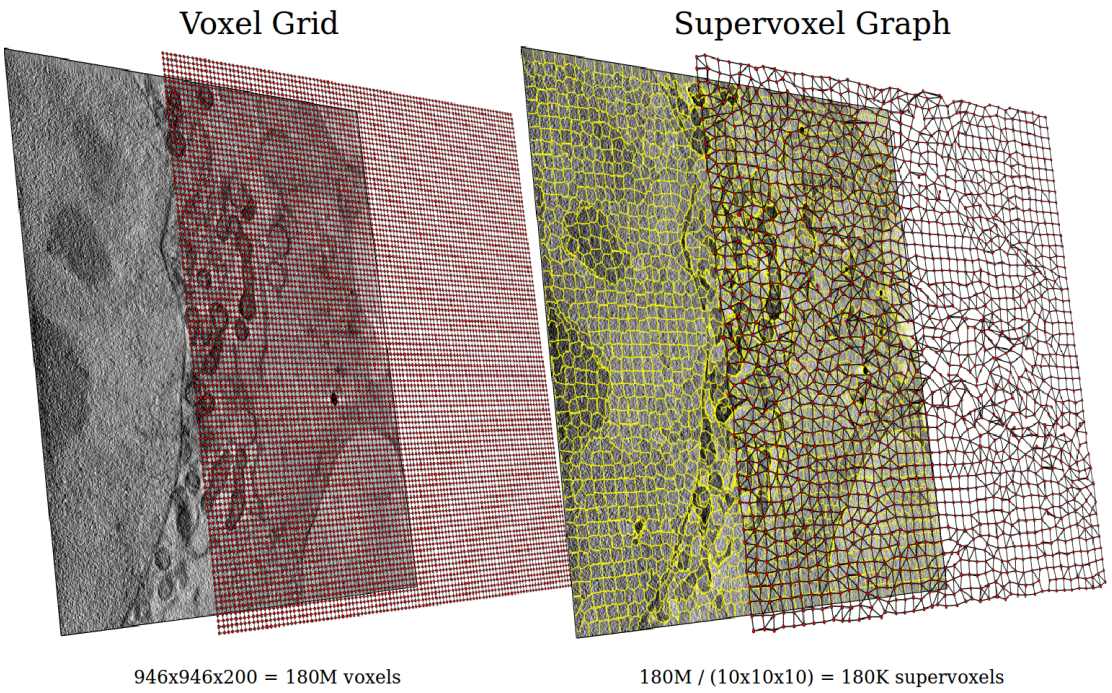


Figure 6.2: Left: A voxel grid is shown over a cryo-SXT image. Right: supervoxel over-segmentation set to approximately $10 \times 10 \times 10$ *supervoxels and their corresponding Region Adjacency Graph. For visualization purposes only the central slice is shown. It can be seen that the data can be represented with much less representative units by using supervoxels, which would speed-up subsequent processing operations.*

- Hierarchical Segmentation Layers will be first introduced (section 6.1) in an interactive segmentation framework. This framework is based on the grounds of ilastik (Sommer et al., 2011) and the work on supervoxels for volume segmentation of (Lucchi et al., 2012b). Our algorithm will take de advantages from both works and combine them in a hierarchical fashion to further improve the segmentation accuracy and reduce the amount of manual annotation needed from the user.

- Selective labelling will then be presented (section 6.2). Interactive segmentation is not a trivial task and often users don't know which part of the volume to annotate. To both guide the user through the annotation task and minimize the input needed from the user, a subset of representative regions will be extracted from the volume and then combined with an active learning procedure to allow the user to segment large volumes by giving annotations in small windows.

- Last, a more robust segmentation model will be shown (section 6.3). This model can learn more complex properties from the datasets by following the sequential window labelling procedure of Selective Labelling. Having a set of regions already segmented allows for a more robust modelling as context can be extracted from the surrounding regions of voxels or supervoxels.

6.1 Interactive segmentation with Hierarchical Segmentation Layers

Adding contextual information to the training and testing process has been gaining a lot of focus in computer vision. The general idea of incorporating contextual information to the estimation of the class-probabilities of a pixel seems quite appealing since standard classification techniques don't exploit the underlying 3D neighbouring structure of voxels. In 2009 Tu and Bai (2010) proposed an auto-context algorithm to train hierarchically probabilistic boosting trees (PBT) (Tu, 2005) using the output of one classifier as an input for the next one with interesting results for Brain MRI volume segmentations resulting in increasingly more accurate probability estimates in further layers of the contextual tree. Later in 2011 Montillo et al. (2011) proposed Entangled Decision Forests (EDT) to segment CT images. EDT forest also add contextual information to the training stage by adding class-probabilities of neighbouring voxels in a previous node of the decision trees as additional features for the voxels.

The works above, however, learn the contextual information during training step and for a single set of labels, improving the probability estimates of the same problem over the contextual layers. Here we propose a similar approach to add contextual information to our interactive segmentation by adding a n-step segmentation. To illustrate our approach an example will be shown with 2 steps (contextual layers), but it is to note that our approach is directly generalizable to n arbitrary layers. The first step requires quick user scribbles to infer probabilistic volume areas that are used as contextual information for the second step supervoxel classification. The second step will use the predictions of the cell areas to add contextual information to the predictions for other, more challenging sub-cellular organelles or small areas. Because of this 2-step segmentation, initially undistinguishable structures (such as mitochondria and nucleoli in our example bellow) will now be more easily separable due to the contextual prior of the first layer indicating that both structures belong to different cell areas.

The following sections will highlight the main contributions of our hierarchical segmentation pipeline, including:

1. **GPU Implementation of SLIC Supervoxels:** a 3D variant of SLIC superpixelsAchanta et al. (2012), inspired by g-SLIC (Ren and Reid, 2011), which allow us to extract anisotropic supervoxels from a $1000 \times 1000 \times 200$ volume in around 5-10 seconds and add prior information such as sample size in each axis. The 3D extension of SMURFS (subsection 4.7.4), that is capable to extract more robust supervoxels at the expenses of larger computational time, could also be used to

obtain better segmentation results, however, for an interactive segmentation our modification of gSLIC will be sufficient as it can be used over different features with near real-time updates.

2. **3D textural features based on Sigma Set:** simple yet robust and efficient descriptors using Sigma Set (Hong et al., 2009) will be extracted, which are covariance-like descriptors that reside in a Euclidean space (). These features are very efficient to compute and are also parallelized in the GPU.

3. **Use of contextual information to refine classification:** the user is asked to draw hierarchically more informative scribbles to allow us to partition the image and obtain better estimates with an Online Random Forest, as training a Online Random Forest using the contextual information of previous levels of the hierarchy reduces the bias towards common classes and the overall variance by spatially separating the classes. This effect is similar to the widely studied auto-context (Tu and Bai, 2010) adapted to an interactive segmentation problem. We also include contextual information as a hierarchical label cost in the MRF refinement.

6.1.1 Supervoxel over-segmentation

Superpixels consist of a group of neighbouring pixels in a given image that share some properties, such as texture or color. Each of the pixel of the image belong to exactly one superpixel, and by adopting the superpixel representation of an image, the complexity of a problem can be reduced two or three orders of magnitude. A sample image with $500 \times 500 = 250000$ pixels can be encoded with just ≈ 800 superpixels, which has a dramatic effect on the speed of any subsequent algorithms, which can act at the superpixel, rather than pixel, level. Superpixels, in order to allow an efficient representation of the image, must share a certain characteristic: every strong boundary of the image should lay along superpixel boundaries, or in other words, all the pixels of a superpixel must lay in a single higher lever object of the image.

3D G-SLIC Supervoxels with variable supervoxel shape

Simple Linear Iterative Clustering (SLIC) (Achanta et al., 2012) is one of the most common superpixel over-segmentation algorithm that splits a given image into superpixels by performing a k-means-based local clustering of the pixels in the 5-D $\{labxy\}$ space. The algorithm take as an input the approximate desired number of superpixels K and a compactness factor m. Then, for an image of N pixels, the approximate size of each superpixel becomes N/K and the image is split into a grid of uniformly distributed

superpixels of size $S \times S$ with $S = \sqrt{N/K}$. Then, an iterative local k-means is applied by searching for every super pixel in the grid the pixels that belong to it in a $2S \times 2S$ window according to a custom distance function that controls the compactness of the superpixels D_{sp}. Each pixel is assigned to the cluster with smallest distance. For a given pixel $i \in N$ and a given superpixel centre $k \in K$, the distance from the pixel i to the superpixel k would be defined as $D_{sp}(i,k)$:

$$\begin{aligned} d_{lab}(i,k) &= \sqrt{(l_i - l_k)^2 + (a_i - a_k)^2 + (b_i - b_k)^2}, \\ d_{xy}(i,k) &= \sqrt{(x_i - x_l)^2 + (y_i - y_k)^2}, \\ D_{sp}(i,k) &= d_{lab} + \frac{m}{S} d_{xy}, \end{aligned} \tag{6.1}$$

where m is the compactness factor, $\{l, a, b\}$ is the L*a*b color space and $\{x, y\}$ corresponds to the x and y position of the pixel in the image and the cluster centre. As the k-means algorithm, a 2-step expectation and maximization algorithm optimizes the $\{l, a, b, x, y\}$ feature of the cluster centres and each pixel is finally assigned to its nearest cluster after $T = 5$ iterations.

Later, g-SLIC(Ren and Reid, 2011) introduced a GPU version of SLIC that replaces the search in $2S \times 2S$ windows around each superpixel centre by a lookup table that contains the $M = 9$ nearest cluster centres for each pixel. Thus, instead of performing K sequential searches for every superpixel, M searches are made for every pixel in parallel. The computational cost would be much higher in a sequential CPU, however, by running it in a GPU with hundreds of cores the performance is greatly improved.

To apply SLIC to a volumetric X-ray dataset we sightly change the formulation of (Achanta et al., 2012) and (Ren and Reid, 2011). First, as superpixels are going to be extracted in 3D, from voxels rather than pixels, they are termed supervoxels. To handle the anisotropicity of some X-Ray datasets, we replace the desired number of supervoxels K with the desired approximate shape of the supervoxels $\mathbf{S} = \{S_x, S_y, S_z\}$ in each dimension. This allows us to make for example less deep superpixels (i.e. $\mathbf{S} = \{S_x, S_y, S_z\} = 10 \times 10 \times 3$ supervoxels) if X and Y are the isotropic axes and Z suffers anisotropicity due external factors like limited data or limited available angle in CT-reconstruction. The total number of supervoxels is extracted by the relation between their expected size and the total number of voxels N in the image $K = N/(S_x \times S_y \times S_z)$. By initially sampling K supervoxels of shape $\mathbf{S}$ we already encourage superpixels of configurable shapes in the image. However, to enforce that constrain over the optimization scheme, we introduce spacing factors $\mathbf{d} = \{d_x, d_y, d_z\}$, standard in biological image acquisition, to weight spatial distance from a voxel to a superpixel centroid along the different axes. Finally, the 5-D $\{l, a, b, x, y\}$ feature vector is replaced with a 4-D $\{I, x, y, z\}$ vector where I is the intensity of a voxel in a X-ray volume. Equation 6.1

can then be re-written as

$$\begin{aligned} d_I(i,k) &= \sqrt{(I_i - I_k)^2}, \\ d_{xyz}(i,k) &= \sqrt{\left(d_x(x_i - x_l)\right)^2 + \left(d_y(y_i - y_k)\right)^2 + \left(d_z(z_y - z_k)\right)^2}, \\ D_{sp}(i,k) &= d_I + \frac{m}{|S|} d_{xyz}. \end{aligned} \tag{6.2}$$

The spacing factors can be set to existing sampling units (in mm, μm or nm) if known from the volume or empirically defined by the user. Note that by setting $\mathbf{d} = \mathbf{1}$ the same spatial distance function d_{xy} is obtained (its 3D extension).

Last, in order to parallelize the algorithm in the GPU, we use a similar approach to g-SLIC, and parallelize over the voxels by finding at each Expectation-step (of the local k-means) the nearest 26 supervoxel centres (26-connected in 3D) for each voxel and assign to each one its nearest one according to Equation 6.2.

6.1.2 Robust Feature Extraction

Once supervoxels are constructed from the volume, we represent each supervoxel by means of simple first and second order statistics. Typical approaches include using the mean intensity of the supervoxel, or the histogram of intensities within each supervoxel. More advanced features include SIFT (Lowe, 1999) or Histogram of Oriented Gradients (HOG) (Dalal and Triggs, 2005), which have been proven to be extremely robust for object detection, but lose performance when the objects of interest have a lot of variability in shape and size and training data is limited.

Here, we use a simple yet robust and efficient Sigma Set descriptor (Hong et al., 2009), based on regional covariances that has been used before for object tracking (Hong et al., 2010) and saliency estimation and object detection (Kocak et al., 2014). With $\theta(\mathcal{V}, x, y, z)$ a mapping function that extracts d-dimensional feature $\mathbf{f_i}$ for each voxel i in a volume $\mathcal{V}$ (such as position, intensity, derivatives or other local textural cues). Then, a region $\mathcal{R}$ inside the feature space θ can be described by means of a $d \times d$ covariance matrix

$$\mathbf{C}_{\mathcal{R}} = \mathbf{F}_{\mathcal{R}} \mathbf{F}_{\mathcal{R}}^T, \tag{6.3}$$

where $\mathbf{F}_{\mathcal{R}} = [\hat{\mathbf{f}}_1, \hat{\mathbf{f}}_2 \ldots, \hat{\mathbf{f}}_n]$ denotes the feature matrix of the centred d-dimensional feature vectors $\hat{\mathbf{f}}_i = 1/\sqrt{N}(\mathbf{f}_i - \boldsymbol{\mu})$ of all the voxels $i \in \mathcal{R}$. As noted in Hong et al. (2009) a non-singular covariance matrix (or symmetric positive definite matrices) lie in a non-linear Riemannian manifold and not in an Euclidean space. Thus, in order to perform classification or evaluate differences between descriptors, these need to be projected onto a linear space. To achieve this, the Sigma Set creates a set of points S that satisfies

$\mathbf{C}_S \simeq \mathbf{C}_\mathcal{R}$ in terms of first and second order statistics and define efficient Euclidean distance transformations.

For this purpose, the Cholesky decomposition of the covariance $\mathbf{C} = \mathbf{C}_\mathcal{R}$ is extracted $\mathbf{C} = \mathbf{L}\mathbf{L}^T$, where $\mathbf{L}$ is a lower triangular matrix, and multiplied by a scalar that as Hong et al. (2009) suggests is set to $\sqrt{d}$, yielding $\hat{\mathbf{L}} = \sqrt{d} \times \mathbf{L}$. Thus, our first regional descriptor for a supervoxel k is constructed as:

$$\boldsymbol{\phi}_k = \{\hat{\mathbf{L}}_1, \ldots, \hat{\mathbf{L}}_d, (-\hat{\mathbf{L}}_1), \ldots, (-\hat{\mathbf{L}}_d)\}, \tag{6.4}$$

where $\hat{\mathbf{L}}_i$ is the i-th column of $\hat{\mathbf{L}}$. This descriptor has the same second order statistics as $\mathbf{C}_\mathcal{R}$, and by adding the mean vectors, the same 1st orders statistics can also be recovered as

$$\boldsymbol{\phi}_k^+ = \{(\hat{\mathbf{L}}_1 + \boldsymbol{\mu}), \ldots, (\hat{\mathbf{L}}_d + \boldsymbol{\mu}), (-\hat{\mathbf{L}}_1 + \boldsymbol{\mu}), \ldots, (-\hat{\mathbf{L}}_d + \boldsymbol{\mu})\}, \tag{6.5}$$

which yields the final sigma set descriptor. We will later see (in section 6.1.3.2), that adding the mean $\boldsymbol{\mu}$ vector to the descriptor is helpful in most situations, but has to be used with caution as samples will be biased towards the mean descriptor.

In this work, we use a *7*-dimensional texture descriptor and a *10*-dimensional feature vector to describe the position and texture of a voxel $i \in \mathcal{V}$

$$\mathbf{f}_1 = \{I, I_x, I_y, I_z, I_{xx}, I_{yy}, I_{zz}\}, \tag{6.6}$$

$$\mathbf{f}_2 = \{x, y, z, I, I_x, I_y, I_z, I_{xx}, I_{yy}, I_{zz}\}, \tag{6.7}$$

where x, y and z indicate the absolute position (in range $[0, 1]$), I indicates the intensity of the voxel and subscripts indicate directional derivatives. The final Sigma Set descriptors are then constructed from the covariance matrix of all the voxels belonging to each of the supervoxels.

We will see later that despite of the simple form of the textural descriptors, they achieve sufficient discrimination power when combined into Sigma-Set descriptors. We will later show in the qualitative evaluation that using better features, such as the ones that will be described later in section 7.3 or the ones used by (Sommer et al., 2011) the quality of the segmentation can improve dramatically at the expenses of computational time and power. The feature engineering for volume segmentation is volume specific and application specific (as different features enhance different properties of the data) and thus, the feature engineering topic is out of the scope of this work.

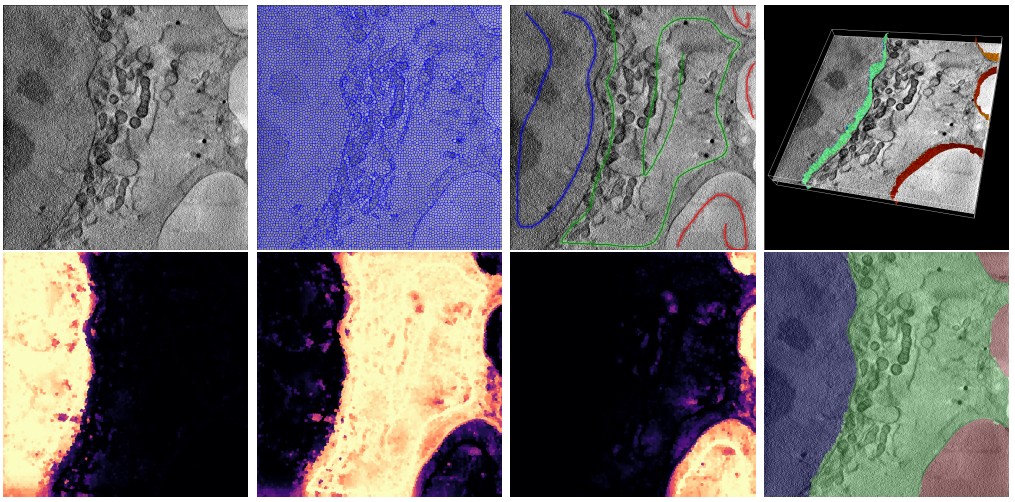


*Figure 6.3: Overview of the algorithm. **[Top row]** user experience, from left to right: central 2D slice of target volume, supervoxel segmentation, user drawn scribbles (blue: nucleus, green: cytoplasm/organelles, red: ice), final 3D segmentation showing boundaries between segmented areas. **[Bottom row]** intermediate class likelihoods (nucleus, cytoplasm/organelles, ice) given by the ERF and final segmentation with the MRF model. (Note for visualization purposes, only central 2D slices are shown, but processes happen in 3D data)*

6.1.3 Interactive Classification with Contextual Segmentation Layers

Our hierarchical segmentation approach is based on Extremely Random Forests (Geurts et al., 2006), an ensemble of T Decision Trees. In Extremely Random Forests, the input is formed by a training samples $\mathbf{X}_{N\times K} = \{\boldsymbol{x}_1, \boldsymbol{x}_i \ldots, \boldsymbol{x}_N\}$ and ground truth labels $\boldsymbol{y} = \{y_{1,i}, \ldots, y_n\}$, where $\boldsymbol{x}_i = \boldsymbol{\phi}_i^+$, N is the number of samples and K the number of features per sample. During the training step, each tree receives the full set $\mathbf{X}$ and, since the optimal tree training is NP-hard, trees are grown in a greedy manner. The root node selects $k = \sqrt{K}$ random split tests $\mathcal{S}$. A split test contains a randomly selected feature index j (from $\boldsymbol{\phi}$) and randomly sampled threshold t, which can be parametrised as $s = \{j, t\}$. Out of the k split criterion sampled, the best one is selected according to the information gain of the Gini criterionBreiman (2001), yielding the optimal split $s^* = \{j^*, t^*\} = max_{s\in\mathcal{S}} \quad G(\mathbf{X}, s)$ (where G stands for information gain). The node splits into a left and right node, and all the samples are split in two sets $\boldsymbol{X}^l = \{X_i \,|\, \forall i \in N\,,\, X_{i,j^*} \leq t^*\}$ and $\boldsymbol{X}^r = \{X_i \,|\, \forall i \in N\,,\, X_{i,j^*} > t^*\}$, which are passed to the left and the right node respectively. The process is repeated in the binary tree structure until a stopping criterion is reached, such as the information gain for the optimal split being lower than a threshold at a given node or a maximum depth is achieved. During testing, each test sample is fed to each of the T trees and routed through the root node to a leaf node (according to the optimal split tests learn during the training phase). The leaf node then, contains a class probability distribution of

the nodes that during training reached that same node. The probability distributions returned by each of the trees are then added and normalized to give the final class-probability estimate for each of the samples.

Each trained tree is unique, as each of them selects the split criterion (both features and thresholds) randomly, and thus, despite that each individual tree grown greedily are not by themselves particularly robust, it is shown in (Breiman, 2001) that an ensemble of T randomly trained trees increase the variance of the classification while reducing the bias.

ERF with Contextual Prior

We use the Extremely Random Forests for interactive supervoxel classification in the following way. First, the user is asked to define a 2 layer hierarchy of labels with semantic information in a tree structure. The first layer should contain information about areas of the image the user is interested in, such as a *sky*, *vertical* and *horizontal* for a street image (Tighe and Lazebnik, 2010), or *nucleus* and *cytoplasm* for a more related 3D volumetric image. The second layer of labels aims to focus on the biology-specific labels appearing in each of the regions, such as *mitochondria* or other *vesicles* in the cytoplasm, or *nucleoli* inside the nucleus. For each of the layers a random forest is trained and the probabilistic output of the first layer is used as features in the second layer, similar to the auto-context Random Forests (Tu and Bai, 2010), (Montillo et al., 2011), just with different training data in each layer. By doing so, it reduces the variability in the training of the second layer, as different classes with similar appearance in different regions of the image (as is usual in X-ray data) are spatially separated by the contextual information of the first layer, and thus, becomes easier for the classifier to discriminate between them.

First layer classification

In practice, the user is asked to draw a few scribbles using different colors for each of the labels. Those supervoxels marked by a scribble are tagged as ground truth and a standard Extremely Random Forest is trained with $T = 50$ trees and $max_depth = 30$. For training and testing the $\mathbf{\Phi}^+ = \{\boldsymbol{\phi}_1^+, \boldsymbol{\phi}_k^+, \ldots, \boldsymbol{\phi}_n^+\}$ features from Equation 6.5 formed by $\mathbf{f}_2$ (Equation 6.11) are used. Note that these features include the absolute position of the scribbles x, y, z, and thus, the centroid coordinates of the training supervoxels are part of the training set. In other words, supervoxels of the volume will be biased towards their nearest scribble in the Euclidean space, but other factors such as volume intensity and first and second order derivatives will also play a role in the classification. In practice,

this classification will be similar to calculating geodesic distances (Kontschieder et al., 2013) from the scribbles to the volume, but with much lower computational cost. Since the desired output is a an approximate probability estimate of different image areas, these features provide an excellent compromise between complexity and robustness. Figure 6.3 shows the power of using a few annotations to spatially separate the cell in regions.

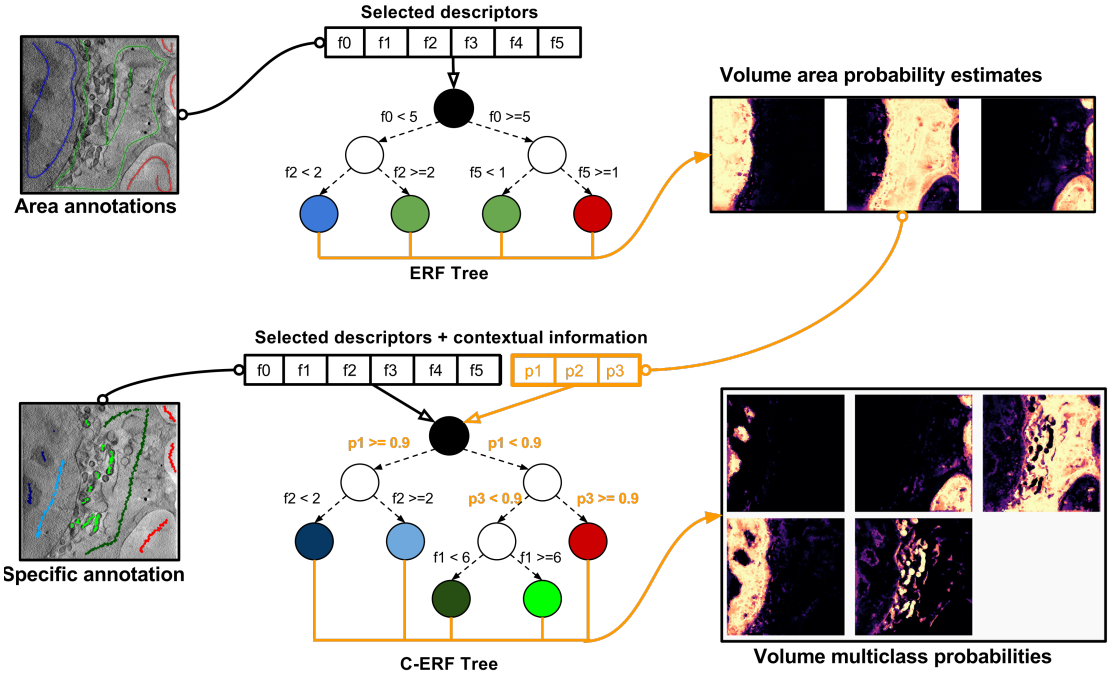


Figure 6.4: Overview of the hierarchical classification. The predictions of volume areas are used as features for the 2nd layer's contextual ERF (C-ERF), which at each node chooses the best split criterion among the features or probabilities.

Second layer classification

In this second layer of the hierarchy, the user is again asked to draw some more specific scribbles, to tag different organs/organelles within the image (such us mitochondria/-cytoplasm or nucleus/nucleoli in our later example) and those supervoxels are marked as ground truth. This time, $\mathbf{f}_1$ features are chosen to represent the texture of voxels, and a modified *contextual* Extremely Random Forest with $T = 100$ trees is trained with $max_depth = 50$.

This time each training sample i is represented by its feature vector $\boldsymbol{\phi}_i$ and the output class-probabilities of the previously trained spatial classifier (in layer 1) $\mathbf{p}_i = \{p(X_i = l) \,|\, \forall l \in \mathcal{L})\}$ with $\mathcal{L}$ the set of user defined labels. Each node then selects $t = k + d = \sqrt{K} + \sqrt{|\mathcal{L}|}$ splits tests, sampling k features from $\boldsymbol{\phi}_i$ and d features from the spatial

probabilities $\mathbf{p}_i$. Thus, each node is able to choose between an appearance feature (textural feature) or a spatial density map. Classes with similar appearance will be spatially separated, whereas appearance will be useful to separate supervoxels with different textural features.

Figure 6.4 shows a diagram of the hierarchical interactive classification approach.

6.1.4 MRF-based Segmentation Refinement with prior Label Structure

To refine labels, contextual information is taken into account. The final global volume labelling problem is formulated as a minimization of a standard MRF energy function defined over the graph of supervoxel neighbours with labels $\mathbf{c} = \{c_i\}$:

$$E(\mathbf{c}) = \sum_{s_i \in SV} E_{data}(s_i, c_i) + \lambda \sum_{(s_i,s_j)\in\mathcal{N}} E_{smooth}(c_i, c_j). \tag{6.8}$$

Here, the data fidelity term (or unary potential) E_{data} is defined by the negative log likelihood of the output of the last layer of our contextual ERF, while the pairwise potential is defined as

$$E_{smooth}(c_i, c_j) = W_{i,j}\, exp(\sigma||\mathbf{f}_{1_i} - \mathbf{f}_{1_j}||_2)\ \mathbf{C}[c_i, c_j] \tag{6.9}$$

which is a standard Gaussian similarity kernel with a label based cost matrix $\mathbf{C}$ and weighted by $W_{i,j}$, a weighting factor depending on the amount of boundary pixels that share supervoxels s_i and s_j, defined as

$$W_{i,j} = \frac{min(t, b_{i,j})}{t} \tag{6.10}$$

where $\mathbf{b} = \{b_{i,j}\}$ are the boundaries between supervoxels s_i and s_j, $t < \hat{\mathbf{b}}$ is a truncation parameter to avoid extremely large boundaries to unbalance the weighting, and $\hat{\mathbf{b}} = max(\mathbf{b})$ is the maximum amount of boundaries shared between two supervoxels. Therefore, $W_{i,j}$ is then a weighting factor in the range $[0, 1]$.

Finally, we define the label cost matrix $\mathbf{C}$ using the hierarchical label information (instead of using the standard Potts model). Here, we empirically proposed label costs $c1$, $c2$ and $c3$ with increasing cost for our hierarchical labels. Figure 6.5 shows how the 5×5 label cost matrix is weighted. Being $0 < c1 < c2 \ll c3$, costs are empirically set to $c1 = 1$, $c2 = 5$ and $c3 = 100$. Figure 6.5 shows an example setup of the matrix cost $\mathbf{C}$. This is, user knowledge about the domain is used to create a tree-like label structure that is then used to condition the MRF optimization.

The MRF is then in a form that can be efficiently approximated by Graphcut (Boykov and Kolmogorov, 2004) or QPBO (Rother et al., 2007) with α-expansion (Boykov et al., 2001) or FastPD (Komodakis and Tziritas, 2007).

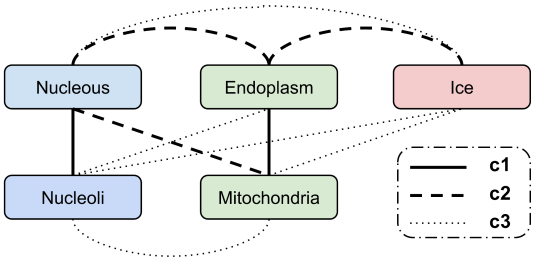


Figure 6.5: Creating the label cost matrix $\mathbf{C}$ *from label structure.* $0 < c_1 < c_2 \ll c_3$.

6.1.5 Quantitative evaluation

We empirically test our segmentation pipeline over 3 different $946 \times 946 \times 200$ cryo-SXT datasets. These datasets contain the following classes of interest: nucleus, nucleoli, cytoplasm, mitochondria and ice. For their segmentation, labels are separated in 2 stages, for the first one, nucleus, cytoplasm and ice regions are used to draw contextual information (as shown in Figure 6.3). For the second stage, the user is asked to annotate mitochondria, nucleoli and other nucleus, cytoplasm and ice supervoxels to train a ERF to learn how to discriminate between them, using both appearance features and contextual information from their position. Figure 6.3 and Figure 6.10 show a qualitative evaluation of our approach, showing 3D volume renderings of obtained segmentations.

For validation purposes, we have manually segmented 10 uniformly separated slices (1 every 20 slices) to demonstrate the segmentation accuracy (in terms of the per-supervoxel label) of our algorithm compared to a baseline ERF trained in the whole data, as shown in Table 6.1. We trained an ERF using different splits of the manually segmented 10 slices as ground truth data. The splits are formed of randomly selected 1%, 10%, 30%, 50%, 70% and 90% of the data for training and the rest for testing respectively with the aim of detecting the amount of ground truth needed (using textural sigma set features) to segment a volume. To perform the split, we used a Stratified Split 10-fold cross-validation, where the percentages of data for training and testing are taken on a per-class basis and report averages, meaning that 1% represents 1% of samples for each of the classes are used for training, and the rest for testing. Table 6.1 shows the result, indicating that overall around 30% of the data (which is the 30% of 10 slices and can be obtained with a couple of minutes of user scribbles) are enough to get a good classification for most of the volume. The table contains results for a standard ERF

with sigma set features in the first section, while the second sections shows results of our contextual ERF assuming the output of the first layer of the hierarchy is known (the separation between nucleus, cytoplasm and ice is known). Thus, it can be seen that adding contextual information to the second layer results in an performance increase as with only 10% of training data already achieves overall better results than using 50% of the data without the contextual split.

	Overall	Mitochondria	cytoplasm	Nucleus	Nucleoli	Ice
1%	0.8704	0.5655	0.9349	0.9597	0.3626	0.8760
10%	0.9319	0.6108	0.9630	0.9594	0.8200	0.9404
30%	0.9513	0.7070	0.9714	0.9662	0.9029	0.9707
50%	0.9561	0.7352	0.9759	0.9676	0.9080	0.9779
70%	0.9589	0.7632	0.9750	0.9716	0.9233	0.9810
90%	0.9638	0.8096	0.9799	0.9705	0.9150	0.9893
	Assuming contextual information is known					
1%	0.9569	0.6316	0.9709	0.9593	0.8530	0.9885
10%	0.9601	0.7314	0.9702	0.9767	0.8756	0.9911

Table 6.1: Textural SigmaSet feature performance over different classes with different training splits. 10 slices has been manually labelled for each of the datasets. Left column indicates split percentages in a per-class train-test split, meaning the percentage of training samples from each class used for training (while the remaining is uses for testing). Note that the datasets are unbalanced and 1% of nucleus is not the same amount of training data than 1% of mitochondria. Overall score shows per-volume test scores. Top section of the table shows predictions over the whole volume using a standard ERF, bottom sections shows our results using a contextual ERF.

First section of Table 6.1 is equivalent of training an standard Extremely Random Forest over all the labelling data. By using the contextual information of the previous layer as contextual features, the second section of Table 6.1 shows a boost on the accuracy by using much less annotations.

Last, in Table 6.2 we provide timings of different stages of our algorithm using a standard i5 processor with a K40 Tesla GPU (for CUDA implementations). As can be seen, after supervoxels and features are computed, the pipeline provides a near real-time experience for the user, as the classification requires around a second to predict the class of all the supervoxels of the volume, as individual trees of the ERF are trained in parallel. It is to note that by using supervoxels for both ERF prediction and MRF refinement the complexity of the problem gets reduced to an state where all the 200 slices of the volume can be segmented efficiently in a couple of seconds. The same problem formulated over the voxel grid would take a few hours to be solved.

1. Supervoxel over-segmentation	5 sec
2. Descriptor Extraction	2 sec
3. User interaction (scribbles)	-
4. Train+predict ERF	1 sec
5. Refine segmentation with MRF	∼2 sec

Table 6.2: Computational times for a $1000 \times 1000 \times 200$ *volume. Note that the user can loop steps* $3 - 5$ *hierarchically.*

6.1.6 Qualitative evaluation

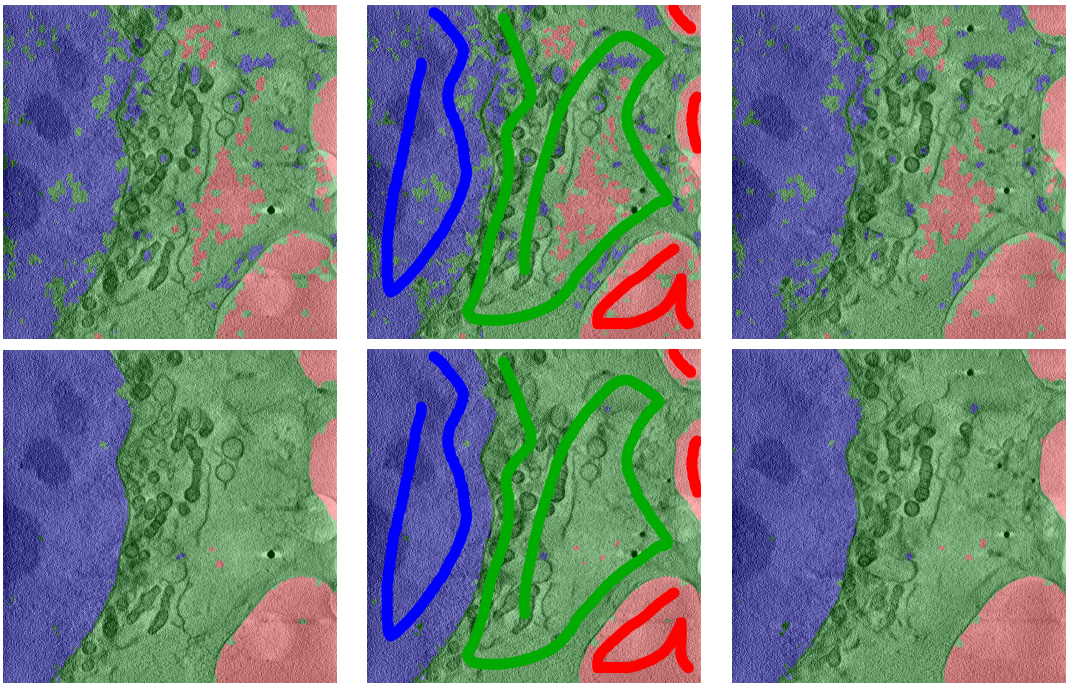


Figure 6.6: Results of interactive segmentation with supervoxels. Columns from left to right show the 15th, 50th and 85th slice of a region of interest of size $946 \times 946 \times 100$*. First row shows the prediction of a standard RF. Second row shows the result of regularizing the predictions with our graph based MRF formulation using simple features to model the appearance.*

As an interactive segmentation technique, the position and quality of user annotations have a huge impact in the quality output of the segmentation. However, our proposed MRF formulation with similarity weight and truncated *neighbouriness* weight (Equation 6.10) encourage consistency across nearby supervoxels, making the segmentation process less sensitive to user input. Figure 6.6 and Figure 6.7 show segmentation results using our interactive segmentation model based on Region Adjacency Graphs (supervoxel graphs) and learned discriminative appearance model (with ERFs) with simple yet powerful Sigma Set features. Three uniformly separated slices from the volume are shown, while scribbles are drawn *only* in the central slice. First row shows the result of

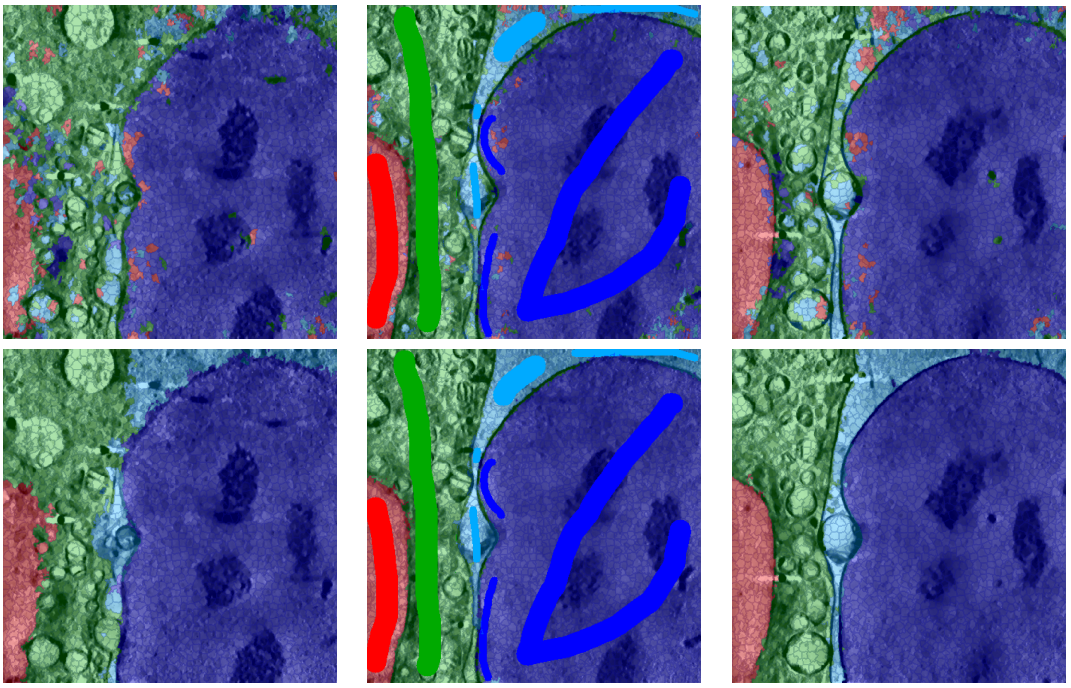


Figure 6.7: Results of interactive segmentation with supervoxels. Columns from left to right show the 15th, 50th and 85th slice of a region of interest of size $946 \times 946 \times 100$. *First row shows the prediction of a standard RF. Second row shows the result of regularizing the predictions with our graph based MRF formulation using simple features to model the appearance.*

applying the learned ERF to classify the supervoxels, while it is very fast, the result is very noisy. The appearance model learned by an ERF is not strong enough by its own to correctly classify regions of the volume. This is due to the inherent noisy appearance of it, with missing boundaries and abundant artefacts. However, by connecting supervoxels in a graph and formulating the segmentation problem with an MRF prior (or equivalently, energy-based graph partitioning), spatial consistency is introduced into the solution yielding much better results. When scribbles are drawn roughly following the boundary of the objects of interest and passing through every possible texture of the areas of interest, Sigma Set descriptors with simple local features do a great work extracting the contours of up to 100 slices of the volume. Left and right column show results for distant slices from the centre, where the segmentation has been successfully propagated. There are still some flaws, particularly in complex areas where the boundary is vanished and hard to distinguish, but those errors could be further fixed with minor user interaction. The aim of this qualitative evaluation is to show that minor and quick user input is converted into a strong global appearance model with sufficient discriminative power to segment several slices of the volume.

For comparison purposes, we also show in Figure 6.8 results on the same image, this time with much more vague annotations. First two rows show the result of applying

our proposed segmentation model (with and without MRF optimization respectively) with local features. On the other hand, last two rows show the result of applying our formulation with more robust features (that will be introduced later in section 7.3). Results show that with just few vague annotations, simple local descriptors don't have enough discriminative power to yield good results. This could be fixed, as in the previous experiment, by drawing more concrete annotations or, as shown in the last two rows, by using more robust features. When using more discriminative features the output of the ERF alone approximates the quality of our MRF model with weak local features. When the same MRF is applied with the appearance model learnt with robust features, the result gets much more accurate, allowing our model to perform well under vague annotations and thus, making it less sensitive to the user input. Nevertheless, we assume that a knowledgeable user will be segmenting the datasets interactively, and thus, weak features are sufficient for this purpose as they also boost the performance of the whole optimization (weak features take few seconds while complex features might take up to few minutes to compute).

This increase in accuracy of our model is further boosted when used with the contextual priors for segmentation of more concrete organelles. We show an example experiment with the same dataset by using the classifier trained to predict the 3 regions in Figure 6.8. The probabilistic predictions of this classifier are concatenated to the supervoxel features and a contextual ERF is trained to predict nucleus, nucleoli, cytoplasm, mitochondria and ice labels. Figure 6.9 show the results of such attempt, with first row showing the predictions of a standard ERF and second row the refined labels obtained through our MRF optimization. Results are far from desirable since the predictions of both classifiers are too noisy, mainly due to the reduced subset of input present only on the central slice and the large variance of texture across the volume. However, when used the probabilistic region predictions as priors for the contextual ERF, predictions start to become much more meaningful and less noisy (3rd row). With few and fairly quick user annotations the contextual ERF is able to extract more meaningful segmentation results. The example shows the ability of the hierarchical segmentation layers to provide contextual information and further improve segmentations with minor user interaction, just few scribbles for 1st (used as contextual priors) and 2nd hierarchical segmentation layers.

Last, Figure 6.10 shows the 3D rendering of the segmentation result after few minutes of further user interaction. It can be seen that the algorithm is able to extract meaningful information from the volume and discover the tubular mitochondrial networks (predicted by the biologist).

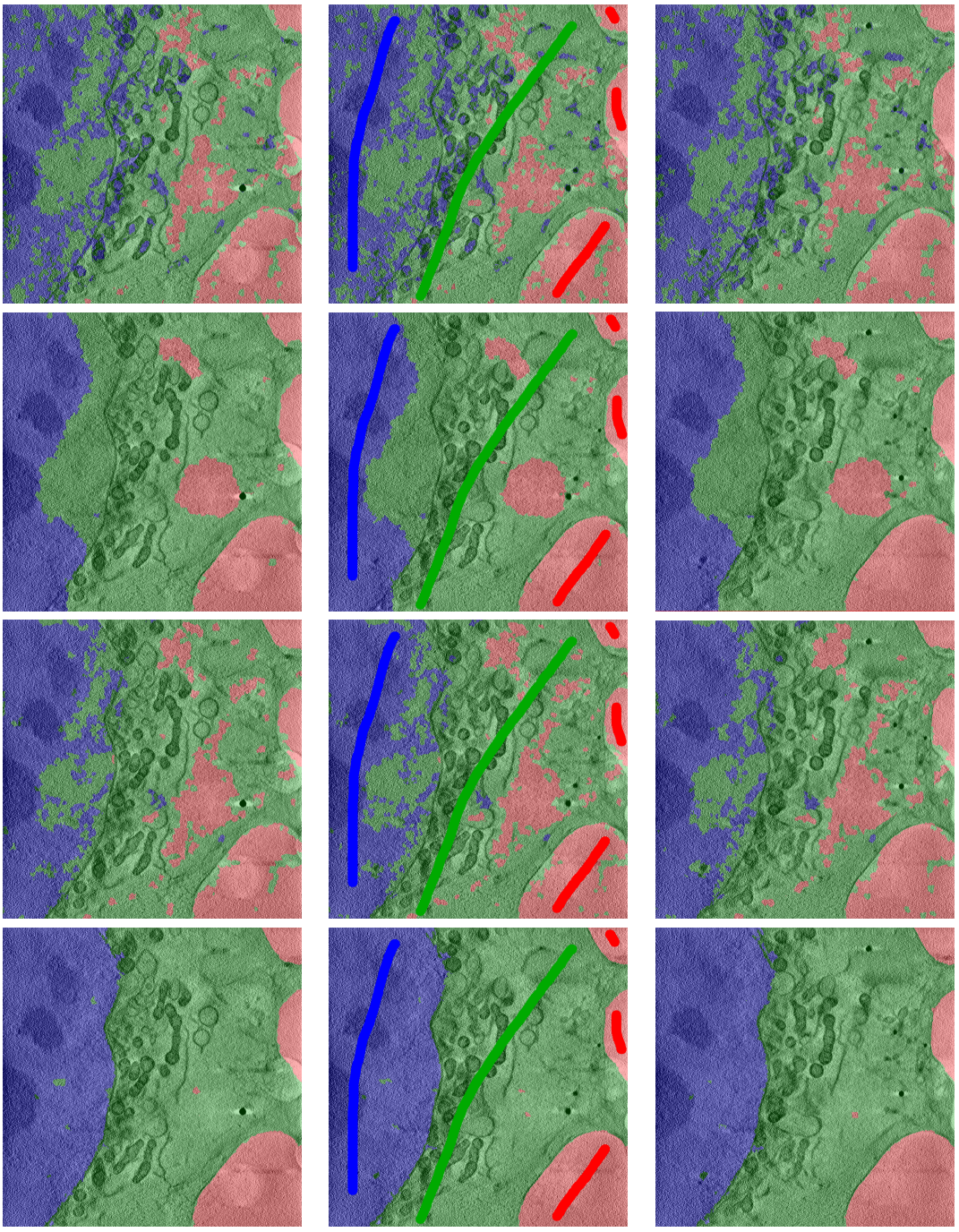


Figure 6.8: Results of interactive segmentation with supervoxels. Columns from left to right show the 15th, 50th and 85th slice of a region of interest of size $946 \times 946 \times 100$*. First row shows the prediction of a standard RF. Second row shows the result of regularizing the predictions with our graph based MRF formulation using simple features to model the appearance. Third and fourth rows show standard RF segmentation and our MRF formulation with more complex features (discussed in section 7.3).*

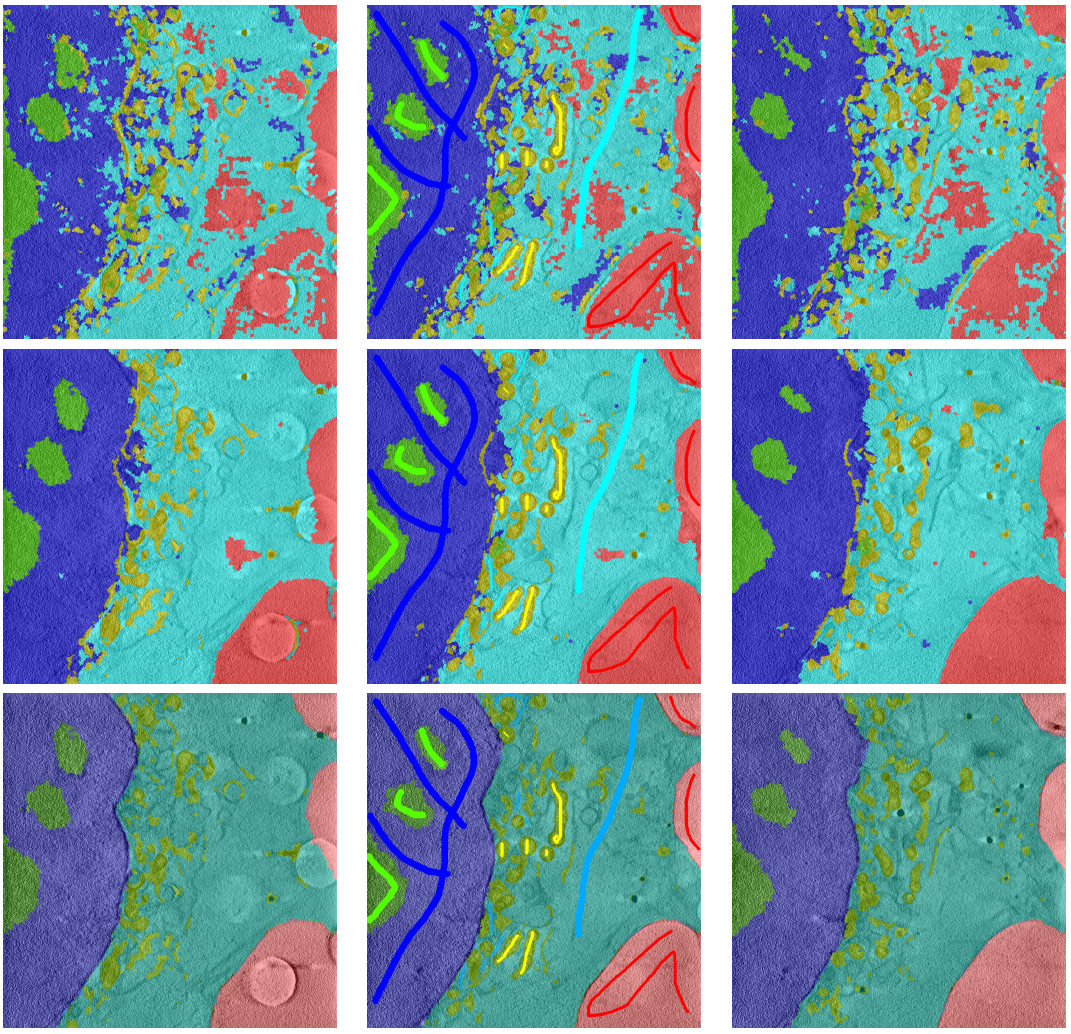


Figure 6.9: Results of interactive segmentation with supervoxels. Columns from left to right show the 15th, 50th and 85th slice of a region of interest of size $946 \times 946 \times 100$. *First row shows the prediction of a standard RF. Second row shows the result of regularizing the predictions with our graph based MRF formulation. Third shows improved results by using as contextual prior the probabilistic prediction of the classifier of Figure 6.8.*

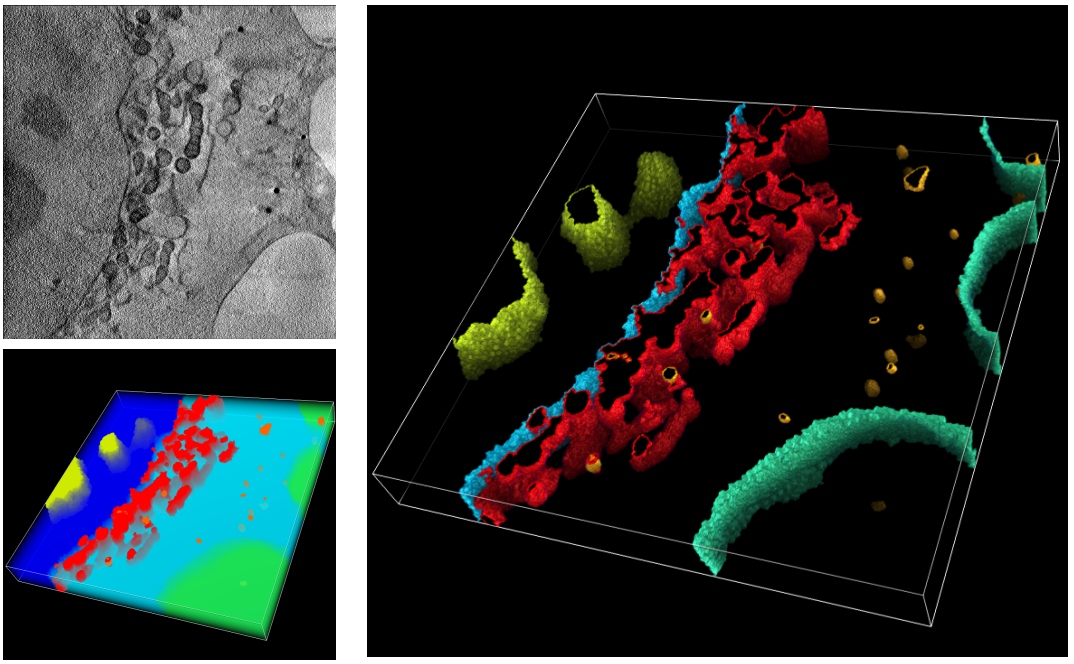


Figure 6.10: Further cell segmentation with contextual information and MRF-based refinement. The green region from Figure 6.3 was segmented again, allowing for resolution of even smaller, harder-to-distinguish objects (mitochondria: red, nucleoli: yellow and feducials: orange). Feducials where previously extracted by following a basic intensity thresholding approach (as they completely absorb X-rays and can easily be detected as spherical outliers).

6.2 Representative Sub-Volumes for Interactive Volume Segmentation

More robust MRF-based segmentation models have been proposed in the literature, such as (Lucchi et al., 2012b) where both unary and pairwise terms are learnt from the data or in (Shotton et al., 2006) where all the weights for their CRF formulation are learnt from the data and efficiently inferred. These methods perform usually better because the segmentation model used is more complex, however, it is known that more complex models need more training data to avoid over-fitting and perform poorly otherwise. Due to the very sparse annotations in interactive segmentation, pairwise potentials cannot be learnt as there is no information about the labels of the neighbours of the supervoxel being annotated.

To deal with the absence of training data and assist the human expert with the interactive segmentation task we propose an sub-region proposal with active learning approach, which we term Selective Labelling. This consists of a novel unsupervised sub-volume[2] proposal method to identify a massively reduced subset of windows which best represent all the textural patterns of the volume. These sub-volumes are then combined with an active learning procedure to iteratively select the next most informative sub-volume to segment. This subset of small regions combined with a smart region-based active learning query strategy preserve enough discriminative information to achieve state-of-the-art segmentation accuracy while reducing the amount of training data needed by several orders of magnitude.

The work presented in this section is based in the recent work of Uijlings et al. (2013) (Selective Search) which extracts a reduced subset of multi-scale windows for object segmentation and has been proved to increase the performance of deep neural networks in object recognition. We adapt the idea of finding representative windows across the image under the hypothesis that a subset of representative windows have enough information to segment the whole volume. Our approach differs from Selective Search in the definition

[2]The terms *sub-volume* and *window* will be used interchangeably along the section.

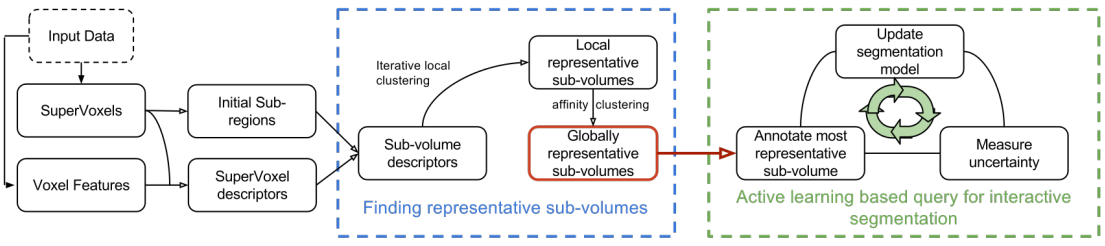


Figure 6.11: Overview of our proposed pipeline

of what *representative windows* are. Selective Search tries to find windows that enclose objects, and thus, they apply a hierarchical merging process over the superpixel graph with the aim of obtaining windows that enclose objects. Here, we adopt a completely different definition of *representative windows* by searching for a subset of fixed-sized windows along the volume that best represent the textural information of the volume. The aim is then to identify a very small subset of representative windows that properly represent all the textural variance of the volume and thus, would provide a subset of data where to train algorithms more efficiently that generalize to the rest of the volume equaly well.

Active learning techniques have been applied before in medical imaging (Top et al., 2011, 2010), but they have been focused on querying the most uncertain voxels or slice according to the current performance of the classification model in single organ medical images. The algorithm that will be presented here differ from other active learning approaches in medical imaging in:

1. It operates in the supervoxel space, making the whole algorithm several orders of magnitudes faster.

2. It first extracts a subset of representative windows which are used to loop the active learning procedure. Training and querying strategy is only applied in a massively reduced subset of windows, reducing computational complexity even further. This is, common active learning approaches have to predict the uncertainty for the whole volume (all the voxels) in order to select the next most informative regions. By pre-computing those sub-regions in advance with a completely unsupervised process, our segmentation pipeline has to predict the labels for those sub-regions, reducing dramatically the computational time.

3. The queries for the user are fixed-sized sub-volumes which are very easy to segment with standard Graphcut, Active Contours or Contour tracing techniques, or even manually by using the supervoxel graph. This makes the segmentation task much more intuitive, as the user doesn't have to look for regions of interest manually since those regions are given to them in the form of small windows.

To summarize, the main contributions of the current work can be listed as follows:

1. A novel representative patch retrieval system to select the most informative sub-volumes of the dataset.

2. A novel active learning procedure to query the window that would maximize the model's performance from a reduced subset.

3. Our segmentation framework, used as an upper bound measure, achieves similar performance to (Lucchi et al., 2012a) while being much faster.

6.2.1 Segmentation framework

To be able to segment large volumes efficiently, we combine the preprocessing, feature extraction and ERF-based classification of the previous section with the learned pairwise weights by Lucchi et al. (2012a) to segment mitochondria from Electron Microscopy (EM) volumes. Thus, we extract local textural features from each voxel of the volume:

$$\mathbf{f} = \{G^{\sigma_1}, G_x^{\sigma_1}, G_y^{\sigma_1}, G_z^{\sigma_1}, G^{\sigma_5}, G_x^{\sigma_5}, G_y^{\sigma_5}, G_z^{\sigma_5}\}, \tag{6.11}$$

where G^{σ_x} represents the input volume convolved by a Gaussian filter of $\sigma = x$ and subscript indicates directional derivatives. Supervoxel descriptors $\boldsymbol{\phi}_k$ (for supervoxel k) are then computed as Sigma Set (Hong et al., 2009) features of all the voxels belonging to them. These descriptors map the supervoxel covariance to a Euclidean space which has been proved to be a very efficient and robust for classification in sub-cellular volumes in Luengo et al. (2016a) (and previous section). To improve the accuracy and the robustness of the supervoxel descriptors, contextual information is added by appending for each supervoxel the mean $\boldsymbol{\phi}$ of all its neighbours, as initially proposed in (Fulkerson et al., 2009):

$$\boldsymbol{\psi}_k = \left[\boldsymbol{\phi}_k\,,\ \frac{1}{m}\sum_{i\in\mathcal{N}(k)}\boldsymbol{\phi}_k\right] \tag{6.12}$$

Segmentation is then formulated as a Markov Random Field optimization problem defined over the supervoxel graph with labels $\mathbf{c} = \{c_i\}$:

$$E(\mathbf{c}) = \sum_{s_i\in SV} E_{data}(s_i, c_i) + \beta \sum_{(s_i,s_j)\in\mathcal{N}} E_{smooth}(c_i, c_j). \tag{6.13}$$

Here, the data fidelity term E_{data} is defined as the negative log likelihood of the output of an Extremely Random ForestGeurts et al. (2006) (ERF), with $T = 100$ trees, trained on the supervoxel features $\boldsymbol{\psi}_k$. The pairwise potential E_{smooth} is also learnt from data (similar to Lucchi et al. (2012a)) with another ERF by concatenating the descriptors of every pair of adjacent supervoxels with the aim of modelling the *boundariness* of a pair of supervoxels.

Since the aim is to extract representative patches from the volume and give them to the user for annotation, pairwise potentials can also be learn as the sub-volumes will be completely segmented. This allows us to train the model above on the whole data and individual patches separately and use the model learnt on the whole data as an

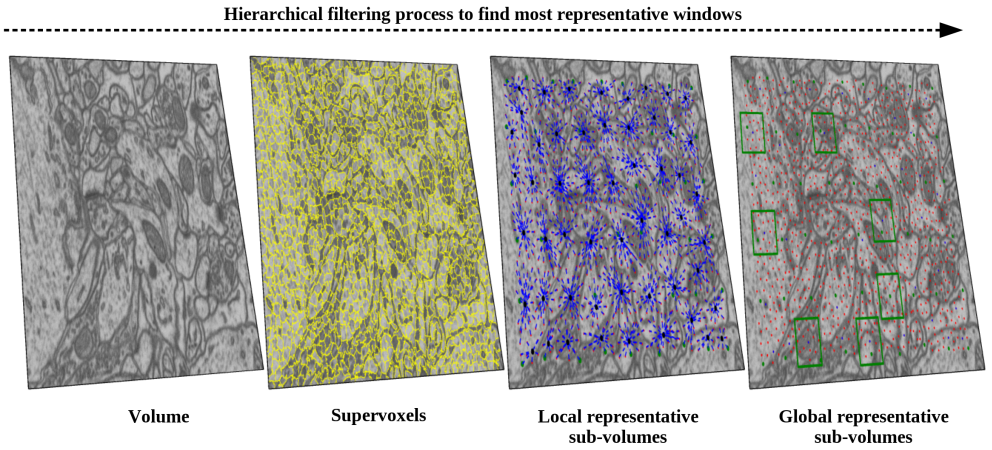


Figure 6.12: Overview of the window proposal method. For visualization purposes a 2D slice is shown, but every step is performed in 3D.

upper bound measure while using the same model but learnt from just a few patches as comparison of the quality and generalization properties of the extracted representative patches.

It is to note that the aim of this section is not to provide *the best segmentation model*, but a state-of-the-art segmentation model to use as an upper bound measure to validate the quality of the extracted sub-volumes when a more complex model is trained on them.

6.2.2 Finding representative sub-volumes

Biological volumes are usually very large (here for example $1024 \times 768 \times 330$). In order to efficiently segment them, we provide a framework to extract a reduced subset of the most representative sub-volumes which can then be used to learn to segment the rest of the volume. We start by defining a fixed size V_s for the sub-volumes, set empirically to a size that preserves information context whilst being easy (not too big) to segment. In this work, we set $V_s = [100, 100, 10]$. Considering every possible overlapping window centered at each voxel of the volume would generate too many candidates (around $200M$ voxels). Thus, we start by considering the set of proposed windows $w \in \mathcal{W}$ from N windows centered at each of the supervoxels, as we already know these regions are likely to have consistent properties (i.e. supervoxels are of homogenous nature by definition). We extract $10 \times 10 \times 10$ supervoxels which reduces the amount of windows by 3 orders of magnitude to roughly $200K$. Next, in order to extract representative regions from the image we first need to define how to describe a region. To do so, we first cluster all the supervoxel descriptors $\boldsymbol{\phi}_k$ in B bins to assign a *texton* to each supervoxel. The regional descriptor $\mathbf{r}_k$, assigned to the window proposal w_k centered at supervoxel k, is the $\ell 1$-normalized histogram of supervoxel textons in that window. Thus, $\mathbf{r}_k$ encodes the

different textural patches and the proportion of each of them present in each window. The descriptor is also rotationally invariant as measures occurrences.

6.2.3 Grouping similar nearby sub-volumes

Once sub-volume descriptors are extracted, we perform local clustering. Similar to the one SLIC uses to create supervoxels but to cluster together nearby similar sub-volumes. To do so, we first sample a grid of V_s cluster centers $C_i \in \mathcal{C}$ uniformly across the volume and assign them to their nearest window w_k. For each window we use their position $\mathbf{p}_k$ in the volume and their descriptor $\mathbf{r}_k$. Then, the local k-means clustering iterates as follows:

1. **Assign each sub-volume to their nearest cluster center.** For each cluster C_i compute the distance to each of the windows in a neighbourhood. The neighbourhood is set to $2 \times V_s = [200, 200, 20]$.

$$d(C_i, k) = \left\| \mathbf{p}_{C_i} - \mathbf{p}_k \right\|_2 + \frac{\lambda}{\sqrt{2}} \left\| \sqrt{\mathbf{r}_{C_i}} - \sqrt{\mathbf{r}_k} \right\|_2 \tag{6.14}$$

   where the first term represents the standard Euclidean spatial distance between windows and the second term is the Hellinger distance that measures the difference in appearance of the windows. Hellinger distance (also known as Bhattacharyya distance) provides a good measure between two probability distributions (e.g. the case of normalized histograms). As demonstrated in (Arandjelović and Zisserman, 2012), Hellinger distance also enhances the discriminative power of $\ell 2$ normalized descriptors used in object retrieval.

   Each window w_k is then assigned to the neighbouring cluster C_i(label L_k) that minimizes the above distance.

2. **Update cluster centers.** The new cluster center is the assigned the window to minimizes the sum of differences with all the other windows, or in other words, the window that best represents all the others assigned to the same cluster:

$$C_i = argmin_{k \in \{k | L_k = i\}} \sum_{j \in \{j | L_j = i\}} d(k, j) \tag{6.15}$$

   which has similarities with the centroid update of the k-medioids algorithm (Kaufman, 1987), which was originally developed to cluster $\ell 1$ normalized samples, as it is known that k-means behaves well and have strong convergence properties only in centered Euclidean spaces.

The above update is very efficient as the amount of windows has decreased and clusters nearby and similar windows into a even smaller set. After 5 iterations (or until convergence but we saw no benefit in runing more than 5) of the above procedure, the number of proposal windows $w_k \in \mathcal{W}$ is reduced from $200K$ to 3500 by only considering the windows that best describe their neighbouring windows $w_{C_i} \in \mathcal{W}$. Let us refer to this reduced set of windows as $\mathcal{R}$.

6.2.4 Further refining window proposals

After filtering the window proposals that best represent their local neighbourhood, still a large number of possible sub-volumes remain. To further filter the most representative regions from $w_k \in \mathcal{R}$ we apply a affinity propagation based clustering (Dueck and Frey, 2007). Affinity propagation clustering is a message-passing clustering that automatically detects *exemplars*. The inputs for affinity clustering consist of an *affinity matrix* as the connection weights between data points and the *preference* of assigning each of the data points as *exemplars*. Then, through an iterative message-passing procedure the affinity propagation refines the weights between data points and the *preferences* until the optimal (and minimal) subset of *exemplars* is found. After local representative regions are extracted from section 6.2.3, the pairwise similarity between all the remaining regions $w_k \in \mathcal{R}$ is extracted as

$$a(i,j) = intersection(\mathbf{r}_i, \mathbf{r}_j) \tag{6.16}$$

to form the $M \times M$ affinity matrix A, where $A_{i,j} = a(i,j)$ is the similarity (only in appearance, measured by the *intersection* kernel) between all pairs of windows w_i and w_j. The *preference* vector P is set to a constant weighted by the ℓ_∞ norm of the appearance vector $P_i = \gamma\,(1 - \|\mathbf{r}_i\|_\infty)$. The ℓ_∞ norm of a vector returns the maximum absolute value of the vector. For a ℓ_1 normalized histogram is a good measure of how spread the histogram is. Thus, the weight $(1 - \ell_\infty)$ will encourage windows that contain wider variety of textural features to be selected. This is a desired feature, since we aim to extract a very small subset of window proposals for the whole volume, we would expect them to represent all the possible textural features of the volume or if not the training stage will fail to model unrepresented features. After the affinity propagation clustering, we now have a manageable set of < 100 sub-volumes which together represent the global appearance of the whole volume. Let us denote this final subset of proposals as $\mathcal{P}$.

6.2.5 Querying the next most informative sub-volume

The active learning cycle starts once a minimal representative set of sub-regions $\mathcal{P}$ has been extracted and at least 1 window (containing both foreground and background) has been segmented by the user. From there, the segmentation model from section 6.2.1 is trained and used to predict the labels of all the supervoxels belonging to all the windows in $\mathcal{P}$. Here, we average the probabilistic prediction of all the trees $t \in T$ of the ERF in order to model the probability of a supervoxel to belong to foreground or background. The uncertainty of the predictions is then estimated as the entropy:

$$U_x = -\sum_y p(x|y) \log p(x|y) \tag{6.17}$$

Then, the average uncertainty of all the supervoxels U_s in a window $w_k \in \mathcal{P}$ is defined as the average uncertainty in the predictions of all the supervoxels contained in that window. Similarly, the average uncertainty of *boundariness* U_e of all connected pair of supervoxels in a window is extracted from the other ERF trained to identify their boundariness. The average window uncertainty is then defined as $U_w = U_s + \beta\, U_e$.

The window with larger average uncertainty is selected as the next sub-volume to be segmented. As all the windows have been previously reduced to a minimal subset, the query strategy is very efficient and is able to return a globally representative sub-volume that, if annotated, would maximize the performance increase of the segmentation models.

6.2.6 Experiments evaluation

In our experiments we used the publicy available EM dataset[3] used in (Lucchi et al., 2012a). The data set consists of a $5 \times 5 \times 5 \mu m$ section taken from the CA1 hippocampus region of the brain. Two regions of interests of size $1024 \times 768 \times 165$ (one for training and the other one for testing) belonging to the same larger volume are available, where mitochondria are manually annotated. We first validate the results of our segmentation pipeline by using one of the volumes for training while the other for testing. Table 6.4 shows results of different stages of our segmentation pipeline:

1. **ERF**$_{raw}$ evaluates only the prediction of the ERF trained in the supervoxel features. This is our baseline method just by using a standard Extremely Random Forest. The predictions of ERFs are inherently noisy as shown in previous sections.

[3] http://cvlab.epfl.ch/data/em

Table 6.3: Performance of our segmentation pipeline in the testing dataset

	$\mathbf{ERF}_{raw}$	$\mathbf{ERF}_{nh}$	$\mathbf{MRF}_{nh}$	$\mathbf{MRF}_{learnt}$
Accuracy	0.975	0.984	0.987	0.991
DICE coefficient	0.751	0.825	0.851	0.871
Jaccard index	0.601	0.702	0.743	0.780

2. $\mathbf{ERF}_{nh}$ is the prediction of the ERF after aggregating neighbouring supervoxel features. Concatenating descriptors of neighbouring supervoxels help to add contextual information to the classifier, and thus improve the accuracy.

3. $\mathbf{MRF}_{nh}$ is (b) refined with a contrast-sensitive MRF. This is, the predictions of the random forest are used as unary potentials in an MRF optimization problem. Supervoxels are connected together by a Region Adjacency Graph (RAG) as in section 6.1 to encourage labelling consistency across neighbouring supervoxels.

4. $\mathbf{MRF}_{learnt}$ is the full model with learned unary and pairwise potentials. Similar to (c) the learn appearance model from ERFs is used as unary potentials, but the pairwise potential is also learnt by concatenating descriptors of neighbouring supervoxels trying to learn the likelihood of two supervoxels being merged together.

Our model, used as an upper bound of the *maximum achievable accuracy* of the following experiment. It has similar segmentation performance to the one reported in (Lucchi et al., 2012a), while being much faster (15 minutes of processing and training time vs 9 hours).

Table 6.3 shows a benchmark of the quality and descriptive power of a reduced subset of our extracted windows. To evaluate the quality of our extracted windows, we simulate different user patterns:

Random User : will define the behaviour of a user selecting n random patches for training across the training volume. However, we only accept the completely random patch sampling if at least 1% of the voxels in the selected sub-volumes contain foreground labels.

Random Oracle will select n random patches for training centered in a supervoxel that belongs to mitochondria (thus, assumes ground truth is known and simulates the user clicking in different mitochondria). By selecting patches centered in a supervoxel belonging to the foreground it guarantees that enough foreground voxels will be selected.

Selective Random simulates a user choosing n windows at random from a reduced subset of windows $w_k \in \mathcal{P}$ obtained using our sub-volume proposal algorithm. This is, instead of randomly selecting n windows from the whole volume, it will select n windows randomly from our reduced pool of proposed windows. It is to note that the sub-regions were proposed in a completely unsupervised fashion, and thus, its random behaviour will serve as a comparison of the *quality* of the proposed windows. Similarly to Random User, a random selection from our region proposal pool is accepted only if at least 1% of the voxels contain foreground objects.

Selective Labelling will select the first window at random from $w_k \in \mathcal{P}$ (containing both background and foreground) while the next $n-1$ sub-regions will be selected by our active learning based query strategy proposed in subsection 6.2.5. It will show that by querying the sub-volumes in intelligently, the accuracy will be boosted while making the annotation process much more intuitive for the user.

In this experiment we aim to evaluate how well our proposed regions and active learning approach generalize from few annotated windows to the rest of the volume (in this case a test volume). In a real scenario, the regions of interest would be given to a user for manual annotation, which could be easily segmented by our previously presented interactive segmentation method (section 6.1), any interactive segmentation method in the literature of by manually annotating super-regions (described in subsection 4.7.3). Figure 6.13 shows 10 example regions from the representative subvolumes along with SMURFS Supervoxels (in 3D) segmentation. It can be seen that with just 20 supervoxels whole $100 \times 100 \times 10$ regions can be properly represented, and mitochondria could be extracted with just a few user clicks. The image also shows the variability of the proposed regions, as different kind of organelles (microtubules, vesicles and mitochondria) along with background and random noise are present in them. For this experiment, however, we will assume that the user has annotated them and use the ground truth labels for the selected windows as user input. This will allow us to evaluate the quality of the proposed windows assuming a proper segmentation is given. Considering that user annotations are noisy and ground truth annotations were manually labelled, this is not far from a real use case. Then, all different patterns are trained only on the selected windows of the training volume (with the full model) and tested in the whole testing volume. All the user patterns are averaged from 100 runs and results are shown in Table 6.4.

Few interesting results can be extracted from the above table. Since the volume is split in foreground and background labels even the completely random subvolume query performs decently in some situations. It is, however, extremely variable, as in our previous experiments (published paper (Luengo et al., 2016c)) the scores were all similar for the other 3 approaches, but a bit higher for the completely random approach. The second

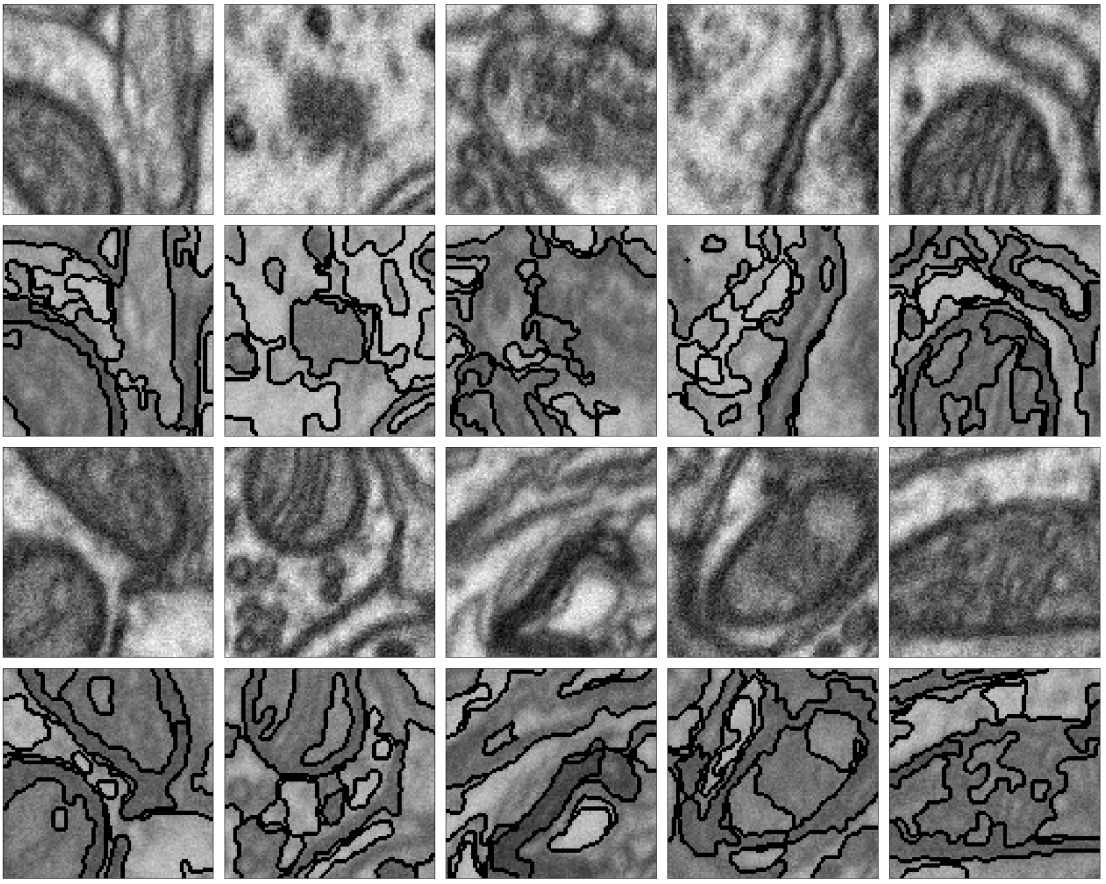


Figure 6.13: Example representative sub-volumes extracted in an unsupervised fashion.

Table 6.4: DICE coefficient of the simulated retrieval methods. Percentages indicate fractions of total training data.

	Random User	*Random Oracle*	*Selective Random*	*Selective Labelling*
3 sub-volumes (< 1%)	0.213	0.661	0.653	0.787
5 sub-volumes (1%)	0.254	0.735	0.741	0.794
10 sub-volumes (2%)	0.299	0.763	0.760	0.821
30 sub-volumes (5%)	0.510	0.810	0.807	0.862

and probably the most interesting one is that our extracted windows without the active learning (Selective Random) achieve similar performance to the Random Oracle (which assumes ground truth is known). This proves the quality of our windows as our unsupervised method is able to represent properly all the textural elements of the volume. It is to note that Selective Random does considerably better (by a large margin) than Random User, despite of both using the same region query strategy (random selection). This also strengthens our premise that globally clustered regions contain enough discriminative information to characterize the whole volume, and thus, selecting regions from this pool (even randomly) has guarantees to select windows with enough discriminative potential.

With the active learning, our method outperforms all the others and is able to obtain similar performance to the baseline trained in the whole volume (table 6.3) with much fewer training data (up to 5%). By looking at the results we can conclude that, as discussed at the beginning of the chapter, within volume segmentation can be achieved by intelligently segmenting a small part of it (up to 5%). It does also give insight into the future of interactive volume segmentation, as it is widely understood that large parts of the volume (at least half or a quarter) are needed for a proper segmentation propagation, while segmenting large parts of the volume is very tedious and time consuming. Thus, we not only show that whole volume segmentation can be achieved with much less user input, but we also propose an interactive approach that makes the segmentation much more intuitive for a user.

In practical applications, after the volume to be segmented is loaded, instead of giving the whole volume to be segmented to the user, which is sometimes very overwhelming as it is not trivial where to start performing segmentation, the user would be given small windows easy to segment. After few windows are segmented the whole volume could be segmented for visual inspection and interactive refinement (as in section 6.1).

As a final discussion, we could say that the segmentation model proposed for segmentation is bounded by its design, not the amount of training data. The model achieving similar results with a reduced subset of the training set (5%) than the whole training set suggests that adding more data to the model wouldn't improve its segmentation accuracy indefinitely, but would start to decrease at some point due to over-fitting. Thus, while adding a few more windows will improve its accuracy further, segmenting all the windows in the volume would surely not help much. If enough sub-volumes are annotated, more complex models could be trained, such as the 3D Deep Convolutional Neural Networks recently introduced by Çiçek et al. (2016), although this work is mainly focused on frameworks towards interactive segmentation of large volumes, and thus, evaluation of better models is out of the scope of this work, specially since the aim of this thesis is to present solutions and frameworks to scenarios with none or very few existing annotations.

6.3 Extremely Contextual Forests

In the previous sections a framework towards interactive segmentation was presented. Learning to segment from user scribbles has proven to be very effective and intuitive with near real-time feedback due to the usage of pre-calculated super-regions. However, limitations become apparent when multiple objects of interest are present and the low

signal-to-noise ratio difficult their detection and segmentation. The limitations of interactive segmentation methods is characterized by the low amount of input that they receive. It is known that more complex models tend to behave better give more accurate predictions for any kind of problems, since they have the ability to model more complex interactions. However, the accuracy of a complex model is very closely related to the amount of data available for training it. Without sufficient training data simple models tend to give bad yet reasonable predictions since they *underfit* the underlyinh data, whereas complex models have a lot of ease to *overfit* the training data and give terrible predictions for the test data (example on Figure 6.14).

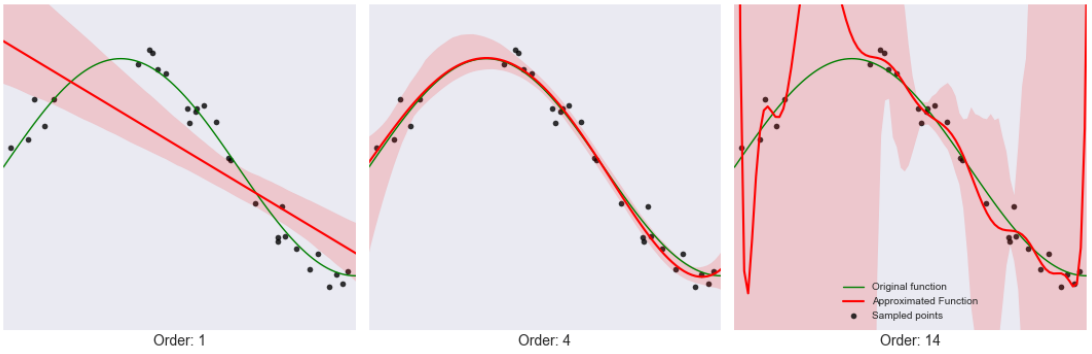


Figure 6.14: Example of difference between model underfitting (simple models) and overfitting (too complex models) when scarce data is available. The green line shows the true function to be approximated ($f(x) = \cos(1.5\pi x)$) and blue dots shows randomly sampled points with added noise. From left to right: linear regression model estimation with different polynomial order for the models. It can be seen that a polynomial of order 1 (straight line) underfits the model whereas a polynomial of order 14 overfits the given data-points, resulting in even higher error. Red regions shows 95% confidence interval for the prediction model.

In section 6.1 we introduced an initial segmentation model based on learning an appearance model for supervoxels from user annotations to segment volumetric datasets with Markov Random Fields priors. When input data is very scarce or there are plenty of small and difficult to segment objects of interest in the volume, this approach does not generalize well and fails to accurately segment objects. Better features or more input annotations would overcome the issue, but hand-crafting features is tedious and specific for each volume which requires high image processing knowledge, and requesting more input annotations is sometimes not very trivial, as the user rarely understands where and how to provide them to maximize the learning of the classifier. Thus, blindly requesting more annotations might result in a huge ground truth class unbalance, which would only make it harder for the classifier to learn anything.

Later in section 6.2 we introduced Selective Labelling, an approach to extract subvolumes from a large 3D dataset with the aim of reducing the user annotations needed to

segment a given volume. With this sub-volume extraction technique the input of the user is focused in specific windows, locally easier to segment, and allow for an Active Learning procedure to manage the next window to be segmented. By fully segmenting small windows, using either the process on section 6.1 or any other interactive segmentation technique in the literature (chapter 8), more complex models can be trained. Since full sub-volumes are now available, models that learn the relationship of not only supervoxels, but also interactions between adjacent supervoxels can be learnt. With this, we did not only improve the accuracy of an interactive pipeline previously presented, but we also showed that only a fraction of a volume is needed (about 3-5%) to segment the rest of it, which goes against the current belief that large regions (1/3 to 1/2 of the volume) are necessary, and thus, manually segmented.

In this section a more robust yet efficient segmentation model will be presented, which is able to learn to better model more complex appearances with the same input data and give joint predictions for voxels or supervoxels. We call our model Extremely Contextual Forests, as it is built on top of Entangled Decision Trees (Montillo et al., 2011) and the Random Forest regressor for cell detection of Kainz et al. (2015) and exploits the 3D structure of the datasets to use contextual information from neighbouring nodes to predict the class of each one.

6.3.1 Extremely Contextual Forests for Volume Segmentation

An Extremely Contextual Forest (ECF) is an ensemble of Extremely Contextual Decision Trees (ECDT), which are a novel set of decision trees designed to take advantage of the underlying dataset structure and exploit the 3D spatial plane to improve the appearance model.

Most of the machine learning models focus on learning appearance models for either the voxel level or the supervoxel level. Here, however, instead of making a distinction for them, we will incorporate both supervoxel features and voxel features into a joint modelling framework. Voxel-level features will be extracted as before, while supervoxel-level features will this time be represented by mean descriptors (of al the voxels of the volume). Since both descriptors have the same length and measure the same properties, both supervoxel data and voxel data can be used to jointly model the limited available training annotations. In this section we will assume that, as in previous chapters, the training data is limited or not available, and thus, we will use the approach presented in the previous section to extract and label representative volumes from the data. This section will then introduce, a more complex model that is able to learn form voxels

and supervoxels jointly to maximize the usage of the limited data, and then be used to predict either voxels of supervoxels.

The aim of the Extremely Contextual Forests is then to increase the amount of training examples without increasing the amount of input annotations. By learning over the voxel grid more robust features classifiers can be learn as more data samples exist. The features for each of the voxels are more noisy compared to the ones of the supervoxels, as biomedical datasets usually contain poor signal to noise ratio and this is reflected on the voxel level. By using supervoxels, the computational complexity is not only reduced, but the segmentation accuracy becomes higher as the descriptors over the supervoxel grid help dealing with the high local variance. To be able to use the voxel grid for training without losing powerful descriptors ECF is built on the grounds of Entangled Decision Trees (Montillo et al., 2011), where is shown that sampling patches[4] from the vicinity of a voxel greatly increases the accuracy of the classifier. The framework proposed in (Montillo et al., 2011) selects as features for a given voxel the mean intensity of a randomly sampled patch at random distance and with random radius (or the difference in intensity between two random patches). It helps reducing the local noisy variance on the neighbourhood of a voxel and it works very well for the segmentation of organs in a human body, where the intensity of a voxel is good at discriminating organ type, but it still does not provide robust enough features for the segmentation sub-cellular organelles. Extremely Contextual Forest extends the work from Montillo et al. (2011) by

1. Extending the framework to multiple feature channels.
2. Using not only the mean, but also the higher order central moments of the feature distribution within a patch.
3. Generalizing the framework to multi-scale segmentation by using voxels, supervoxels and megavoxels.

6.3.2 Extremely Contextual Decision Trees

Let $\boldsymbol{x} = \{\boldsymbol{x}_1, \cdot, \boldsymbol{x}_N\}$ be a set of N data points and $\boldsymbol{y} = \{y_1, \cdot, y_N\}$ their corresponding ground truth labels. For the segmentation of biological volumes a data point represents a voxel within the volume. Different from previously seen machine learning techniques, ECDT is not train over the $\{\boldsymbol{x}, \boldsymbol{y}\}$ space, but over the spatial mapping ϕ, with $\psi(p) = \{\boldsymbol{x}_p, y_p\}$. This will allow us to easily define context operations during training without loss of generality.

[4]In this chapter we refer as patches to a small 3D region, not just 2D.

An ECDT then receives a set of training samples $\boldsymbol{p} = \{\boldsymbol{p}_1, \cdot \boldsymbol{p}_N\}$ where $p_i = \{x_i, y_i, z_i\}$ and whose feature mapping $\psi(p_i)$ is described by low-dimensional feature vectors $\boldsymbol{x}_i = \{f_1, \cdot, f_K\}$ (due to memory limitations) and its ground truth label y_i. Each of the nodes of an ECDT uses as features the four central moments (mean, variance, skewness and kurtosis) of randomly sampled patches as features as illustrated in Figure 6.15. By doing so, an ECDT can learn to model the appearance of the neighbourhood of each pixel, its texture or variance, and even to model the shape of the intensity distribution of the feature patches (with skewness and kurtosis), a much more robust yet efficient to compute (as seen later) descriptor of a voxel that adds contextual information from neighbouring voxels into account.

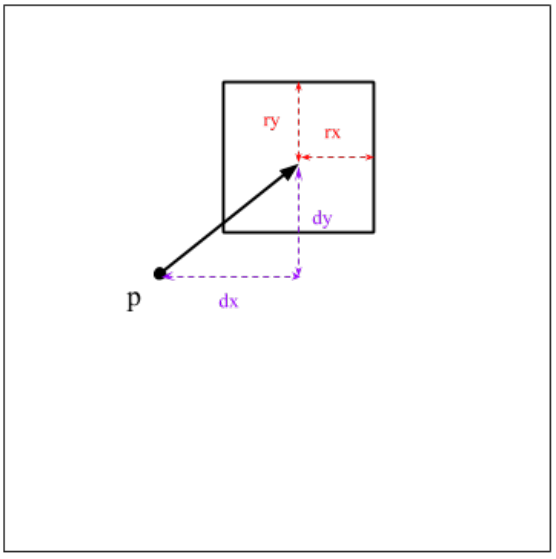


Figure 6.15: For every voxel of the volume p, features are automatically computed by randomly sampling a displacement vector $\boldsymbol{d}$ *and a radius vector* $\boldsymbol{r}$*. The moments of the feature responses inside that patch will be used as feature responses for p, e.g. mean response for 1st-order moment, standard deviation of feature responses in the patch for the 2nd-order moment and so on. Note that image shows 2D displacements and radius for visualization purposes but they are applied in 3D in practice.*

Split Functions

The split function of a node can be described as an axis-aligned hyperplane, similar to a Random Forest, but this time, aligned to an axis of a mapped feature space *phi* given by a random patch at displacement $\boldsymbol{d} = \{d_x, d_y, d_x\}$ and of radius $\boldsymbol{r} = \{r_x, r_y, r_x\}$, yielding

$$h(\boldsymbol{p}_i; \boldsymbol{\theta}_j) = [\phi(\boldsymbol{p}_i) > \tau] \tag{6.18}$$

where ϕ is defined over a patch of size $\boldsymbol{r}$ on a given feature channel at a displacement of $\boldsymbol{d}$. The split function is then parametrized by $\theta_j = \{\boldsymbol{d}, \boldsymbol{r}, \tau, w, o\}$ where w stands for the index of the feature to choose ($x_{i,w}$) and o the moment order ($1 \leq o < 4$) for the four central moments described above. Thus, the mapping ϕ can be described as,

$$\phi(\boldsymbol{p}_i) = moment_o(\{x_{z,w} \quad \forall z \in \mathcal{N}(i)\}), \tag{6.19}$$

where $\mathcal{N}(i)$ describes all the voxels in a patch of radius $\boldsymbol{r}$ at displacement $\boldsymbol{d}$ from $\boldsymbol{p}_i$, this is, all the voxels within the $[\boldsymbol{p}_i + \boldsymbol{d} - \boldsymbol{r}, \boldsymbol{p}_i + \boldsymbol{d} + \boldsymbol{r}]$ region. By using the axis-oriented hyperplanes on a transformed space of volume patches more robust appearance models can be learnt with little complexity.

Training Extremely Contextual Decision Trees

During training the search space for an optimal split is prohibitively to broad as it involves every possible patch of arbitrary size and at random displacement. Thus, the range of the parameter is initially limited by two parameters $\boldsymbol{d}_{max}$ and $\boldsymbol{r}_{max}$ which control the maximum displacement and maximum radius from each voxel. This parameters are problem specific, and low values (e.g. $\boldsymbol{d}_{max} = \boldsymbol{r}_{max} = \{5, 5, 5\}$) work well when looking for small objects while larger values (e.g. $\boldsymbol{d}_{max} = \boldsymbol{r}_{max} = \{50, 50, 50\}$) allow the classifier to gather long range information to better predict large regions. Similarly, we introduce another two parameters that represent the minimum displacement and radius $\boldsymbol{d}_{min}$ and $\boldsymbol{r}_{min}$. It can be seen that by setting $\boldsymbol{d}_{min} = \boldsymbol{r}_{min} = \boldsymbol{d}_{max} = \boldsymbol{r}_{max} = \{0, 0, 0\}$ we recover a standard Decision Tree that predicts the label for a single voxel based on its feature channels (if we ignore the moment calculation described in the previous section).

We then introduce randomness during the training process at the node level, similar to on Extremely Random Forests (Geurts et al., 2006) and Regression Trees for cell detection (Kainz et al., 2015), by defining $k = \sqrt{d}$ as the number of features to test and ρ to control the number of tests to sample for each feature. For each test $t \in \{1, \ldots, \rho\}$ a random displacement vector $\boldsymbol{d}$ with $\boldsymbol{d}_{min} \leq \boldsymbol{d} \leq \boldsymbol{d}_{max}$ and random radius for the patch $\boldsymbol{r}$ with $\boldsymbol{r}_{min} \leq \boldsymbol{r} \leq \boldsymbol{r}_{max}$ are sampled. Let then $\boldsymbol{\Theta}$ be the set of ρ parameters sampled for displacement vectors, radius vectors, order of the moments and threshold for the feature mapping τ (as shown in Figure 6.15). Among those displacements and radius vectors the optimal one is evaluated according to the aforementioned split function

$$\boldsymbol{\theta}_j^* = \underset{\boldsymbol{\theta}_j \in \boldsymbol{\Theta}}{\arg\max} \quad I_G \tag{6.20}$$

where the information gain I_G is defined as gain according to the Gini Coefficient of nodes that would be separated onto left $\mathcal{S}_j^L$ and right $\mathcal{S}_j^R$ sets respectively:

$$I_G = Gini(\mathcal{S}_j) - \frac{N^L}{N} Gini(\mathcal{S}_j^L) - \frac{N^R}{N} Gini(\mathcal{S}_j^R) \tag{6.21}$$

Thus, for each split $\boldsymbol{\theta}_{j,t} \in \boldsymbol{\Theta}$, the o-th order moment mm is calculated for every data point $\boldsymbol{p} \in \mathcal{S}_j$ in a patch at $\boldsymbol{d}_t$ displacement and of radius $\boldsymbol{r}_t$. The data point is then set to go left if $mm < \tau_t$ or right otherwise. Selecting the $\boldsymbol{\theta}_j^*$ that maximizes the information gain is then straight forward.

We finally introduce the last parameter of our ECDT which is γ and defines the maximum number of data points on which to compute the information gain I_G (and thus, the split function). This parameters allows us to control the efficiency during training, at the cost of reduction on the robustness of the learned split parameters. It can be seen that for $\gamma = |\mathcal{S}_j|$ every possible data point would be evaluated and thus, the parameter would be ignored and the training proceed as an standard randomized tree. However, for a fixed $\gamma \ll N$ (e.g. $\gamma = 200$), where N is the total number of voxels to train from on the whole volume, the split function would be evaluated only in a subset of the samples that arrive to node j (or to all of them if j is a node at high depth and $\mathcal{S}_j > \gamma$). It might look counter intuitive to limit the training samples this way, as the split parameters would be learned from a small subset and thus might not generalize well. However, we will show that it increases the training speed dramatically without altering the overall accuracy. This is due to the fact that extremely randomized trees are usually very deep, and thus, the function of the root nodes is just to hierarchically split the sample space by routing the samples to deep nodes, where less samples arrive and split functions become more meaningful. Thus, it makes sense to limit the samples on which to test the split functions at the starting nodes that would contain all the data points from the dataset, which in this case it could be a very large 3D volume. As a result of this, nodes will travel quickly at the top nodes of the tree until their numbers are reduced and then the learned split functions will have a larger impact on their classification.

Predicting with Extremely Contextual Decision Trees

Prediction (or testing) of a data point is straight forward as in other ensembles. A data point $\boldsymbol{p}$ to be tested is routed from the root node until it reaches a leaf node by applying the learnt split functions at every intermediate node and deciding whether to route it to the left or the right branch.

Once it arrives to a leaf node, its label is predicted as the probability class distribution of the samples that arrived to that node during training.

Efficient Implementation Details

Since features are going to be extracted on-the-fly, efficient methods for the calculation of the feature responses are needed. Calculating statistics (of 1st to 4th order) of a patch of size $M_1 \times M_2 \times M_3$ is computationally expensive when it need to be calculated for every voxel of the training set multiple times for each node on the tree. In order to speed-up the feature calculation Integral Images are used. Integral images were first introduced to vision by (Viola et al., 2003) (or at least, it is when they gained the attention) and represent summed area tables (or cumulative sums) of all the pixels in the image. The interesting property they contain is that the sum of all the pixels in a 2D region can be obtained by 4 arithmetic operations in the integral image. The extension to 3D is trivial, taking 8 operations to compute the sum of intensities within a ROI. (Phan et al., 2012) (Section 3.1.2) contains detailed derivation of how to use integral images to compute moments in a region of interest. Let s_1, s_2, s_3 and s_4 be the sum over the intensities of the region of interest defined as,

$$s_i = \sum_{x,y \in \mathcal{N}(i)} I^i_{x,y}. \tag{6.22}$$

The mean μ, standard variation σ, skewness δ and kurtosis κ of the feature responses in a given patch of size $S = M_1 \times M_2$ are given by:

$$\mu = s_1/S \tag{6.23}$$

$$\sigma = \sqrt{\frac{1}{M_1 M_2}\left(s_2 - \frac{s_1^2}{M_1 M_2}\right)} \tag{6.24}$$

$$\delta = \frac{1}{M_1 M_2 \sigma^3}\left(s_3 - 3\frac{s_1 s_2}{M_1 M_2} + 2\frac{s_1^3}{(M_1 M_2)^2}\right) \tag{6.25}$$

$$\kappa = \frac{1}{M_1 M_2 \sigma^4}\left(s_4 - 4\frac{s_1 s_3}{M_1 M_2} + 6\frac{s_1^2 s_2}{(M_1 M_2)^2} - 3\frac{s_1^4}{(M_1 M_2)^3}\right) \tag{6.26}$$

It can be seen that the moment calculation is now independent on the size of the region, assuming that the integral images are precomputed prior to training. This allows for super fast feature extractions with few arithmetic operations for each of the regions. Extensions to 3D are straight forward by extending the integral images to 3D, as the above formulas don't change. For more details have a look at (Tapia, 2011).

6.3.3 Multi-scale generalization: from voxels to supervoxels

It was previously mentioned that the model trained on voxels and supervoxels could then be used to predict either of them. However, it was discussed in previous sections that

the mean descriptor of a supervoxel (extracted as the mean of all the voxel descriptors of all voxels within a supervoxel) fails to model some properties for supervoxels with high variance in texture. This limitation is particularly strong when compact supervoxels are used (such as the ones of SLIC (Achanta et al., 2012)), as their homogeneous shape makes them capture a large variation in texture, and thus the mean supervoxel descriptor fails to model all that variability. This can be overcome by either using Sigma Set features as in previous section, or by using more robust super-regions, such as SMURFS (Luengo et al., 2016d) (subsection 4.7.4). Using Sigma Set wouldn't allow us to use the same classifier for voxel and supervoxel classification, as Sigma Set extends the supervoxel descriptor into a (flattened) covariance-like matrix. Instead, we define the supervoxel classification problem as maximum voting of the classification of a fraction of voxels within the supervoxels. This is, let γ be a constant with $0 \leq \gamma \leq maxsize$ representing the number of voxels to sample from a supervoxel, where $maxsize$ represents the size of the largest supervoxels on the volume. For a given supervoxel s, maximum of γ voxels will be randomly sampled without replacement from the supervoxel and the supervoxel will be classified as the maximum voting of the prediction of all the sampled voxels. This is, for $\gamma = 1$ a random voxel will be sample from a supervoxel and the supervoxel will be classified according to that prediction. Opposite, with $\gamma = maxsize$ every voxel in the volume will be predicted, and the supervoxels will be classified as the average prediction for all the voxels within it. It can be noted that the parameter γ then controls the efficiency of the classification, from very quick by classifying few voxels to very slow by classifying all the voxels, aswell as the robustness of the solution proportional to the number of samples selected. γ can also be generalized to a per-supervoxel basis, by representing it as a fraction $0 \leq \gamma_s \leq 1$ of the size of the supervoxel s, this is, instead of picking a fixed number of voxels per each supervoxel, a number of voxels will be picked (e.g. 1% or 10%) proportional to the size of each supervoxel. This will allow to sample more voxels for large regions while fewer voxels for small regions.

6.3.4 Experiments and Results

In this section a experiment is performed to show the advantages of using Extremely Contextual Forests against other (Extremely) Random Forest (Geurts et al., 2006) models for the classification of sub-cellular organelles. To that end, we selected small region of interests from the public FIB-SEM dataset used in section 6.2. These regions of interest are sub-volumes of sizes $50 \times 300 \times 300$ and $50 \times 500 \times 500$, and for all of them the following voxel level features were extracted:

- Gaussian Filter and Total Variation denoising (Chambolle, 2004; Goldstein and Osher, 2009) to remove data noise while preserving volume boundaries.

- Laplacian of Gaussian to highlight blob-like structures, such as mitochondria in these images. These were extracted at two different scales, $\sigma = 2$ and $\sigma = 5$ for different size regions.

- Hessian eigenvalues (similar to (Frangi et al., 1998)) to highlight tubular structures (i.e. membranes) at different scales ($\sigma = 1$ and $\sigma = 3$) for membranes of different thickness.

This yielded a 7-dimensional descriptor (including the raw data) for each voxel in the dataset. Although more features could be extracted, this section aims to evaluate the accuracy of different Random Forest models with the same feature set, and having in mind that these features are extracted in 3D for the whole volume, we found a feature vector of 7 discriminative features to be enough for practical purposes, moreover since the input annotations are minimal and adding additional feature dimensions to a small dataset would just make the problem harder. Then, four different classification models are considered:

- **Voxel-RF**: Standard Random Forest trained on input **voxels** and used to predict all the voxels of the region of interest.

- **SV-RF**: Standard Random Forest trained on **supervoxels** annotated by the user scribbles, of which descriptors (as in section 6.1) were extracted from SLIC supervoxels (of approximate size $10 \times 10 \times 10$). Then used to predict the rest of the supervoxels of the volume.

- **SV-ECF**: Extremely Contextual Forest (ECF) trained on the annotated **voxels** and used to predict **supervoxels** by setting $\gamma_s = 0.05$, that is, 5% of the voxels inside each of the supervoxels are randomly sampled and the ECF will use them to average the prediction of the supervoxel.

- **MS-ECF**: A Multi-Scale ECF, where the ECF will initially be trained on **voxels** and used to predict **supervoxels** by setting $\gamma_s = 0.05$ as before. However, this time a refinement step will follow up, first identifying the supervoxels with highest uncertainty in their classification (by measuring the entropy of the probabilistic classification) and then classifying all the voxels of that supervoxel individually. This classifier can be seen as a multi-scale classifier that will start in the higher level scale (in this case, supervoxels) and fall back to more coarse segmentations for areas with high uncertainty (usually boundaries of membranes and organelles), allowing for a more accurate pixel level segmentation in those areas.

For our ECF models a maximum radius distance of 5 and displacement of 10 is selected. This allows to sample upt to 11^3 patches from a distance up to 10 voxels away to use as contextual information. These values were empirically chosen as in the dataset the object of interest were mainly small organelles and membranes, and thus selecting a larger radius or displacements would've introduced too many background pixels into the same ROI. Figure 6.16, Figure 6.17 and Figure 6.18 show qualitateive results of our Extremely Contextual Random Forests. It can be seen that both the ECF based on supervoxels or the one using voxels to further improve the accuracy give much better results than standard forest models. This is mainly due to the use of contextual information sampled from a distance, which makes the classifier to give predictions without spurious pixel level noise. The contextual prior presented here is similar to the one in previous MRF models, but allows for the segmentation at either the voxel, supervoxel or multi-scale level, allowing for more accurate pixel-level results.

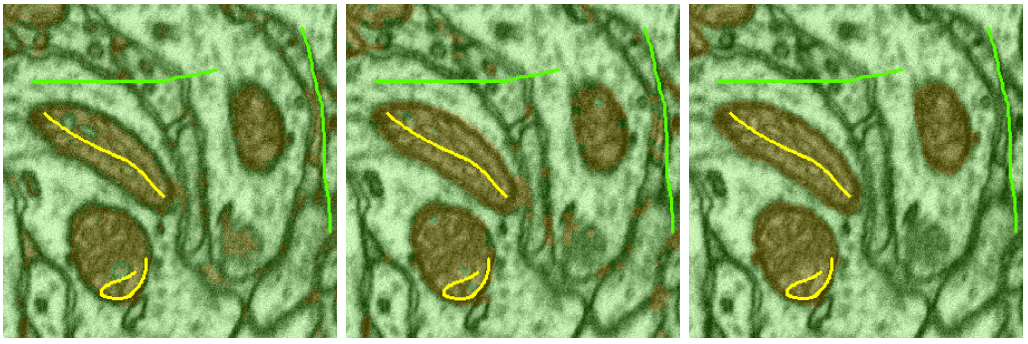


Figure 6.16: Example of segmentation of mitochondria vs background. From left to right: Voxel-RF, SV-RF, SV-ECF. It can be seen that the voxel-based RF has too many spurious predictions along the volume. By using supervoxels some of that noise is completely removed, but other spurious predictions remain since the few scribbles don't select enough supervoxels for training (just a few tens of supervoxels) with enough discrimination. Last, by using all the data available from the voxels to train an ECF and then predict supervoxels almost perfect segmentation results can be obtained.

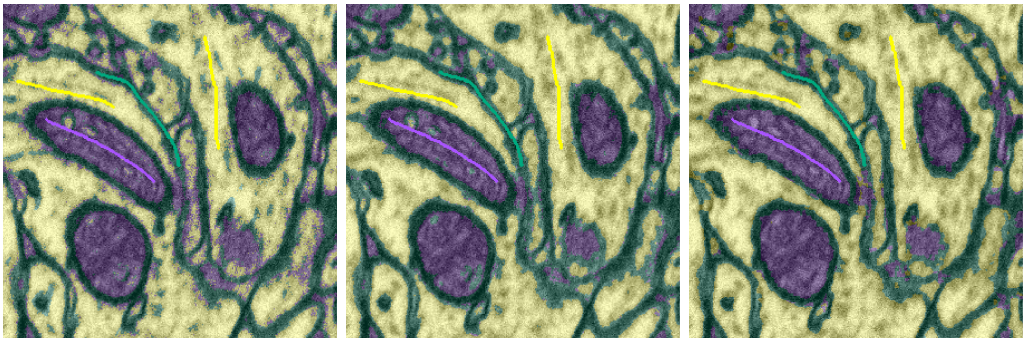


Figure 6.17: Segmentation of background (yellow), mitochondria (purple) and membranes (green) from minimal user annotations. From left to right: Voxel-RF, SV-ECF and MS-ECF. Predicting voxels adds too much sparse noise (purple and green random noise and yellow within mitochondria) that is partly fixed when supervoxels are used for predictions, as they average local noise and yield a more homogeneous results (still with incorrect segmentations, particularly with membranes). By refining the uncertain supervoxels of SV-ECF at the voxel level, MS-ECF is able to remove local noise further and make membranes more clear and accurate.

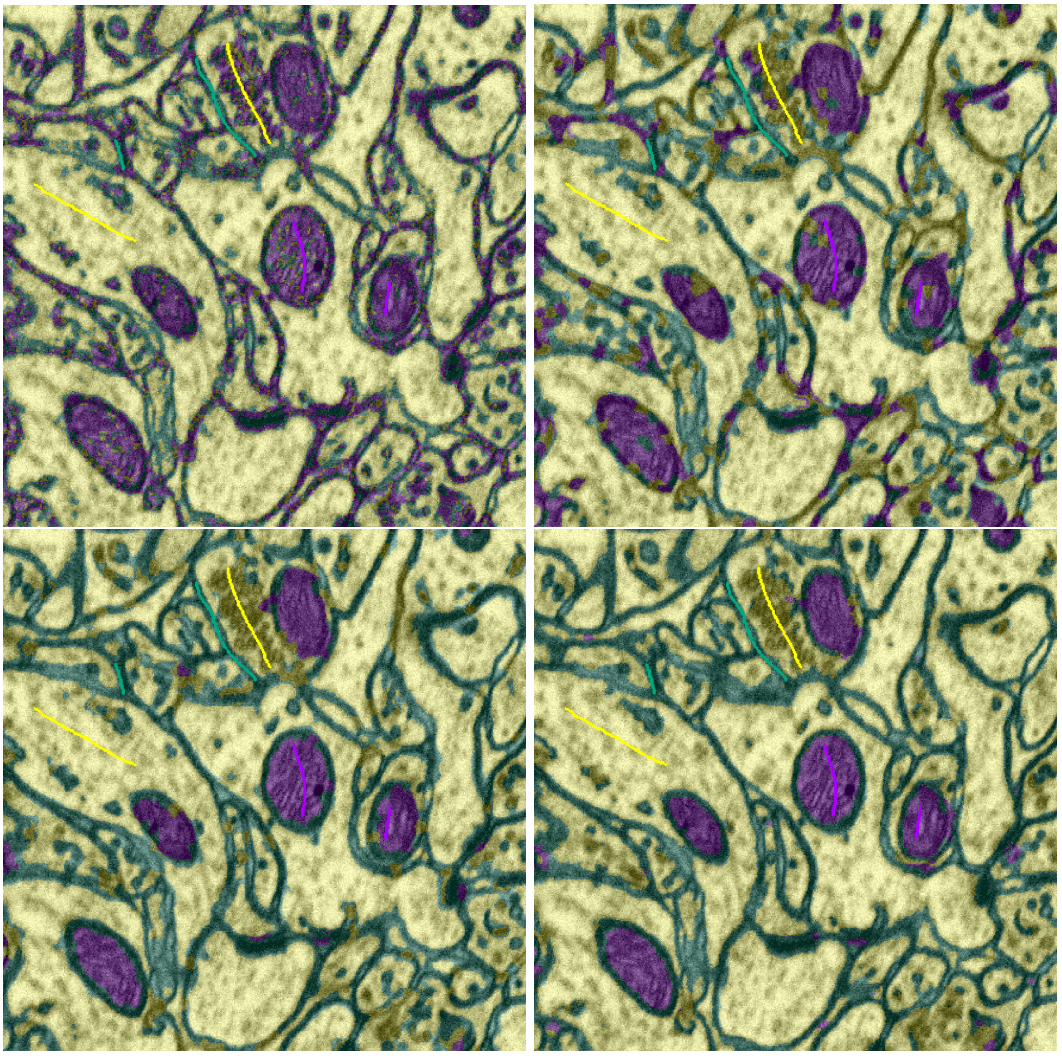


Figure 6.18: Segmentation of background (yellow), mitochondria (purple) and membranes (green) from minimal user annotations. Top row shows Pixel-RF and SV-RF while bottom row shows SV-ECF and MS-ECF respectively. It can be seen the increase in quality, from random voxel noise at Voxel-RF, to not enough training data to predict appropriate labels in SV-RF. However, when voxels are used for training and supervoxels for prediction, SV-ECF is able to obtain more meaningful results, that could be further improved by refining uncertain supervoxels at the voxel level with MS-ECF.

6.4 Discussion

In this chapter three different options or pipelines have been presented to address the problem of interactive segmentation of biological volumes. All the algorithms are designed to inherently deal with large volumetric datasets and arbitrary number of classes, making every pipeline inherently multilabel. User annotation is the key to segmenting a volume that has no prior ground truth available. To exploit it, the first pipeline guides the user in a top-down hierarchical segmentation task, making the segmentation process abstract from large areas to object instances as the user goes from quick to concrete annotations. The second approach first calculates a reduced subset of sub-volumes (or regions of interests) and feeds them to the user one by one, with a smart active learning procedure to maximize the output of the classifier with minimal user annotations. Last, a novel classifier was introduced, extending the work of (Montillo et al., 2011) to multi-channel features and adding higher order moments through integral images, as well as in conjunction with supervoxels. Any of those pipelines could be uses alone or in conjunction to provide fast annotations of biological volumes.

7 SuRVoS: Super-Region Volume Segmentation Workbench

Over the last two chapters the challenges of segmenting biological volumes have been discussed. It has been noted that one of the main challenges of sub-cellular organelle segmentation is the lack of training data, as it is the main barrier that prevents this field from adopting fully automatic approaches. The lack of annotations for these kind of datasets poses such an impediment for them to be properly analysed, that world level leading organizations (such as The Crick Institute, Diamond Light Source or the Structural Biology Department at Max Plank Institute of Biophysics) are working to obtain long term grants to create a database of segmentations of different sub-cellular datasets coming from several imaging modalities.

It should be clear by now that easy ways of annotating datasets is of particular interest in the structural biology field. To this end **SuRVoS**, a toolbox created with the sole aim of assisting biologist, specialist or any other arbitrary user without any prior knowledge to segment large volumetric datasets will be introduced in this chapter.

SuRVoS stands for **Su**per-**R**egion **Vo**lume **S**egmentation, and is a workbench built upon the knowledge and grounds of the previous chapters. It is a fully self-contained workbench that integrates different tools and pipelines within the same user interface. The goal of SuRVoS is to assist a user to visualize, analyse and ultimately segment 3D volumetric datasets. With this, we aim to speed up the segmentation time of biological volumes in order to create a database of segmentations that could then be used to apply automatic approaches that require more complex models.

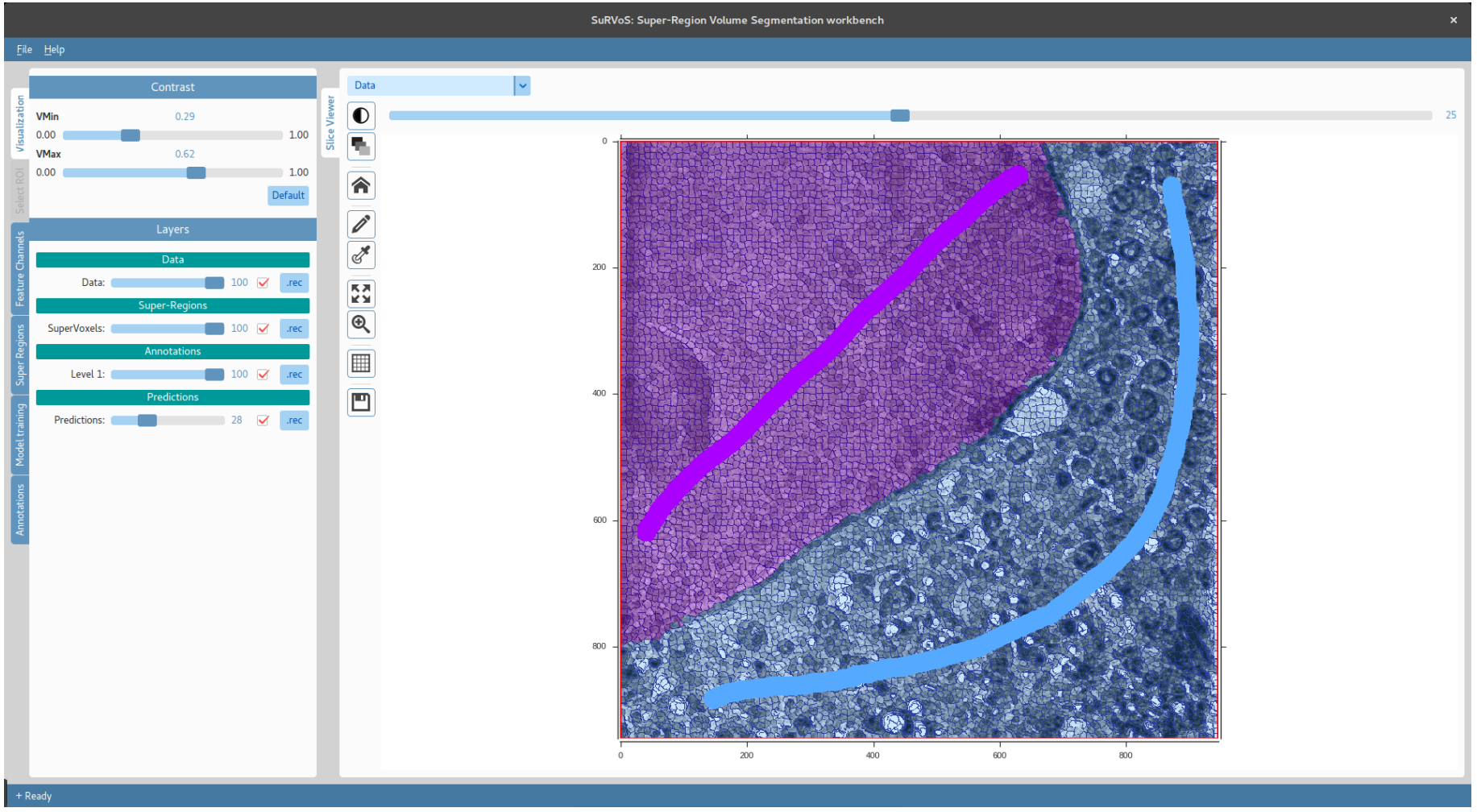


Figure 7.1: SuRVoS Workbench interface. It comprises a left side-panel with all the plugins available, a central viewer and a set of tools. All the plugins within the tool output Channels that are visualizable in the viewer in the form of layers (similar to e.g. Photoshop). Layers can be hidden/disabled and their opacity can be controlled. The screenshot shows an example of two user scribbles (thick lines with no transparency) and the predictions of an MRF model trained on just those two annotations (purple and blue regions with transparency) with supervoxel boundaries (dark blue boundaries) overlaid on it. Left panel shows available Channels for visualization and the Contrast tool controlling the contrast of the data.

SuRVoS (screenshot in Figure 7.1) comprises a standard slice viewer that can be used to navigate through the volume. It allows a user to load arbitrary volumetric datasets and process them with denoising and other feature enhancement techniques (discussed in Appendix B). Region representation using Supervoxels and Megavoxels (First and second order super-regions) generate representative regions that adhere to volume's boundaries without loss of detail, allowing the user to annotate hundreds of voxels through the volume by clicking in a single supervoxel or megavoxel. Machine learning classifiers are then available to learn from the user annotations and extend the knowledge to the rest of the volume. Some of the novelties and key differences of this workbench compared to other state of the art annotation tools, such as ilastik (Sommer et al., 2011), Fiji (Schindelin et al., 2012), Avizo (Westenberger, 2008), Icy (De Chaumont et al., 2012) or Chimera (Pettersen et al., 2004) are listed below:

Key differences from the user perspective

- It incorporates multiple tools in a single simple and intuitive user interface that talk to each other and work within the same framework.
- Every step (preprocessing, feature extraction, learning, annotation...) and action of the toolbox generates a **Channel** as an output. A channel is a 3D volume (or 4D in the case of some n-d features) containing the result of the operation.
- It supports visualization of EVERY Channel computed for visual assessment (whether it comes from a preprocessing, feature extraction, supervoxel creation or classification step). This gives the user a better understanding of what they are really doing and how their action is affecting the segmentation result.
- Introduces Hierarchical Segmentation Layers (section 6.1) to allow the user to hierarchically split manual and semi-manual segmentation. The user can create an arbitrary number of layers with their corresponding arbitrary number of labels, allowing for every single voxel to have multiple annotation labels (one per annotation layer). Classifiers can then be applied to a single layer or even to an specific region.
- It integrates Supervoxels and Megavoxels (1st and 2nd order super-regions as in chapter 3 and chapter 4) to ease the manual segmentation. Hundreds to thousand of voxels can be annotated with a single click. Supervoxels can also be used for classification, speeding the task several order of magnitudes.

Key Software differences

- SuRVoS creates Workspaces, which are folders on disk where all the data will be stored. The input and output for every operation within the workbench is a Channel. Every step done in SuRVoS is instantly saved to disk, and thus, is safe and resistant errors or crashes, none of the intermediate steps, data or annotations will be lost.
- A Channel is stored on disk in the ultra fast HDF5 (The HDF Group) format. HDF5 is a fast on-disk storage that allows for low memory requirements as only the data required is loaded into memory on-the-fly. It is extremely fast and very efficient due to smart automatic caching and chunking.
- All the preprocessing and textural features along with Supervoxels are implemented with custom kernels on the GPU, allowing for very fast data processing.

Key Algorithmic differences

- It is the first workbench to integrate in all its components Supervoxels and Megavoxels. Supervoxels are extracted using GPU code, which allows for supervoxel representation of large volumes in a few seconds (about 10 seconds for a $1000 \times 1000 \times 300$ dataset).
- Similarly, classification can be applied to the supervoxel graph (as in section 6.1 and section 6.2) and can be limited to a region or a parent label (from the segmentation hierarchy). This speeds up the classification task several orders of magnitude allowing for classification of whole $1000 \times 1000 \times 300$ datasets in about a second.
- Last, SuRVoS is the first tool to integrate Markov Random Field (MRF) segmentation models (Maximum A Posteriori inference) for large 3D datasets (section 6.1 and section 6.2). The computational complexity for the inference of standard MRF models for segmentations is too prohibitive for large 3D volumes, but applying them to the supervoxel graph instead of the 3D voxel grid reduces dramatically its computational time, allowing for MRF inference of large 3D volumes in about 3 to 5 seconds (with pixels it can take few hours).

7.1 Overview of SuRVoS Workbench

The SuRVoS Workbench combines the human expert's knowledge with data representation, machine learning and active learning techniques. This workbench is designed to semi-automatically segment large biological volumes with the input of a user and to guide the user through the process (Figure 7.2). The basic framework of SuRVoS follows these steps:

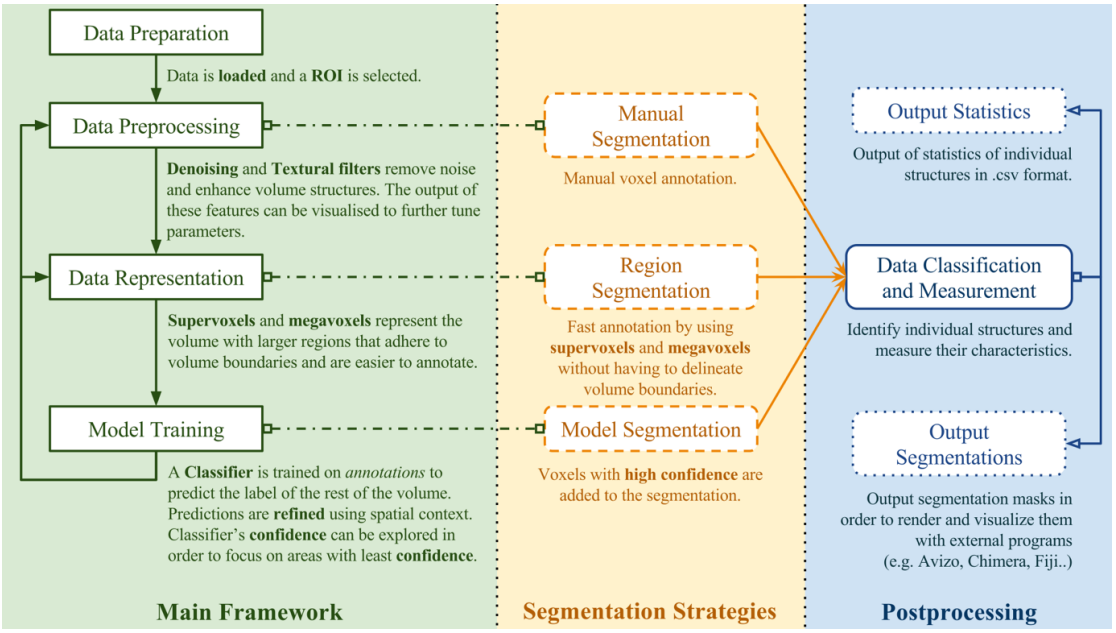


Figure 7.2: Overview of the SuRVoS framework. Each step has configurable parameters, yet sensible defaults that work well in many cases. The main user workflow (green box), which requires user intervention, consists of the data preparation, preprocessing, representation and model training stages. Segmentation strategies (yellow box) include fully manual, region-based and model-based segmentation. Post-processing (blue box) consists of data classification and measurement and output of data. Post-processing steps can be applied to segmentations created by any of the strategies. It is important to note that the user can return to any of the earlier steps at any time to further tune parameters for segmentation of different cellular aspects, or to switch to a different segmentation strategy.

1. Data Preparation: Data is first loaded into SuRVoS and a region of interest (RoI) is selected.

2. Data Preprocessing: Denoising filters are used to enhance volume information and remove noise that would reduce the segmentation accuracy. Textural filters[1] are used to enhance relevant features of the volume to better discriminate between different cellular components.

3. Data Representation: The volume is partitioned into more meaningful regions using hierarchical layers of super-regions: voxels, supervoxels (3D SLIC as described in subsection 6.1.1) and megavoxels by using HPCS (section 3.4) to further cluster supervoxels.

4. Model Training: The user defines an arbitrary number of labels and segments a large amount of data with a few clicks using the super-region hierarchy. With the user annotated data as input, a classifier is trained and applied to the whole volume

[1] Textural filters is used to refer to any kind of feature enhancing technique.

in real time. The user can then explore the confidence maps of the classifier, accept voxels with high certainty, and iterate through model training to further refine the results.

At any point the user can go back to any of the previous steps to improve the pre-processing or data representation steps in order to better highlight new target areas. This allows the user to try various parameter configurations and better segment challenging volume regions.

7.2 Data Preparation

SuRVoS currently supports the MRC file format (.mrc, .rec), HDF5 file format (.h5, .hdf5) and TIFF Stacks (.tif, .tiff), as those are the main file formats in biological microscopy. Once data is loaded into the tool, the contrast is automatically enhanced and can then be manually altered. The user interface of the workbench includes three separate areas, the Plugins, the Visualisation Pane and the Tool Column Figure 7.1. Parameters can be chosen and applied using the Plugins and assessed in the Visualisation Pane, while the Tool Column houses shortcuts to frequently used tools. Multiple RoIs can be created in the form of bounding cuboids ([zmin, zmax], [ymin, ymax] and [xmin, xmax]). All further actions of the tool can be limited to a designated RoI. This allows the user to select a RoI to test the denoising and textural features in a smaller area before expanding them to the whole volume, or, alternatively, it allows the user to constrain automatic segmentation to a specific RoI.

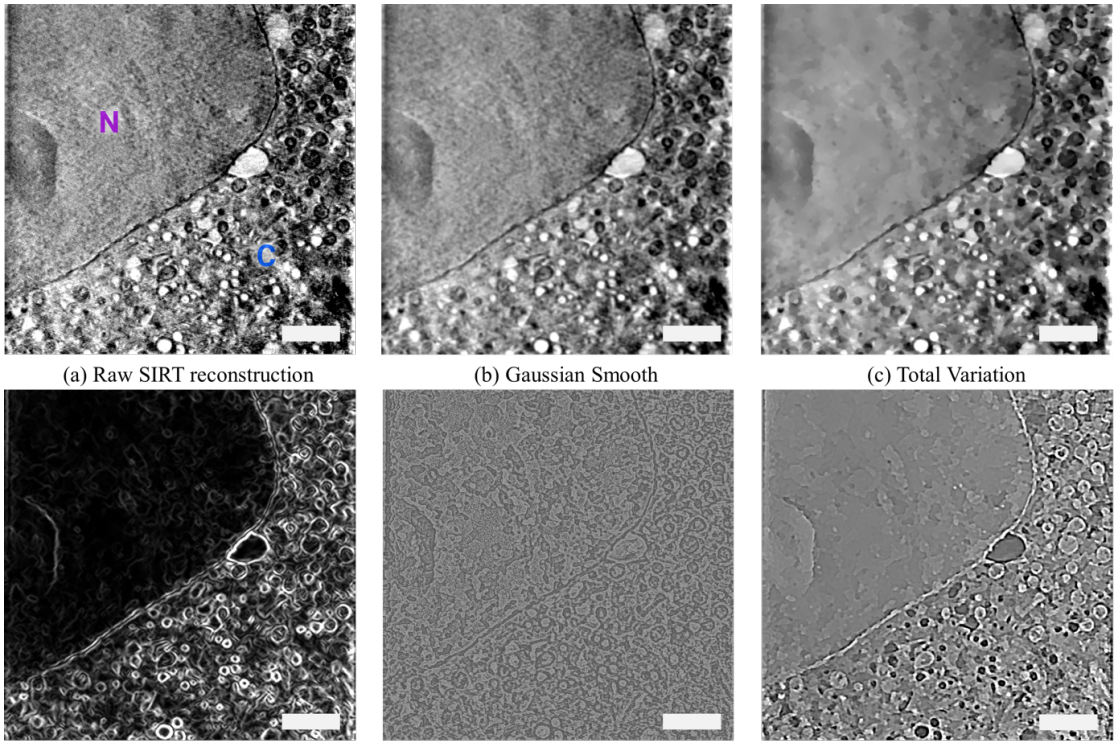
7.3 Data Preprocessing and Feature enhancement

The data pre-processing step combines both denoising and textural feature extraction methods to enhance aspects of the data and make future classification easier. Denoising includes standard Gaussian filters and Total Variation (TV) (Goldstein and Osher, 2009; Chambolle, 2004) denoising. While Gaussian filters usually produce over-smoothed results, TV methods preserve strong volume edges and tend to split the volume in piecewise smooth regions (Figure 7.3).

In addition to denoising options, Gaussian derivative filters, Difference of Gaussians (DoG) and Laplacian of Gaussians (LoG), rotation and scale invariant filters and the eigenvalues of the Hessian matrix and Structure tensor are also available as they are more robust feature extraction methods that help identify hidden characteristics of the

data. Additionally, Frangi filter (Frangi et al., 1998), its improved implementation using directional filters (Truc et al., 2009) and other Gabor filters (Jain and Farrokhnia, 1991) are also available. More details about features in Appendix B.

Every filter and feature extracted in this way is represented as a channel of the image, and the user can explore and visualise any channel of the 3D volume through the SuRVoS interface. This helps the user to understand what aspect of the data each of the features enhances, which parameters to choose for each aspect and whether a filter is useful for a particular dataset or not. For example, a TV filter (Figure 7.3.c) may be more appropriate for defining large regions such as the nucleus from the cytoplasm, while the Gradient Magnitude (Figure 7.3.d) or Laplacian of Gaussian (Figure 7.3.f) filters would be more appropriate for segmenting smaller, more nuanced regions such as organelles.



(a) Raw SIRT reconstruction (b) Gaussian Smooth (c) Total Variation

(d) Gradient Magnitude (e) Gaussian Local Normalization (f) Laplacian of Gaussian

Figure 7.3: Output of different denoising and textural features on a cryo-SXT dataset. a: A central slice of a SIRT reconstructed tomogram from a PC-12 cell showing a large region of nucleus (marked by N) and a crowded cytoplasm (marked by C), with many, various organelles present. b-c: The same central slice was used to illustrate two different denoising effects. Both Gaussian filter and Total Variation denoising reduce noise, however, the former can over-smooth the image while the latter gives piece-wise constant regions and preserves strong volume boundaries. d-f: Finally, the same central slice was used to illustrate different textural filters, which enhance various elements of the volume. Note: All filtering and denoising methods are applied in 3D. Scale bars are $2.5\mu m$.

7.4 Data Representation

Next, the volumetric data is represented as super regions within a 3-layer hierarchical structure. This structure is composed of voxels, supervoxels and megavoxels. Each of these layers is formed by grouping similar, nearby elements of the previous layer. That is, while voxels represent standard volume voxels, supervoxels are groups of adjacent voxels grouped together into a meaningful region that preserve strong volume boundaries (ie. boundaries between different biological features in the image). Similarly, megavoxels are groups of nearby supervoxels that have similar appearance. With this hierarchical partitioning, large areas of the volume belonging to the same object are represented by: a set of thousands of voxels, tens of supervoxels or a few megavoxels (Figure 7.4). This hierarchical structure has several advantages compared to the standard voxel grid:

1. Each of the layers of the hierarchy represents the same volume with many fewer elements than the previous layer, thus, reducing the complexity of annotating or segmenting the volume by several orders of magnitude.

2. Supervoxels and megavoxels are designed to have a strong boundary adherence and are therefore able to represent the same biological feature without significant loss of information.

3. The shape and size parameters of the supervoxels are user-definable in order to properly model volumes or areas with different physical characteristics.

SuRVoS implements a custom GPU version of SLIC supervoxels (Achanta et al., 2012) (section 6.1) which by default extracts supervoxels of size $10 \times 10 \times 10$ voxels from the volume. This means that the same volume is represented by small groups of around 1000 voxels without significant loss of information (Achanta et al., 2012), as homogeneous regions are represented by single supervoxels. Further larger regions can be extracted as megavoxels by further grouping supervoxels using HPCS (section 3.4). These megavoxels are also extracted in 3D and merge large, similar areas while preserving strong edges and small cellular structures as unique entities. Generally speaking, supervoxels are appropriate for segmenting smaller, more varied features, while megavoxels may be more useful for larger, more similar regions.

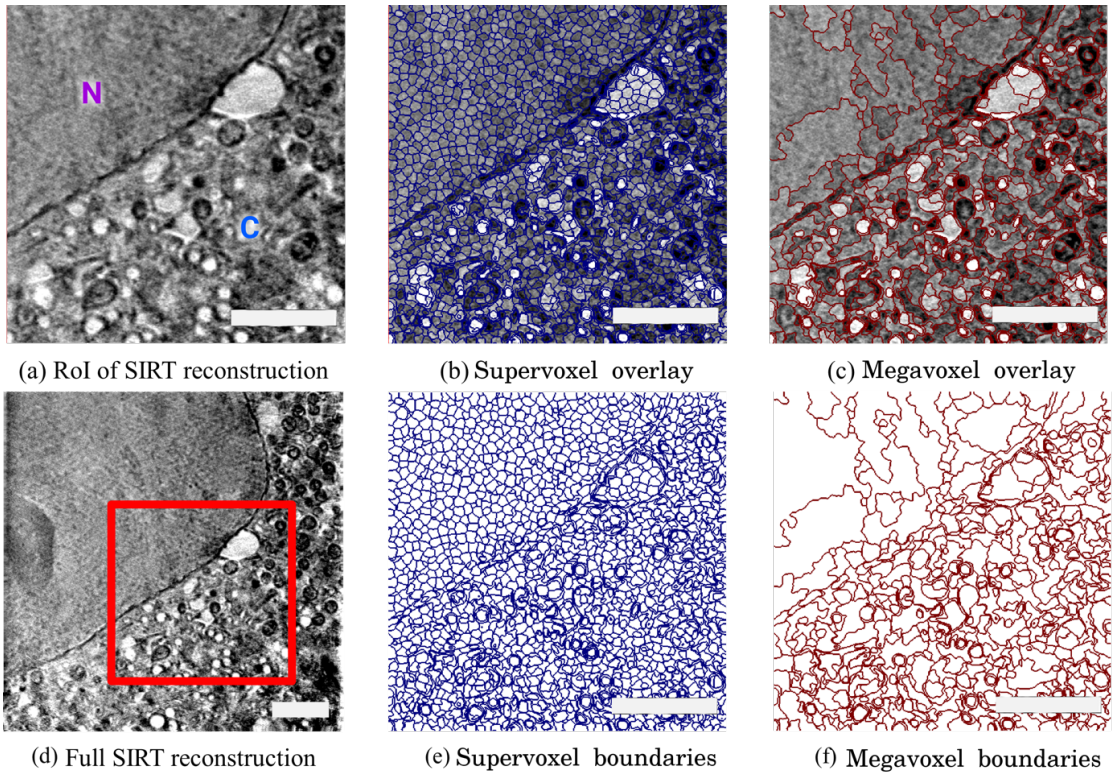


Figure 7.4: Super-region hierarchy example using a RoI of a cryo-SXT dataset (PC-12 mammalian cells). A single slice through a full-volume cryoSXT reconstruction with a smaller RoI marked (d, red box). This RoI was chosen to give a clearer view of the super-regions while still containing representative areas of both the nucleus and the cytoplasm. The RoI is shown again in (a) and with or without supervoxels (b, e) and megavoxels (c, f) overlaid. Supervoxels are formed by adjacent, similar voxels, while megavoxels group nearby similar supervoxels together. Supervoxels preserve strong image boundaries, in this case delineating the boundary between the nucleus and cytoplasm and many of the boundaries of the organelles within the cytoplasm (b, e). Similarly, megavoxels still preserve boundaries and better represent larger regions (c, f). Note: Both supervoxels and megavoxels are in 3D. Scale bars are 2.5μm.

7.5 Model Training

In order to learn from user annotations, voxels or supervoxels need to be described using feature vectors. Features extracted with SuRVoS (section 7.3) are built into discriminative descriptors that are suitable for training. These features form voxel descriptors D_p for pixel p and can be combined into supervoxel descriptors S_i by any of the following methods:

- Mean of all the feature channels for all the voxels inside each supervoxel.
- Histogram of quantised features (using k-means) from voxels inside the supervoxel.

- Textonized features (using Principal Component Analysis + k-means) as described in (Shotton et al., 2006).
- Using a SigmaSet (Hong et al., 2009) descriptor, which combines first and second order statistics (as in section 6.1).

Descriptors can be normalised or standardised as is common in machine learning to boost the performance of the subsequent classifiers. To enhance and add context to supervoxel descriptors, the descriptor of each supervoxel, $\boldsymbol{S}_i$ can be extended with the descriptors of the neighbouring supervoxels, denoted as $\mathcal{N}_i$, to form a more robust descriptor (e.g. a new descriptor is appended to the current supervoxel's descriptor containing the histogram or mean of the features from all the supervoxels that share a boundary with the current one):

$$\boldsymbol{S}_k = \left[\boldsymbol{S}_k\,,\ \frac{1}{m}\sum_{i\in\mathcal{N}_k}\boldsymbol{S}_i\right] \tag{7.1}$$

Once features are extracted, SuRVoS uses machine learning algorithms available through the scikit-learn library (Pedregosa et al., 2011) to classify each super-region into experimentally relevant groups. The available classifiers include: Random Forest (RF) (Breiman, 2001), Extremely Random Forest (Geurts et al., 2006), Gradient Boosting (Friedman, 2002), Support Vector Machine (Cortes and Vapnik, 1995) with different kernels (linear, RBF, etc.) and linear classifiers optimised with Stochastic Gradient Descent (such as linear SVM or Logistic Regression). An experienced user can try different combinations of descriptors and select the most appropriate classifier for each of them. By default, SuRVoS uses an ERF classifier as it requires no parameter tuning and deals well with many kinds of features and normalization.

A selected classifier can be trained on either voxels or supervoxels, the latter being orders of magnitude faster while preserving similar accuracy. The output of the classifier is the label prediction for each of the voxels (or supervoxels) independently. These predictions, however, contain a lot of noise and spurious labels as there is no spatial coherence. To address this, SuRVoS refines the predictions using a Markov Random Field (MRF) formulation, which takes into account neighbouring labels to make the predictions more spatially consistent. Given an undirected graph $G = (V, E)$, where V are the nodes and E is the edge set, and a finite set of labels (or classes) f, the task is to assign the optimal label f_i to each $i \in V$. The general form of a 2nd order MAP-MRF can be seeing as a label optimization problem that enforces unary ψ_p and pairwise $\psi_{p,q}$ constraints to the

set of nodes and edges,

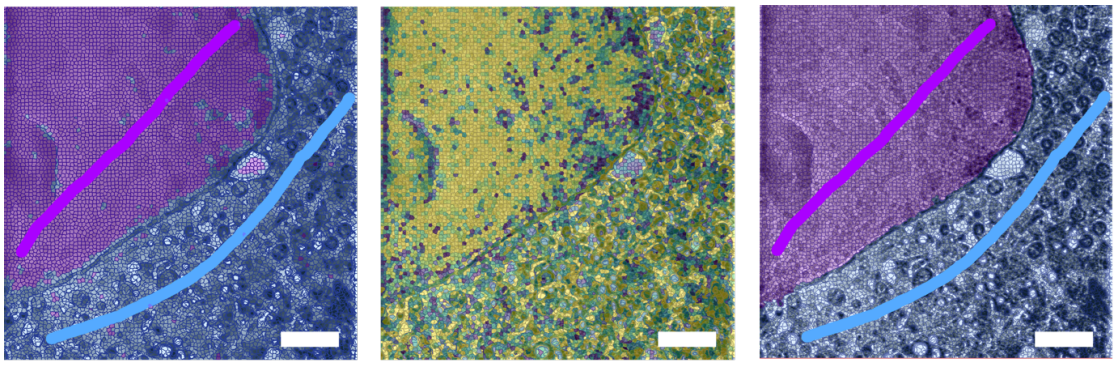
$$E(\boldsymbol{f}) = \sum_{p \in \mathcal{V}} \psi_p(f_p) + \lambda \sum_{(p,q) \in \mathcal{E}} w_{p,q} \cdot \psi_{p,q}(f_p, f_q), \tag{7.2}$$

where f_p is the label assigned to node p, and $w_{p,q}$ is a similarity weight between nodes p and q (descriptors S_p and S_q). Minimising Equation 7.2 yields the optimal labelling $\boldsymbol{f}^*$ for the graph G. For the refinement of the segmentation of large volumes, a graph G is extracted where the nodes are voxels (or supervoxels) and edges connect nearby nodes (in a 6/18/26 neighbourhood for voxels in 3D or supervoxels that are touching, i.e. share boundary voxels). The unary potential is set to the negative log likelihood of the probabilistic output of any of the above classifiers) and the pairwise potentials set as a weighted Potts model $[f_p = f_q]$ where $[\cdot]$ evaluates to 1 if the inside is true and 0 otherwise. The similarity weight between voxels or supervoxels is calculated as a difference between the descriptors of nodes q and p weighted by the amount of boundary voxels they share (constant 1 for voxels, and truncated boundariness for supervoxels). This encourages nearby similar nodes to have the same label, smoothing the classifier's result while still preserving the volume's strong boundaries. This is exploiting the property that volume features are usually consistent within a biological structure. Inference of the MAP-MRF models is performed by means of the FastPD (Komodakis and Tziritas, 2007; Komodakis et al., 2008) or QPBO (Rother et al., 2007) libraries. More details about the setup or computation of the above potentials can be found in the previous chapter, particularly on section 6.1 and section 6.2.

In practice, this approach allows the segmentation of biological volumes with a reduced level of user interaction. Using the default parameters of the refined classifiers, allowed, in one of the examples presented here, for the segmentation of areas corresponding to the nucleus and cytoplasm with just two brief user annotations Figure 7.5. While this figure is a two-dimensional visualisation, model training using these two user annotations is sufficient to segment the central 100 slices of a $946 \times 946 \times 350$ cryo-SXT volume. By using supervoxels, the classification and refinement steps take just 1-5 seconds to give predictions for 100 slices of the volume.

The classifier's confidence can be inspected in order to assess the quality of the classification. Confidence levels can be used as a guidance for further annotations, as annotating uncertain areas (darker areas in Figure 7.5.b) would help the classifier to better understand the data and obtain better performance. Additionally, SuRVoS contains a Confidence Acceptance Tool, which allows the user to save predictions based on the classifiers confidence level. That is, the confidence of the predictions (extracted as the

entropy of the labelling) can be thresholded to save only very confident regions (often accurate) regions and loop the training process with more training data. Iterative training and annotation will eventually yield good segmentations.



(a) Random Forest predictions (b) Random Forest Confidence (c) MRF-based refinement

Figure 7.5: Classification of a representative cryo-SXT volume of a PC-12 mammalian cell into nucleus and cytoplasmic regions. Each image is a central slice of the SIRT-reconstructed volume with supervoxels overlaid. User annotations are shown as elongated lines with no opacity (a,c); purple to denote nucleus and blue for the cytoplasm. (a) A Random Forest (RF) classifier is trained on the user annotations to learn to discriminate between the two regions. (b) The confidence of the RF classifier (yellow for higher and dark blue for lower) can be used to assess the model training. (c) The RF result is refined using a MRF formulation. It is important to note that both classification and refinement are propagated in 3D, and the user annotations shown here are sufficient to accurately segment the central 100 slices of this volume. Scale bars are $2.5\mu m$.

7.6 User Workflows

Applying the user's knowledge to quickly generate training data (in the form of annotations) is a key feature of SuRVoS. The user can annotate volume slices at any of the super-region levels: voxels, supervoxels or megavoxels. Thus, with a single manual annotation, all the voxels, supervoxels or megavoxels that the pen tool passes through are assigned to the selected label. This means the user can annotate vast regions of voxels with a minimal amount of effort. Labels can be created as required to correspond to image content and assigned a custom colour and name.

To make the annotation task more intuitive, SuRVoS uses a Segmentation Label Hierarchy. Segmentation labels can be created as needed and newer labels can be placed within parent labels to intelligently restrict model training and the segmentation space. For example, it may be useful to first segment large regions from each other (in one of our examples, nucleus from cytoplasm) before segmenting the nucleoli within the nucleus and the organelles within the cytoplasm. By exploiting this contextual information,

which is known to improve the performance of classifiers (Li et al., 2009; Tu and Bai, 2010), individual areas and organelles can be segmented separately in a more efficient manner. In most cases, a two-level label hierarchy is sufficient, separating large volume areas in the first level and assigning smaller objects of interest to the second layer. In SuRVoS, each segmentation label creates a separate annotation mask. This annotation mask can either be directly output, or can be used within SuRVoS to output only the data within the mask for downstream data visualization purposes.

SuRVoS provides the user with three main workflows with which to segment data: manual segmentation, region-based segmentation, and model-based segmentation. As with other available tools, such as Amira/Avizo (Westenberger, 2008), IMOD (Kremer et al., 1996), Fiji (Schindelin et al., 2012), Chimera (Pettersen et al., 2004), ilastik (Sommer et al., 2011) or Icy (De Chaumont et al., 2012), the user can manually annotate the voxels of the volume directly. This approach, at the expense of extensive manual labour, can enhance segmentation in difficult areas where alternative schemes do not achieve the desired level of segmentation accuracy. In the second segmentation approach, using super-regions (both supervoxels and megavoxels), different areas of the volume can be rapidly annotated without having to manually delineate region boundaries. As super-regions provide good boundary adherence, annotating cell areas is simplified and more time-efficient. Additionally, region annotations can make use of the Segmentation Label Hierarchy to limit a label to a previously defined area of the volume, preventing segmentations from exploding out of an area of interest. The third approach, using descriptors and annotations from previous steps (both manual and region-based annotations), machine learning algorithms can be used to extend annotations to the rest of the volume. This semi-automatic segmentation is aided by a confidence map and can be used to iteratively add to segmentations to train the descriptor. As before, training and prediction of segmentations can be limited to a certain area of the volume by using the Segmentation Label Hierarchy.

7.7 Data Classification and Measurements

Once the user has segmented the desired structures from the volume, SuRVoS provides two main tools for measurement and analysis. The Label Splitter tool allows the user to classify groups or individual objects within the segmentation. Using various measures extracted from each object (such as shape, size and average intensity within a selected feature channel) rules can be created to describe a type of object and segregate this type of object into a new class. Once the distinctions are made, results can be saved as new labels. Next, the Label Statistics tool can be used to visualize relationships and output

metrics for each object in a label. Plots for each of the metrics extracted for each label allow the user to visually examine the differences between them. Numerical metrics and plots can be exported as .CSV files or images for further analysis or figure preparation (see Figure 7.10).

7.8 Results

In this section segmentation results using SuRVoS will be shown for 5 different volumetric datasets from different imaging modalities:

PC-12 Neuronal-like mammalian cell line (Apostol et al., 2003) collected with cryo-SXT.

Trypanosoma Trypanosoma brucei procyclic cells (2913 strain) (Wirtz et al., 1999) Collected with cryo-SXT.

Goosegras : micro-CT of the fruiting body

Mammalian cell line collected using cryo-SXT.

Human Platelets collected with Cryo-Electron Tomography.

Each of the datasets above presents different challenges. Different imaging modalities, cell types and the scale of the objects of interest will help to assess the usability of SuRVoS for different use cases. The following sections will describe the process followed within SuRVoS to segment each of the datasets, with images showing results and a brief discussion over the pros and cons of using SuRVoS to segment them.

7.8.1 PC-12 Cell

The 3D data was imported into SuRVoS, where the contrast was adjusted and the volume was clipped to the relevant area. In the pre-processing tab, the data was scaled and appropriate supervoxels and megavoxels were identified (Figure 7.6). First, parameters for large, region-defining supervoxels and megavoxels were used in conjunction with model training to segment the nucleus, cytoplasm and extracellular area. Brief manual annotation of a single central slice using supervoxels followed by use of the Confidence Slider and the Area Acceptance tool allowed for the inclusion of voxels that made up the nucleus and cytoplasm through the central slab of the dataset. This process took less than five minutes of user time.

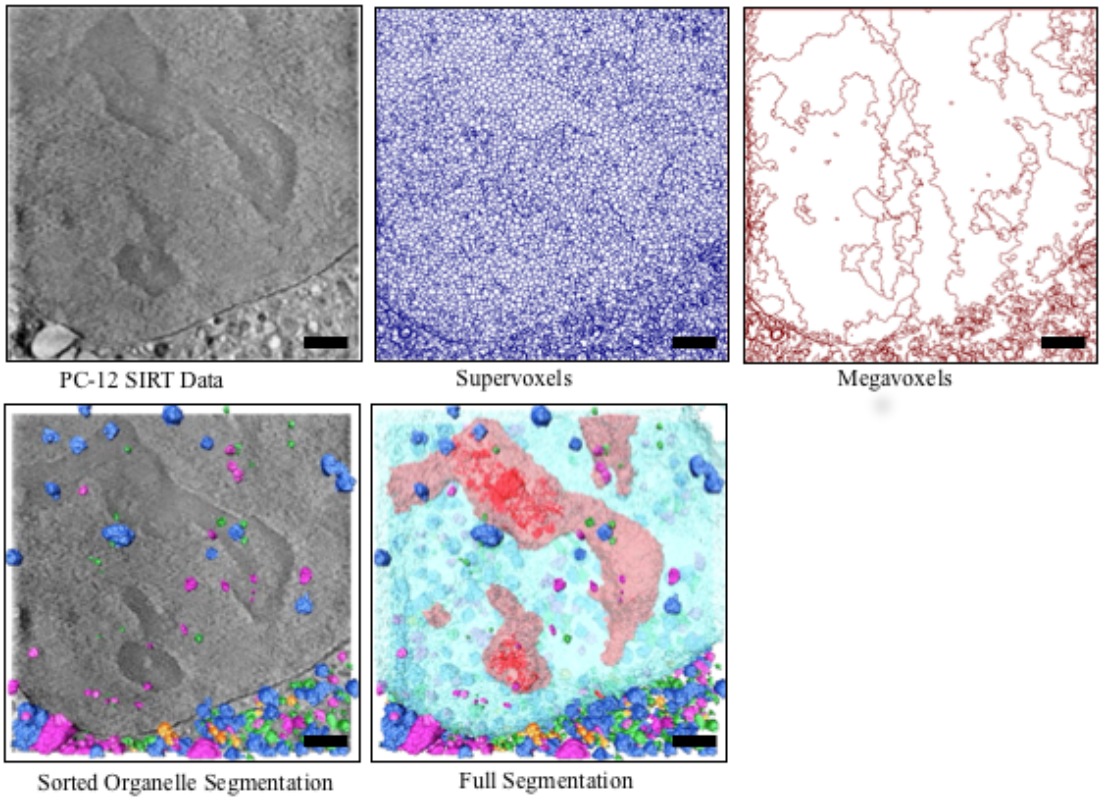


Figure 7.6: Qualitative results for the PC-12 cell segmentation. Top row shows central slice of the raw data (left) with the corresponding supervoxel (center) and megavoxel (right) extraction. Bottom row shows organelle (left) and full (right) segmentation of the volume respectively. In both segmentations, different classes of organelles, such as mitochondria, lipid droplets and empty vesicles, are represented in various colours. In addition, in the full segmentation, the nucleus is light blue, and nucleoli are red. Scale bars are $2.5\mu m$.

It proved more difficult to train SuRVoS to identify the boundaries between the top of the cell and the surrounding ice, and the bottom of the cell as it grew through the supporting holey carbon film. However, when iterating through the same strategy of quick manual annotation, model training, and acceptance of small areas of voxels with high certainty, these boundaries were still identified with minimal manual annotation. The entire region segmentation took approximately 1 hour for a $946 \times 946 \times 311$ slices tomogram, representing an approximately $15\mu m \times 15\mu m \times 5\mu m$ cellular volume. Next, the supervoxel parameters were tuned to best segment the nucleoli from within the nucleus. Additionally, the Segmentation Label Hierarchy feature was used to ensure only voxels within the nucleus were available for annotation as nucleoli. An iterative model training approach with supervoxels to label the nucleoli and not nucleoli was used.

Lastly, appropriate supervoxel parameters for organelle annotation were applied. A higher magnification was used to zoom in on a smaller area of the tomogram and all organelles within that area were annotated approximately every five slices through the

3D volume. Multiple organelle classes were used to ensure separation between near or touching organelles. This rough organelle annotation was repeated across the entire volume, followed by computationally filling holes in each label and smoothing of the segmentation edges using opening and closing operations.

Next, all organelle classes were used as inputs to the Label Splitter tool. Rules were created based on the average intensity, average variation, average standard deviation and organelle size and location in X/Y/Z. These rules were used to split the organelles into biologically relevant classes such as lipid droplets, empty vesicles and mitochondria. Quantitative information, such as number of organelles, voxel number size, and bounding box size, was output for each organelle class.

7.8.2 Trypanosoma brucei cell

The 3D data was imported into SuRVoS, the contrast adjusted and the data clipped to the area of interest. Supervoxels were chosen to allow annotation of multiple types of organelles, while megavoxels were chosen to allow for annotation of the lacey carbon support and cell body of the parasite (Figure 7.7).

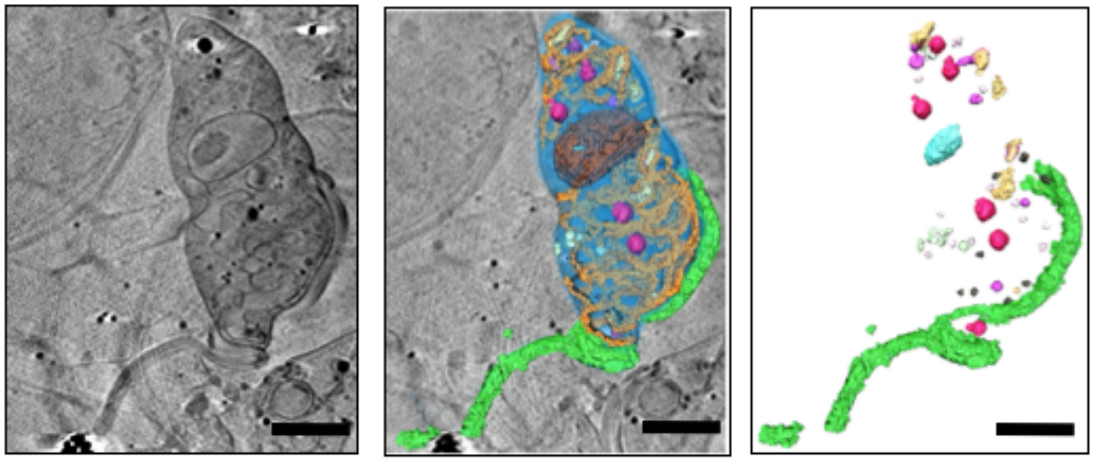


Figure 7.7: Qualitative results for the Trypanosoma brucei cell. Top row shows central slice of the raw data with the corresponding supervoxel and megavoxel extraction. Bottom row shows organelle and full segmentation of the volume respectively with the flagellum in green, the cell body in blue and organelles in various other colours. Scale bars are $2.5\mu m$.

As before, regions were first defined using megavoxels, however, in this case model training was not used. The lacey carbon support and cell body were segmented using megavoxels; while the parasites flagella, nucleus and other organelles were segmented using supervoxels. Computational hole filling and opening and/or closing operations were performed on each segmentation label. In order to segment this dataset, four sets

of supervoxel and megavoxel parameters were used and the total segmentation time was approximately eight hours. Classification of organelles was performed by eye, based on the general characteristics of the data. Quantitative information about each organelle group was output and used to begin to identify each organelle class.

7.8.3 Goosegrass' fruit body (micro-CT)

This dataset of the fruit of Galium aparine, or goosegrass, was collected using X-ray phase contrast tomography on the I13-2 Diamond-Manchester Imaging Beamline at Diamond Light Source, Chilton, Oxfordshire UK. The fresh sample was mounted in air onto a goniometer base on top of the rotation stage, at a sample-detector distance of 30mm. Exposure times were 0.10s with the pink beam spectrum which has a mean energy of around 22keV. Projections were collected through 180^o with a step size of 0.1^o. Tomographic reconstructions were performed using Savu (Wadeson and Basham, 2016) with the Paganin filter for propagation-based phase contrast images (Paganin et al., 2002) followed by filtered back projection reconstruction in the ASTRA toolkit (van Aarle et al., 2016; Palenstijn et al., 2011). This data was then downsized with $2 \times 2 \times 2$ binning to reduce the file size before being input into the SuRVoS Workbench.

First, the input data (Figure 7.8.A) was filtered and clamped (to remove upper and lower intensity values in the data) (Figure 7.8.B). In this way, the background and foreground were made more easily distinguishable and the texture of the internal structure of the fruit was accentuated. Next, supervoxels were built on top of the filtered dataset (Figure 7.8.C). To assess the quality of the supervoxels, they were displayed without the data to verify that pertinent details of the dataset were well represented by the supervoxels (Figure 7.8.D). Next, manual annotations using supervoxels were provided as training data on three slices of the volume (Figure 7.8.E, dark colours). This training data was sufficient to train the classifier to predict (light colours) the areas corresponding to the background (green), fruit bristle (red), seed material (purple), and surrounding flesh (blue). Morphological refinements were used to clean up the segmentations by filling holes, growing or shrinking as needed (Figure 7.8.F). The total time spent to determine appropriate parameters and segment this dataset was 2 hours.

7.8.4 Human Platelets (cryo-EM)

To demonstrate super-region segmentation using SuRVoS Workbench, a noisy and complex dataset was chosen (Wang et al., 2015). This dataset was collected using cryoET at the National Center for Macromolecular Imaging at Baylor College of Medicine, Houston, TX USA. Briefly, platelets were plunge frozen on glow discharged and gold

fiducial-treated holey carbon TEM grids using a Vitrobot Mark IV (FEI Corp). Tilt series were collected using a JEOL 2200 from 65 with a 2 increment. The tilt series was then reconstructed using weighted back projection in IMOD (Mastronarde, 1997).

After loading the data into SuRVoS (Figure 7.9.A), a region of interest was selected and an appropriate filter set was applied. In this case, a smoothing Gaussian filter followed by a total variation filter with the contrast clamped was used to accentuate the edges and textures of the data (Figure 7.9.B). Next, model training with minimal supervoxel-based user input was used to segment the platelet from the background ice and carbon. Then, semi-manual segmentation with megavoxels and supervoxels was used to segment the organelles. Lastly, the supervoxel source parameter was changed to a weaker denoising filter and supervoxel shape was made smaller in order to better preserve the microtubules boundaries for segmentation (Figure 7.9.C). For both the organelles and the microtubules, quick manual annotations were used every 5-10 slices to select the supervoxels that describe the feature of interest (Figure 7.9.D Figure 7.9.E). The total time spent to determine appropriate parameters and segment the region of interest presented was 5 hours. Given that it was the first time a EM dataset was segmented with SuRVoS, most of the time was used to identify features of interest and how to hierarchically stack them to create more discriminative features for later use to create appropriate supervoxels and improve the classification results.

7.8.5 Mammalian Cell-line (cryo-SXT)

To demonstrate label splitting using SuRVoS Workbench, a dataset with many, varied organelles was chosen. This dataset was collected using cryoSXT on beamline B24 at Diamond Light Source, Chilton, Oxfordshire UK. Briefly, HEK293 cells were grown on gold finder grids, appropriately sized gold fiducials were added and the grid was plunge frozen using a Leica EM-GP with back-sided blotting. Tilt series were then collected on a Zeiss UltraXRM-S220c microscope from $\pm 65^o$ with a 0.5^o increment. The tilt series was then reconstructed using weighted back projection in IMOD3335.

After loading the data into SuRVoS (Figure 7.10.A), a region of interest was selected and an appropriate total variation filter was used to enhance the boundaries of the organelles throughout the volume (Figure 7.10.B). Next, organelles were semi-manually segmented using megavoxels and supervoxels, and then refined with fill holes, closing and dilation to smooth edges (Figure 7.10.C). The total time to determine appropriate parameters and segment the region of interest presented was 4 hours. Once the segmentation was finalized, the Label Splitter was used to visualize each organelle as an object in the dataset (Figure 7.10.D) and various characteristics about each object in the data plot

(Figure 7.10.E). The Label Splitter interface is interactive, updating the colour associated with each new label class in both the visualisation and the data plot. This allows for the creation of various rules based on the characteristics inherent in the data that can be used to separate the objects into useful classes (Figure 7.10.F).

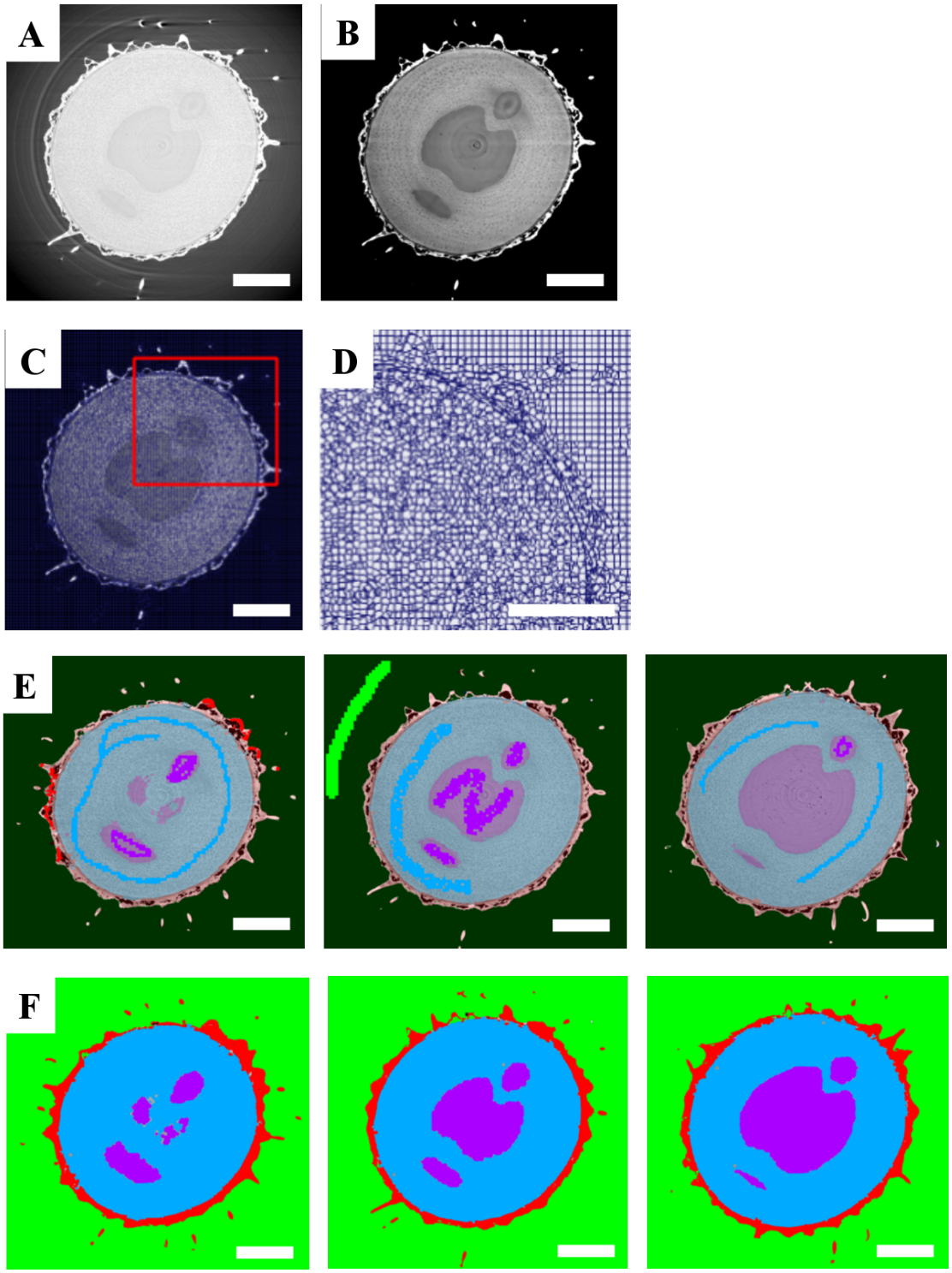


Figure 7.8: Processing of a microCT dataset using a model training segmentation strategy. (A) A single 2D slice of the raw data. (B) Applying a clamped total variation filter to the raw data enhanced the boundaries between the various aspects of the fruiting body. (C) Appropriate supervoxel parameters were chosen. (D) A region of interest (red box in C) is shown to demonstrate that the boundaries of the data are present in the supervoxels themselves. (E) Three slices of the volume with manual annotations of various areas of the dataset displayed in dark colours (green, red, blue and purple) and predictions after running model training displayed in the same light colours. (F) The same three slices with the final segmentation after the model training predictions have been accepted. Scale bars are 1*mm.*

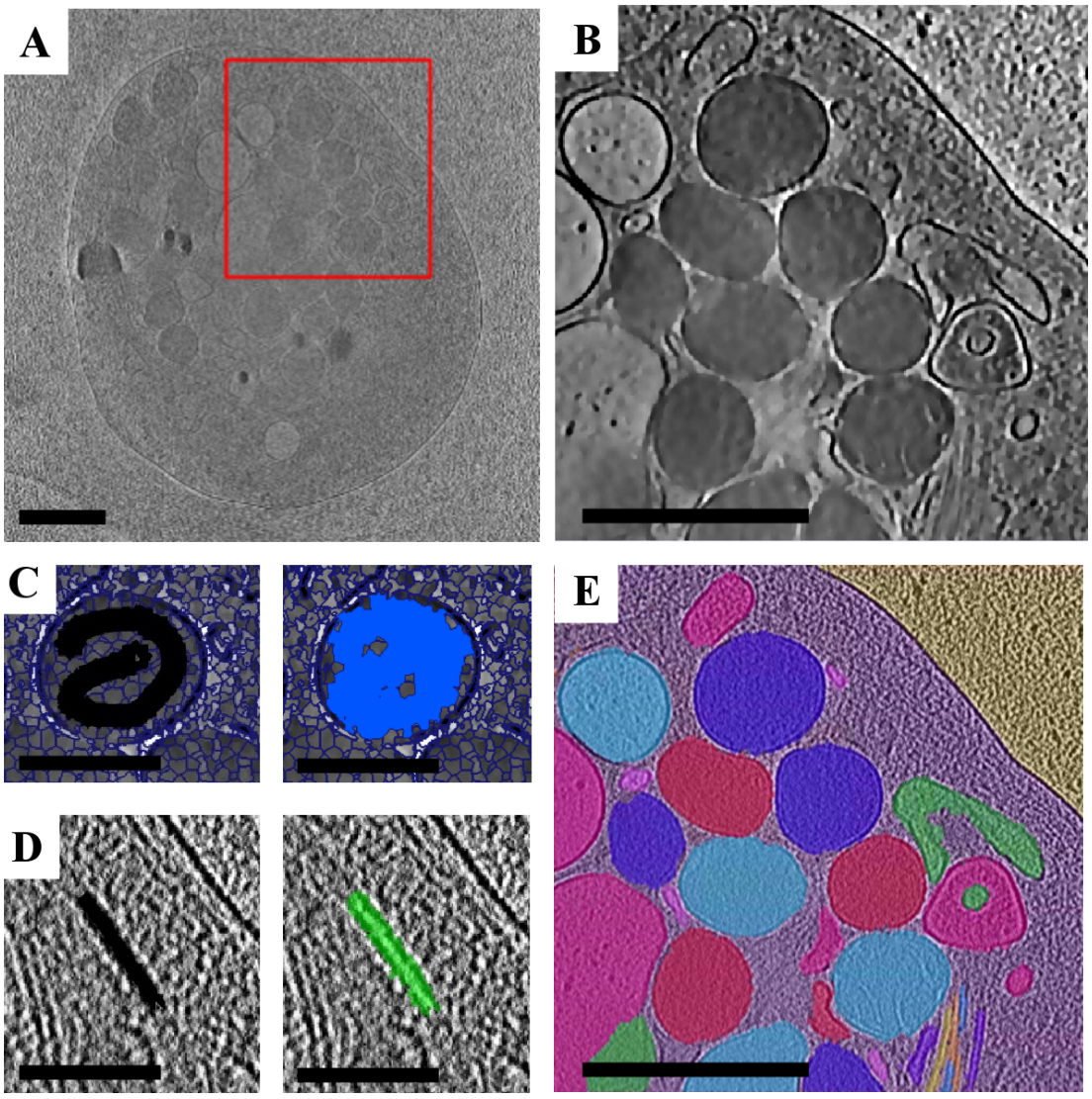


Figure 7.9: Processing of a cryoET dataset using a super-region segmentation strategy. (A) A single 2D slice of the raw data. (B) A region of interest (red box in A) with a layered filter set applied to accentuate the boundaries of the organelles. (C) Example of annotating an organelle using super-regions. A single organelle is shown with supervoxels overlaid with the manual user annotation displayed in black (left) and the supervoxels chosen with that annotation shown in blue (right). (D) Example of annotating a microtubule using super-regions. A single region of microtubule is shown with the manual user annotation displayed in black (left) and the supervoxels chosen with that annotation shown in green (right). (E) The final segmentation featuring the platelet segmented from the background using model training (see Table 1 for details), and various organelles and microtubules segmented using a super-region segmentation strategy. Colors do not indicate specific organelle types. Scale bars in A, B, and E are $1\mu m$ *and in C and D are* $0.5\mu m$.

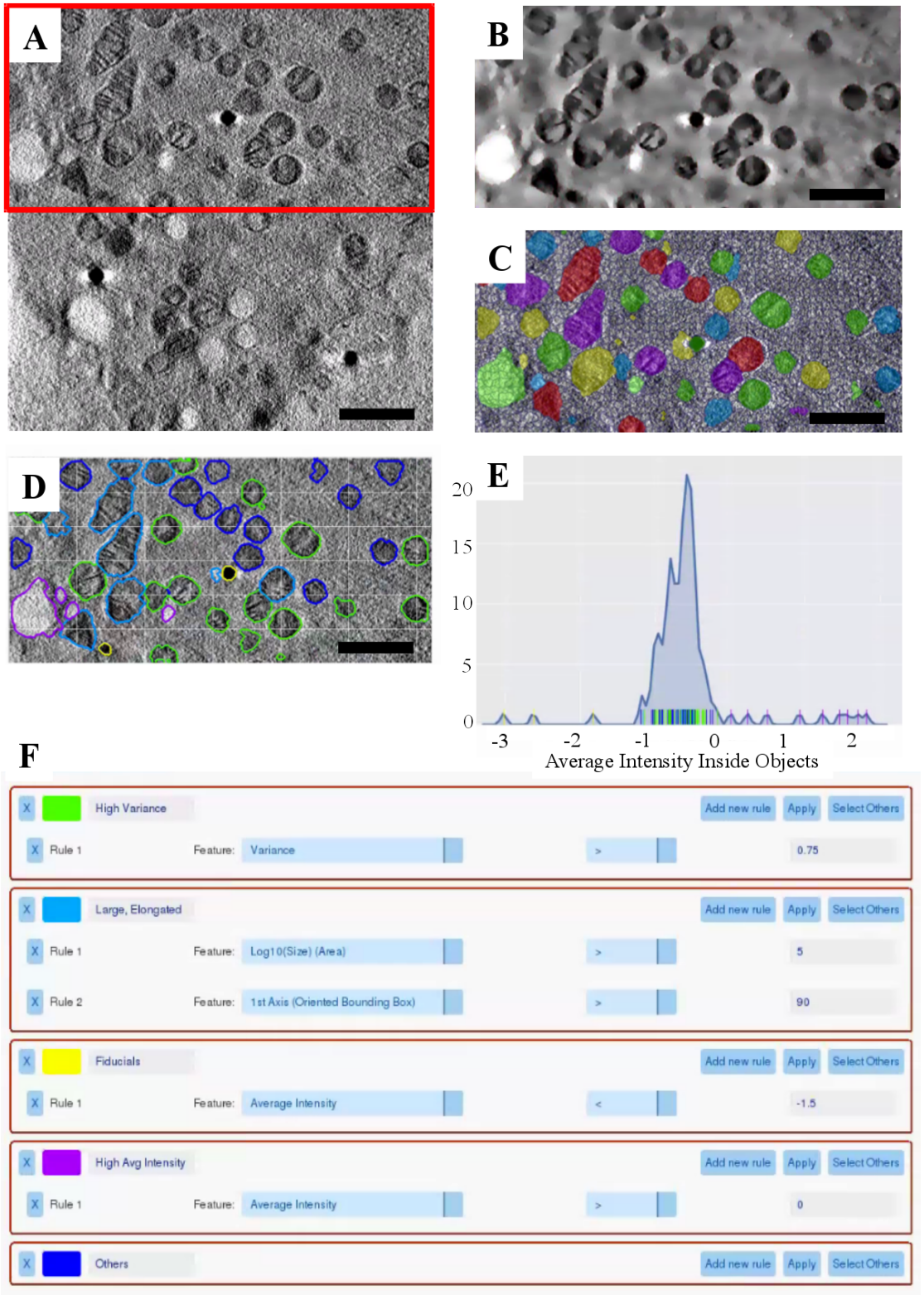


Figure 7.10: Analysis of a cryoSXT dataset using the label splitter tool. (A) A single 2D slice of the raw data. (B) A region of interest (red box in A) with a total variation filter applied to accentuate the organelles. (C) The final segmentation with supervoxels overlaid. (D) The visualisation portion of the label splitter with the organelles classified using the rules displayed in (F). (E) The plot portion of the label splitter displaying the average intensity inside each object, with the rules displayed in F applied. Each vertical line along the x-axis represents a single object and is colour-coded to match the class it has been assigned to. (F) Example classification rules to separate various objects based on their inherent properties. Scale bars are 1μm.

7.9 Discussion

We present SuRVoS, a new workbench comprising machine learning algorithms and computer vision volume features that uses the inherent features of the data to hierarchically build up regions that respect data boundaries. These regions are then used in conjunction with user interaction to segment 3D data. As features extracted from the data itself are used, SuRVoS is suitable for segmentation of any 3D data. Here, we show examples using soft x-ray cryotomography data, however SuRVoS has been used at a recent training workshop to segment data collected by cryo electron tomography, soft x-ray cryo tomography, focused ion beam scanning electron microscopy (FIB-SEM), and hard X-ray microtomography. Based on these experiences, we believe SuRVoS will be generally applicable to segment any volumetric dataset regardless of technique used to collect the data. Additionally, when regions are used to segment the data, the segmentation boundaries are not manually drawn by the user, meaning the resultant segmentation should be less subjective at the voxel level. Additionally, the use of rules based on characteristics of the data paves the way for less subjective identification of organelles within each dataset and between normalised datasets. Further studies would be necessary to fully assess subjectivity in segmentations using SuRVoS Workbench.

There are three ways to use SuRVoS: 1. fully manual voxel segmentation, 2. region-based segmentation (using super- or megavoxels), 3. model-based predictive segmentation. Each of these approaches can be useful depending on the dataset to be segmented and the intended use of the data. In the first approach, segmentation is completed manually without utilising many of the features of SuRVoS, but provides a hand-crafted, expertly derived segmentation. This is essentially equivalent to other available segmentation software. While results will be highly subjective and the process time-consuming (Hecksel et al., 2016b) the segmenter may be able to better identify smaller, finer features of interest. The second approach, using supervoxels based on features within the data to segment every few slices, is appropriate for smaller, less uniform features that are unsuitable for model training. The amount of manual segmentation for these features is still high, but because 3D supervoxels are used, meaning user segmentation is only necessary every few slices and without needing to delineate boundaries, the amount of manual segmentation required is still much less than for a fully manual segmentation. The third approach uses the entire suite of features within SuRVoS to quickly train the program and define the boundaries of regions. It can also be used to segment cellular aspects displaying similar characteristics such as similar organelles. This strategy iterates through quick, minimal, manual annotation, followed by model training and acceptance of small areas of voxels with high confidence. The number of voxels that are manually segmented is much smaller in this approach and, compared to a manual segmentation,

the time required to segment a volume into various regions is greatly reduced (by a factor of approximately 5). In order to use model training to segment organelles, multiple classes of organelle have to be used such that no organelle is touching another organelle in the same class. This prevents the joining together of two nearby organelles into one object, which would skew classification and statistical output during the label splitting process.

The differences between these three ways of using SuRVoS stem from the supervoxel or megavoxel parameters chosen and the features they are based on. The denoising and textural filters can be stacked and built up, creating filter sets that accentuate specific data characteristics. These filter sets can then be used to create targeted supervoxels or megavoxels for segmentation. Therefore, it is helpful to assess multiple filter, supervoxel and megavoxel options, as these can be changed at any point in the segmentation process. Within the same segmentation, specific parameters can be chosen to segment regions, or to ease segmentation of each specific feature or type of feature within the data. Parameter choice is a critical step in the segmentation process using SuRVoS. Choosing appropriate parameters throughout is necessary for ease and quality of segmentation. Each sample type will be different, and some, like the Trypanosoma brucei sample presented here, will benefit greatly from tailored feature, supervoxel or megavoxel parameters.

SuRVoS has been built with a Segmentation Label Hierarchy that allows the assignment of parent and child relationships for labels. Using this feature, segmentation and model training can be restricted to the region of the parent label. Since this hierarchical arrangement constrains the segmentation area, this can speed up computation and further segmentation steps. Template matching tools are not included in SuRVoS, however aspects of the Workbench can ease the use of these tools. For example, SuRVoS can be used to segment large regions to mask away areas where template matching is unnecessary.

The two full example segmentations presented here were completed by a biological and segmentation expert (PC-12 cell), and a biological and segmentation novice (T. brucei cell). In the case of the PC-12 cell, the segmentation volume was approximately $15\mu m$ by $15\mu m$ by $5\mu m$ (946 pixels by 946 pixels by 311 slices) and took approximately 35 hours to segment, classify and visualize fully, including all the time spent in computations, parameter tuning and trial and error. For the T. brucei cell, the segmentation volume was $11.6\mu m$ by $14.8\mu m$ by $2.2\mu m$ (728 pixels by 927 pixels by 135 slices) and took approximately eight hours to segment fully. Although the times might seem a bit high, it is to note that segmentations were performed by non-experts in machine learning or computer vision, and the timings include learning to use the tool and playing

with its options and parameters. Segmentation of these datasets would've taken few weeks for the biologists. The speed-up in segmentation time, in conjunction with the decreased subjectivity of the segmentation method means larger datasets can be more easily analysed in a more quantitative fashion.

Other three different datasets from different image types were presented here to showcase the different features of SuRVoS: semi-manual annotation, model training and organelle classification.

7.10 Conclusion

The SuRVoS workbench brings together machine learning models, computer vision techniques and human knowledge within a user interface to interactively segment large 3D volumes. The introduction of super-regions (supervoxels and megavoxels) reduces manual segmentation and removes manual delineation of boundaries, potentially decreasing subjectivity. Additionally, SuRVoS provides a wide variety of denoising and textural filters that can be tuned to each sample and used to train a machine learning model to extend the users manual segmentations to the rest of the volume. The resulting segmentations can be measured and classified within the tool and exported for further visualization. In addition to the above benefits, using the features introduced in SuRVoS, user segmentation time is reduced by 5x times compared to manual segmentation. Together, these features pave the way for better, more quantitative use of 3D volume datasets in biological sciences.

The workbench is available online and free to use at:

https://diamondlightsource.github.io/SuRVoS/

where other tutorials and explanation of the features within SuRVoS can be found. Additionally, a recently published visual experiment (Darrow et al., 2017) can be found at JoVE:

https://www.jove.com/video/56162/volume-segmentation-analysis-biological-materials-using-survos-super

Part III

Future Work

8 Conclusions and Future Work

This last chapter of the thesis summarizes the overall achievements of this work and describes the impact of this project in current biological imaging fields. Additionally, I would like to present new and exciting opportunities and ongoing future work that have arisen over the the last couple of months, while writing the thesis, to extend the work presented here and move a step further into easing the annotation and segmentation of biological volumes.

I would like to restate that the main purpose of this thesis, which is to advance the annotation collection procedure for structural biologists with the aim of easing the semi-automatic segmentation process, while reducing the subjectivity on the segmentation results and the time needed to obtain them.

I have stated several times throughout the thesis that the main issue holding automatic segmentation of biological volumes back is the lack of large amounts (sometimes even *reasonable* amounts) of annotated data publicly available. Though some areas such as connectomics, a sub-field of neurology that studies the map of the brain, have gained a lot of attention recently and multiple automatic methods have been introduced to the point that the problem would be considered solved in the next few years, other biological fields struggle to obtain the same attention, due the more challenging datasets and limited samples. To put the wide biological imaging area in context, the particular problem of segmenting connectomics could be related to the also very broad problem of segmentation of 2d images, in the form of human segmentation where the pictures were taken on a sunny day, with blue sky, from a distance of 50m and altitude of 20m at a 45 degree angle and with all the persons of the same ethnicity and constitution wearing white t-shirts and blue trousers. This is, it is a very particular sub-problem of structural biology, where the contrast in the volumes, the structures present there and

the morphology of the objects are not representative of the rest of the biological imaging areas.

With the work present in this chapters, the segmentation and annotation process for collection of 3D biological volumes has been greatly improved compared to other open alternatives previously available. To that end, my journey through this project has been focused in three main areas:

Understanding the nature of the data. This included understanding the source of the volumes from both the biological/physical (imaging techniques and microscopes used) and computational (volume reconstruction techniques) perspective, as well as characterizing the structures, noise and artefacts present there.

Understanding the image segmentation problem, and in particular, the segmentation of biological volumes, along with their characteristics, challenges and limitations. After realising the lack of annotations and the complexity of the datasets, I focused on understanding mid-level perceptual grouping techniques, in particular the superpixel problem. With the aim of extending the usability of the superpixels as a intermediate representation of large 3D volumes two new algorithms were introduced, with the ability to inherently extend to N-dimensional data and hierarchical grouping, forming hierarchical over-segmentation layers that I refer to as Super-Region Hierarchies.

Developing new algorithms, pipelines and workbenches for the segmentation of biological volumes. Although none of the techniques introduced in this thesis assume any prior knowledge or model of the underlying data and, thus, are generalizable to arbitrary ND volumes, their main focus has always been segmenting 3D biological volumes. As a result of this, two different pipelines were proposed based on the super-regions and a new tool, SuRVoS, was developed with the aim of making this research open and available for every one to use.

8.1 Overview of the Main Contributions

The exact list of contributions in the form of publications and open source software was detailed in section 1.2 and section 1.3. Here, however, I will focus in concrete instances and applications.

I reviewed the concept of hierarchical segmentation. This concept has arise a few times in the literature, in particular with hierarchical segmentations initially introduced with as a region merging or region splitting problem. I worked on the concept of a hierarchy

of over-segmentations, previously used in (Ladický et al., 2014), but extending it to a bidirectional hierarchy that I named a Super-Region Hierarchy (SRH).

Then two novel superpixel algorithms, HPCS and SMURFS, that outperform the state-of-the-art on superpixels, at the expenses of being slightly more computationally demanding, were presented. The algorithms are generalizable hierarchically to form Super-Region Hierarchies, but in particular, SMURFS uses the full potential of SRH to refine and improve the super-regions at a particular level by using the levels above and below. I defined conceptual merging and grouping operations for using all the information on the hierarchy to create better over-segmentations and described the limitations of this hierarchies. Then, SMURFS was extended to be able to refine individual levels of the hierarchy while maintaining its consistency. Additionally, a GPU version of SLIC (Achanta et al., 2012) in three dimensions was developed, that also takes into consideration the anisotropic data sampling. While SMURFS is shown to have much better accuracy and boundary adherence than SLIC, its computation is also more expensive (particularly when comparing with our GPU version, from seconds to minutes). We then recommend the usage of 3D GPU-SLIC for fast prototyping and near-real time interactive segmentation, where speed and user feedback matters, and the usage of 3D SMURFS where the quality matters and speed is not important, e.g. batch processing once enough data has been gathered or in smaller sub-volumes (such as the ones automatically extracted by Selective Labelling section 6.2).

Going from 2D images to 3D biological volumes, two segmentation pipelines were presented. The first one using Hierarchical Segmentation Layers (HSL, section 6.1) to split the problem of segmenting whole volumes into smaller and easier sub-problems. I showed that segmenting volumes hierarchically not only gives more accurate results, but it requires less training data and it is much more intuitive for the user, as it is closely linked to our ability to perceive the world. The second pipeline, Selective Labelling (section 6.2), is aimed to ease the task of segmenting large and complex volume for the human annotator. It is also known that segmenting a large 3D volume is not only tedious and time consuming, but also very difficult as the annotator does not always knows what to segment or where to start, and this results in huge variability in annotations, even when the same person annotates the same dataset a few months later (Hecksel et al., 2016b). To that end I introduced a completely unsupervised algorithm to extract sub-volumes from the data, and showed that not only makes the segmentation task more intuitive, but allows it to obtain nearly the same segmentation accuracy for the whole dataset with only 5% of the training data.

Last, in order to move the main goal of this thesis forward, a lot of time and effort was put into SuRVoS, a tool I developed with the advice and support of my supervisors and

Michele Darrow (post-doctoral structural biologist at Diamond Light Source) with the aim of extending the research presented here to the broad biological community. The tool was developed with robustness and wide usage on mind, and thus, a lot of effort was put into understanding the datasets and how to best manage them. SuRVoS is mainly written in Python and stores all the data and intermediate results in HDF5 datasets. This efficient data handling allows for working with larger-than-memory datasets of arbitrary 3D volumes. For the complex and computationally expensive algorithms SuRVoS uses external solvers (such as (Rother et al., 2007) for MRF models) and libraries ((Pedregosa et al., 2011) for learning algorithms) that interface C/C++ code, along with custom C++/CUDA code for fast feature and super-regions extraction. SuRVoS has been very successful and has been received with a lot of expectation from the biological community. We ran 3 workshops at Diamond and the Crick institute, and it is being used by many large scale labs and institutions around the world, such as Diamond Light Source (UK), CELLS (ALBA, Spain), Crick Institute (UK) and the Max Plank Institute of Biochemistry (Germany) to name a few.

8.2 Current State and Future Work

The problem of biological volume segmentation is far, very far from being solved. Having in mind that the goal of this thesis is making annotation and segmentation of whole volumes easier, it is clear that there is still a lot to do to improve both data collection and algorithms for automatic segmentation of arbitrary volumes. However, I believe this thesis is a step in the right direction, as the response and the adoption from the biological community clearly reflects the current needs for better, user friendly, free and open pipelines for segmentation of volumes.

The immediate future of this work includes its extension and generalization to a much wider public and collection of segmented datasets at large scale. Working towards that, over the last few months work has been done in that direction for long term studies. The first one includes an in-depth study of the variability of human annotations, for which we started collecting annotations over the last few months at Diamond Light Source from 30 work experience students, and further data collection meetings are planned for the next year. The aim of the study is to understand how non-expert humans annotate datasets and how many of annotations are needed to verify and answer biological questions, as it is currently a major issue in the biological community that much biological *knowledge* comes from custom semi-automatic segmentation techniques designed to model the underlying data with hand-crafted models that are designed based on the *known* reality. This is, semi-automatic segmentation models are created to extract and

validate expert hypothesis and assumptions, and thus, are in general highly biased towards the hypothesis. We aim to study how much of that bias is introduced into the segmentations in the form of segmentation errors or artefacts.

Additionally, we are just starting a long-term collaboration with Zooniverse (Borne and Team, 2011; Simpson et al., 2014) to incorporate many features and algorithms of this thesis (e.g. SuRVoS) as a remote API service to use as a middle-level segmentation (and feature extraction) and allow citizens to easily annotate complex data. We aim to make the annotation sub-volumes super-easy so that any non-experienced user is able to do it and use zooniverse to gather large-scale annotations.

On the extension of this thesis from the computer vision perspective, there is a a lot of room for improvement in many algorithms presented here. SMURFS and HPCS could be further improved and its GPU implementation could be studied. Once *enough* large-scale biologically representative data has been collected, more robust segmentation models could be used, such as Convolutional Neural Networks (e.g. U-Net (Çiçek et al., 2016)), and further studied in order to generate human-level segmentations fully automatically. I believe that the currently trending Deep CNN models for segmentation could provide a lot of help for the structural biological community, the field is just not ready for them yet, as there is no large enough databases out there to exploit their full potential. Algorithms, pipelines and tools likes the ones presented along these chapters can begin to change this.

A | Markov Random Field Models in Vision

This appendix aims to give a brief introduction to Markov Random Field (MRF) models for vision problems. The following content will be based on the exemplar review from Li (1994) and the recent book (Blake et al., 2011), to which I refer the reader for more in-depth details.

MRF models have been very popular since the 1990s to model different vision problems. To model vision problems with MRFs, these are viewed as Graph Labelling problems. A Graph Labelling problem is defined by a graph $G = (V, E)$ where V represents the set of nodes and E represents the edges connecting some (or all if it is fully connected) nodes. Each of the nodes $i \in V = \{1, \ldots, N\}$ will then have a label f_i from a label set $f_i \in \mathcal{L} = \{1, \ldots, L\}$. A label could be discrete or in the continuous domain, but for purposes of this thesis the rest of the chapter will be focus on discrete MRFs. A family of random variables $\boldsymbol{F} = \{F_1, \ldots, F_N\}$ are defined over the nodes on V, each one assuming a value in $\mathcal{L}$. Configurations of F, where all the variables on F are assigned a value, represent different solutions to the labelling problem, and can represent a denoised image, segmented (labelled) image, depth map or any other result, depending on the problem being modelled. The set of all possible configurations $\boldsymbol{C}$ for a labelling problems is too large, as from a combinatorial perspective there are L^N possible solutions.

From a Bayesian perspective, with $\boldsymbol{x}$ the current observation and assuming that both the a priori probability of a configuration $p(\boldsymbol{f})$ and the likelihood densities $p(\boldsymbol{x}|\boldsymbol{f})$ are known, the Maximum A Poseriori (MAP) estimate can be obtained the Bayesian rule

$$p(\boldsymbol{f}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{f}) \cdot p(\boldsymbol{f})}{p(\boldsymbol{x})}, \tag{A.1}$$

where $p(\boldsymbol{x})$, the density function of $\boldsymbol{x}$ does not affect the MAP solution, and thus, the optimal labelling can be written as:

$$\boldsymbol{f}^* = \underset{\boldsymbol{f} \in \boldsymbol{C}}{\operatorname{argmax}} \quad p(\boldsymbol{F} = \boldsymbol{f} \mid \boldsymbol{x}) \tag{A.2}$$

Markov Random Fields can then be used as prior $p(\boldsymbol{f})$ to add contextual or spatial dependencies into the labelling problem. First, a neighbourhood has to be defined, which in our case is given by the edges E in a graph, thus let $\mathcal{N}(i)$ denote the set of neighbouring nodes connected to node i. According to (Besag, 1974), F is an MRF on V respect to E if an only if the probability distribution $P(F = \boldsymbol{f})$ follows a Gibbs distribution, which has the following form

$$p(\boldsymbol{f}) = Z^{-1} \times exp(-E(\boldsymbol{f})) \tag{A.3}$$

where Z is a normalization constant and E defines the *prior energy* over cliques of arbitrary order. A clique C is a sub-graph of G where all the nodes in it are fully connected. There can be cliques C_k of arbitrary orders $k \in \{1, \ldots, N\}$, where C_k denotes all the sub-graphs of G where k elements are fully connected, this yields the general form of E as

$$E(\boldsymbol{f}) = \sum_{i \in V} \psi_i(f_i) + \sum_{i,j \in C_2} \psi_{i,j}(f_i, f_j) + \sum_{i,j,k \in C_3} \psi_{i,j,k}(f_i, f_j, f_k) + \cdots \tag{A.4}$$

where ψ represents the prior potentials at different orders. Hence, with a fixed set of observations $\boldsymbol{x}$, the MAP solution to with an MRF prior is then equivalently found by minimizing the energy E, yielding the optimal labelling over the MRF configuration $\boldsymbol{f}^* \in \boldsymbol{S}$

$$\boldsymbol{f}^* = \underset{\boldsymbol{f} \in \boldsymbol{C}}{\operatorname{argmin}} \quad E(\boldsymbol{f}). \tag{A.5}$$

This yields the general form of MAP-MRF models for vision problems, being the most common ones MRFs with first and second order potentials, also named as unary and pairwise potentials respectively in the literature:

$$E(\boldsymbol{f}) = \sum_{p \in V} \psi_p(f_p) + \lambda \sum_{p,q \in E} w_{p,q} \cdot \psi_{p,q}(f_p, f_q) \tag{A.6}$$

where λ controls the relevance of the unary potential against the pairwise potential (or how much we believe in each) and $w_{p,q}$ being the similarity weight between nodes p and q.

MRFs can be used to model different vision problems, as simple example we could consider image binarization or binary segmentation ($\mathcal{L} = \{0, 1\}$). In both cases the

observed variables $\boldsymbol{x}$ are defined the intensity of each pixel in the gray-scale image, and the unary potential can therefore be defined as $\psi_p(0) = I_p$ and $\psi_p(1) = 255 - I_p$ for labels 0 and 1 respectively. The pairwise potential can be set to the Potts model $\psi_{p,q}(f_p, f_q) = [f_p = f_q]$ which evaluates to 1 if $f_p = f_q$ or 0 otherwise, encouraging adjacent pixels to have the same label. Optimization of the above energy with the defined unary and pairwise potentials results in a piece-wise constant binary image.

Equation A.6 can be seen as a combinatorial problem, and can be solved by different methods, such as Integer Programming, Pseudo-boolean Optimization or other more related methods such as Loopy Belief Propagation. However, in vision problems nodes of the graph usually represent pixels of an image (or voxels in a volume) and the amount of hidden variables (nodes to be labelled) make the exact solution of MRF problems often intractable in practice. For the interest on this thesis, and given the large datasets we are dealing with, we will make use of efficient graph-based algorithms that yield efficient approximated solutions. Boykov et al. (1998) showed that a binary MAP-MRF can be formulated as graph-partitioning problem, and showed that finding the minimum E or the maximum flow in the graph G are equivalent. They developed a new maxflow algorithm (Boykov and Kolmogorov, 2004) orders of magnitude faster than previous approaches for vision problems. The multi-label problems can be solved by move-making algorithms, $\alpha - \beta$-swap and α-expansion (Boykov et al., 2001) that iterate pairwise binary MRFs across the product of all pair of labels. To finish, we make use of more recent and faster approaches such as QPBO (Rother et al., 2007) with a pseudo-boolean optimization framework and FastPD (Komodakis and Tziritas, 2007) that use linear programming and reduced binary instances of maxflow to solve the multi-label energy much quicker.

B Enhancing Features of Biological Volumes

In this appendix different features to highlight or enhance different parts or elements of the volume will be briefly introduced. The features could be split in the following categories:

- Denoising features: as shown in Figure B.1 are used to remove noise from the volume. While Gaussian filter smooths the image (blurring the boundaries), Total Variation denoising makes the volume piece-wise smooth while preserving strong boundaries.
- Local features: either by using a uniform (cube of radius r) or a gaussian neighrbourhood (Gaussian weighted neighgbourhood specified by σ) these features extract local information for each pixel. These include mean (averaging), standard deviation, normalization ($(data - mean)/std$) or mean subtraction to name a few, as shown in Figure B.1.
- Blob detection: Difference of Gaussian or Laplacian of Gaussian filters (either single or multi-scale versions) highlight blob-like structures different from their surrounding area. Blobs can also be very efficiently detected using the Determinant of the Hessian matrix, for which a very efficient approximation using integral images was given in (Bay et al., 2006). Figure B.3 shows an example results.
- Tubular structure detection: local features such as standard deviation and gradient magnitude can highlight boundaries on the volume, but more robust features such as the eigenvalues of the Hessian matrix or the Structure tensor also help highlighting them (Figure B.4).

Most of the features mentioned before can be applied in a multi-scale fashion to retrieve scale-invariant results, by filtering the maximum response.

Features are efficiently calculated in the GPU, as most of the features are computed independently for each voxel of the volume, and thus feature computation is one of the ridiculously parallel examples that GPUs are great at dealing with. Some exceptions apply, such as the Total Variation which is an iterative procedure, but each of the iterations can be speed up in the GPU independently. Most of the features are computed from convolution operations, for which efficient libraries exist (CUFFT).

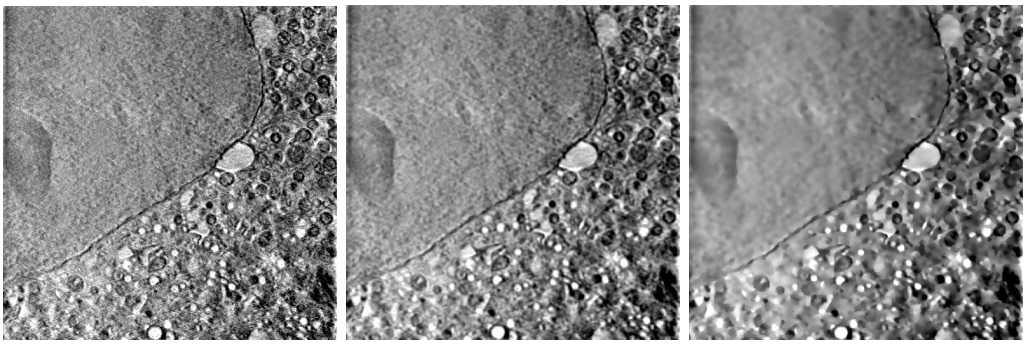


Figure B.1: From left to right: original data, Gaussian filter and Total Variation denoising (Goldstein and Osher, 2009).

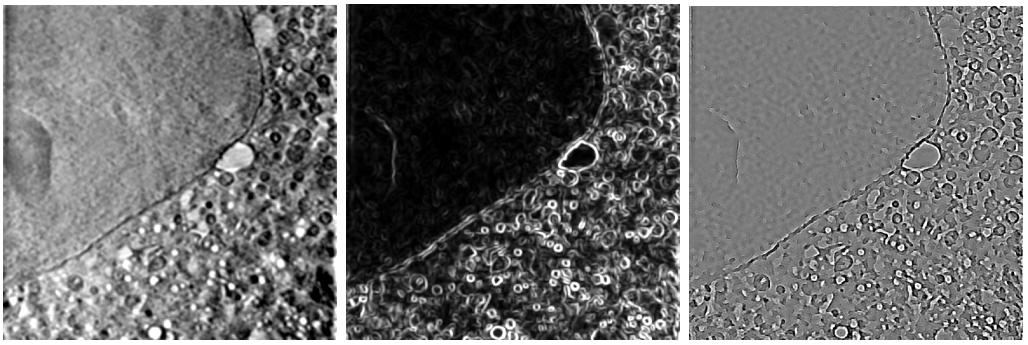


Figure B.2: From left to right: local averaging, local standard deviation and local mean subtraction.

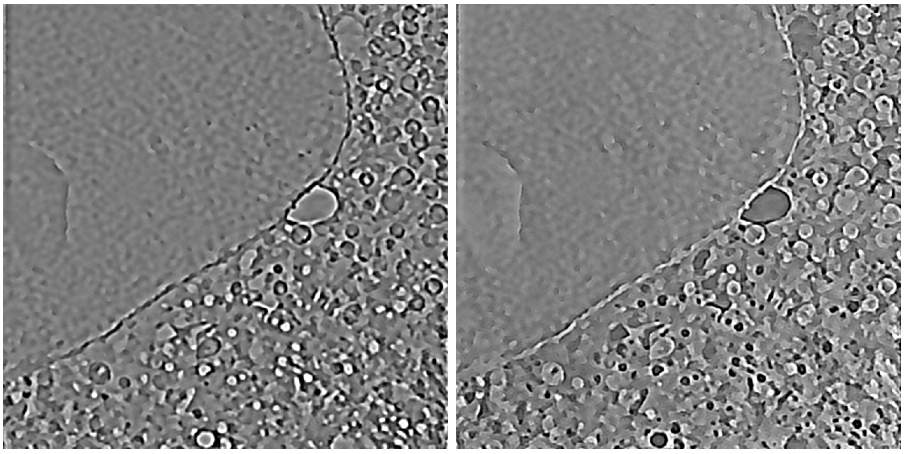


Figure B.3: From left to right: Difference of Gaussians and (inverted) Laplacian of Gaussians.

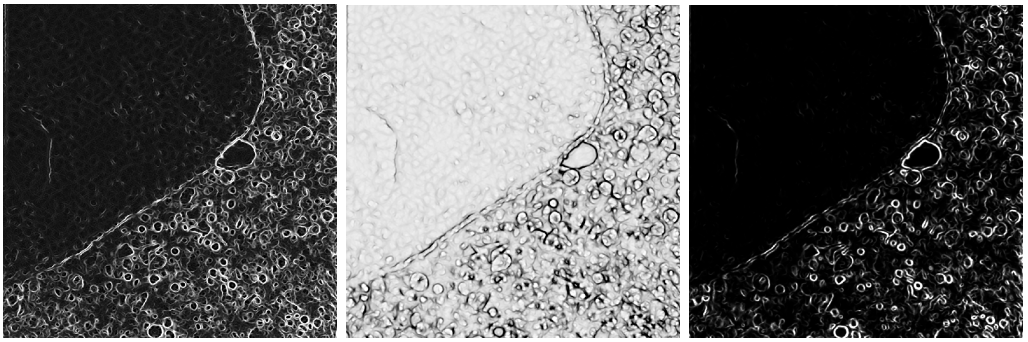


Figure B.4: From left to right: largest eigenvalue of the Hessian matrix ($\sigma = 1$), smallest eigenvalue of the Hessian matrix ($\sigma = 2$) and largest eigenvalue of the Structure tensor.

C | Superpixels for Leaf Detection and Segmentation

While studying superpixels and their possible applications at the begining of the PhD I came up across and participated in a collaborative study (Scharr et al., 2016) for individual leaf segmentation in plant images with the CVPPP dataset. I presented a fully automatic approach for individual leaf segmentation that used superpixels to determine individual leafs and then segment them.

The CVPPP consists in three different datasets $A1$, $A2$ and $A3$, containing images of three different plants imaged at different conditions, and each of them containing different images of similar plants of the same species under the same conditions. $A1$ contains 161 labelled image, $A2$ contains 40 and $A3$ 83. Each of the images has each of the leafs of the plant labelled individually, in a different colour, and thus, the goal was to come up with automatic methods for individual leaf segmentation. Figure C.1 shows an example of the images of $A1$ dataset, with its colour, grayscale and ground truth segmentation.

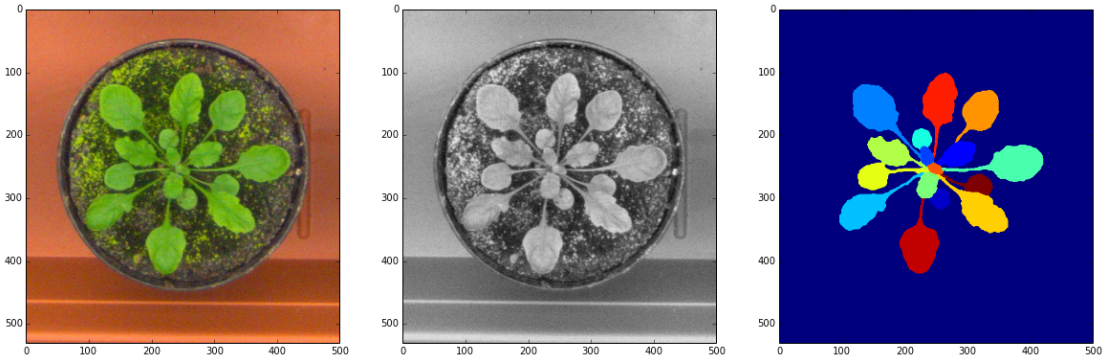


Figure C.1: Sample colour and grayscale images of 'A1' dataset, along with its ground truth.

At the time the the testing dataset was not provided, and the goal was to come up with a method and the organizers would then evaluate the results of it on previously unseen test datasets (the ground truth for the testing dataset was never provided).

Segmenting individual leafs from plant images is very challenging, as the algorithms have to deal with illumination changes, shadows and occlusions (among other challenges). I decided to propose then a completely unsupervised approach that was comprised of three different steps:

- Plant Segmentation
- Leaf Detection
- Leaf Segmentation

In order to perform plant segmentation and leaf detection the image was first over-segmented using SLIC (Achanta et al., 2012), and the whole process was split in five different steps:

- Superpixel over-segmentation in $L \cdot a \cdot b$ colour space using SLIC.
- Foreground (plant) extraction using seeded region growing in the superpixel RAG.
- Distance map calculation on extracted foreground.
- Individual leaf seed matching by identifying the super-pixels whose centroid lays in the local maxima of the distance map.
- Individual leaf segmentation by applying watershed transform with the extracted seeds.

Figure C.2 gives an overview of the detailed steps in a sketch version, and the full pipeline is described as follows:

Data preparation

For a given image, RGB image, it is first converted to the Lab colour space, to enhance discrimination between foreground leaves and background. A fixed number of superpixels are computed over the image. Empirically, 2000 pixels was found to provide good coverage of the leaves. The mean colour of the a channel (which characterizes the green colour of the image well) is extracted for each superpixel. A Region Adjacency Graph (superpixel neighbourhood graph) is created from the resulting superpixels.

Foreground extraction.

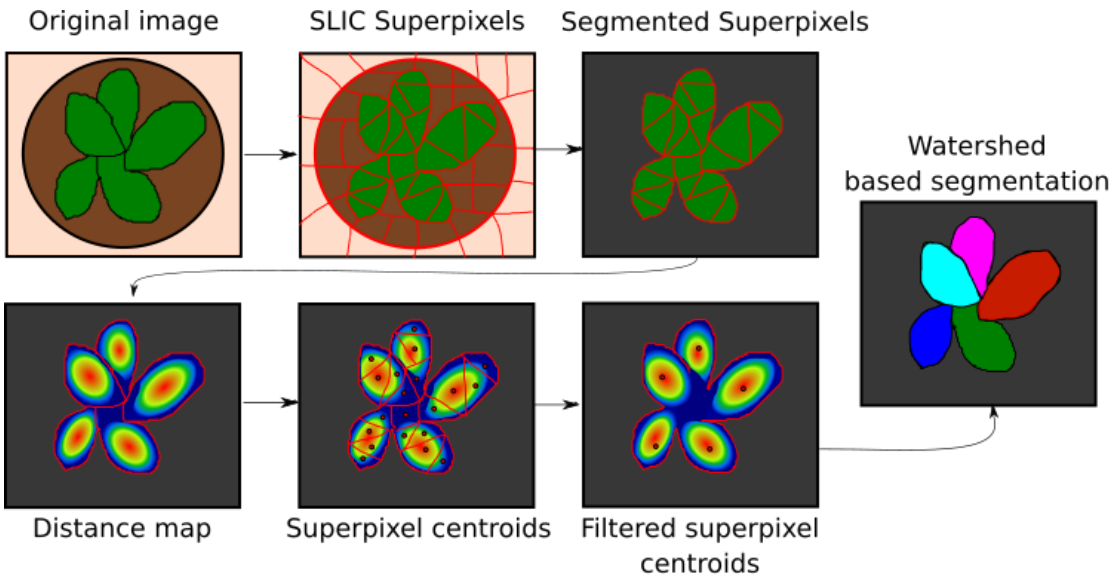


Figure C.2: Drawing of the proposed pipeline.

For dataset $A3$, which is more challenging and contain shadows, having the mean colour of each superpixel for channel a, a region growing approach (Adams and Bischof, 1994) is used, but instead that growing pixels from seeds, it is modified to operate in the superpixel space, which allows the complete plant to be extracted as superpixel mean colour descriptors are more robust to data noise than pixels itself. The superpixel with the lowest mean colour (the most bright green superpixel) defined in Lab space is used as the initial seed.

For A1 and A2, instead, since they do not contain shadows, a simple thresholding of the mean colour of each superpixel allows faster yet still accurate segmentation of the plant. Thresholds for the A1 and A2 are set to -25 and -15 respectively.

Leaf identification.

Once the plant is extracted from the background, superpixels not belonging to the plant are removed. A distance map is computed (first removing strong edges using the Canny detector) and the centroids are calculated for all superpixels. The centroids are calculated as the mean x, y coordinates from all the pixels within it. A local maxima filter with the radius of $2x$ the expected size of a superpixel (calculated from the number of superpixels and the size of the image) is applied to extract the superpixels that lay in the centre of the leaves. A superpixel is selected as a seed only if it is the most central in the leaf compared to its neighbours within a radius. This is implemented by considering the superpixel centroid value in the distance map, and filtering the superpixels that do not have the maximum value within its neighbours.

Leaf segmentation.

Finally, watershed segmentation (Vincent and Soille, 1991) is applied with the obtained initial seeds (the filtered superpixel centroids) over the image space, yielding the individual leaf segmentation.

Results

Using a Python implementation running on a i3 quad-core desktop with i3-4130 (3.4GHz) processor and 8GB memory, on average, each image takes < 1 second for dataset A1 and A2, and 15 seconds for A3. Overall, it is a fast method with no training required. It also could be tuned to get a much higher accuracy on a per-image basis, as it is completely unsupervised and parameters were empirically chosen for all the images in each of the three datasets. Figure C.5, Figure C.6 and Figure C.4 show qualitative results for different images, completely unsupervised. On the other hand, table C.6 shows quantitative results comparing to other methods present in the study. We refer the reader to the published study (Scharr et al., 2016) for more in-depth details and comparisons.

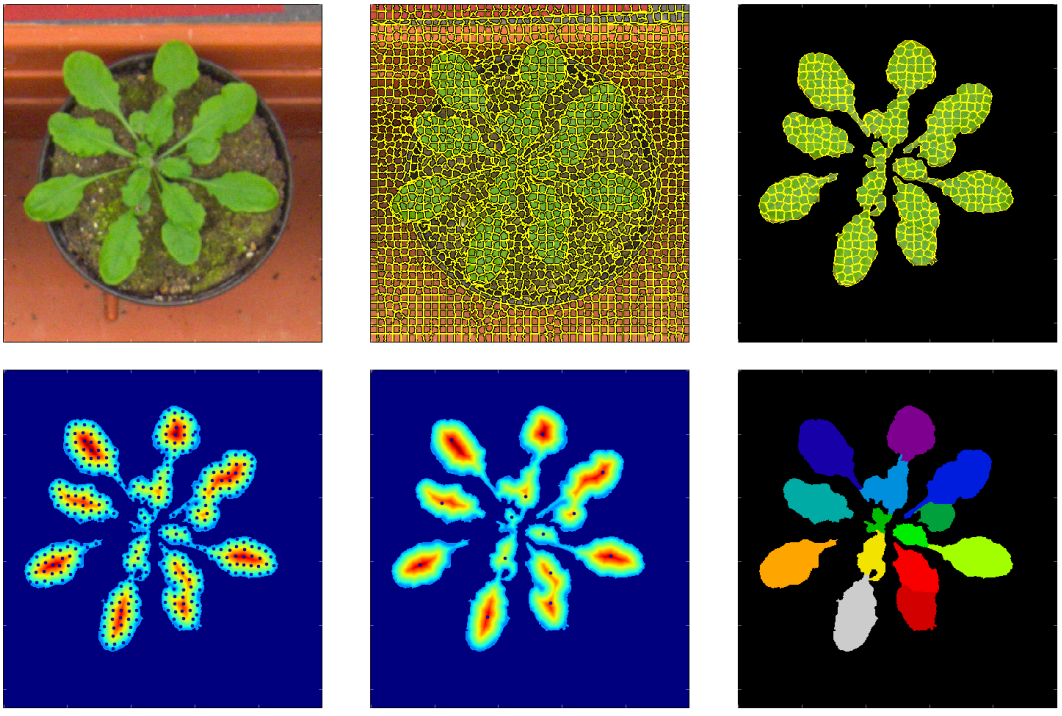


Figure C.3: Example results of our proposed pipeline. Top row, from left to right: image, superpixels and foreground extraction. Bottom row, from left to right: Distance map with superpixel centroids overlaid on it, distance map with filtered superpixel centroids and final segmentation by using watershed from the filtered centroids. Almost perfect location and segmentation of the leafs by removing the stems of the leafs.

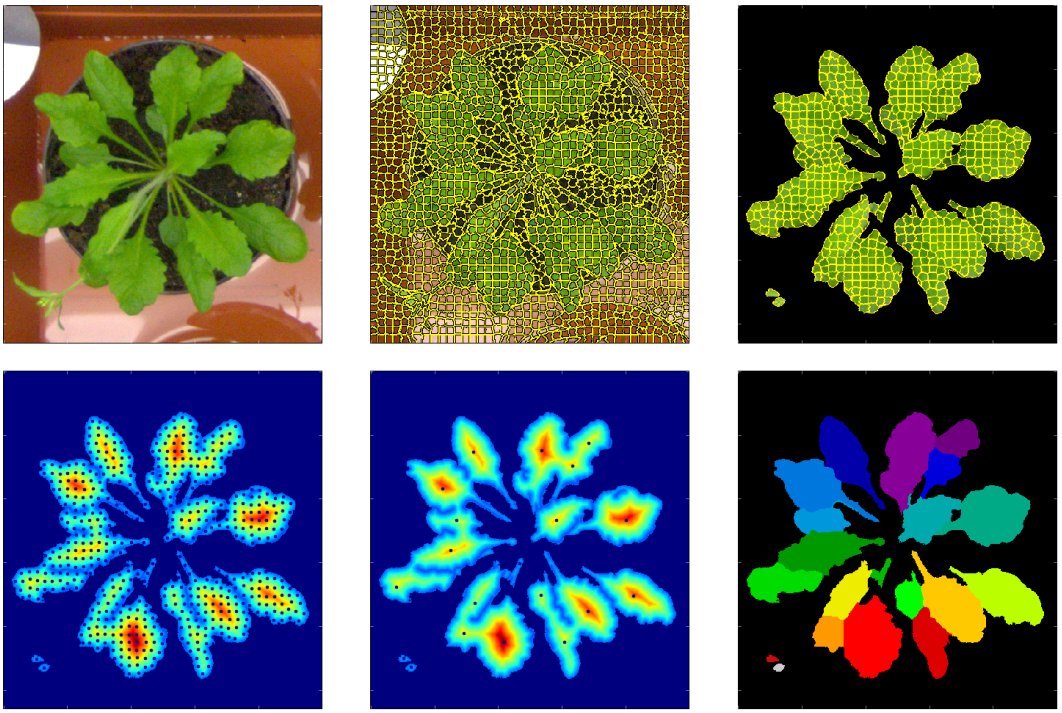


Figure C.4: Example results of our proposed pipeline. Top row, from left to right: image, superpixels and foreground extraction. Bottom row, from left to right: Distance map with superpixel centroids overlaid on it, distance map with filtered superpixel centroids and final segmentation by using watershed from the filtered centroids. This time the algorithms fails to segment the plant properly and two leafs are wrongly detected on the bottom left part.

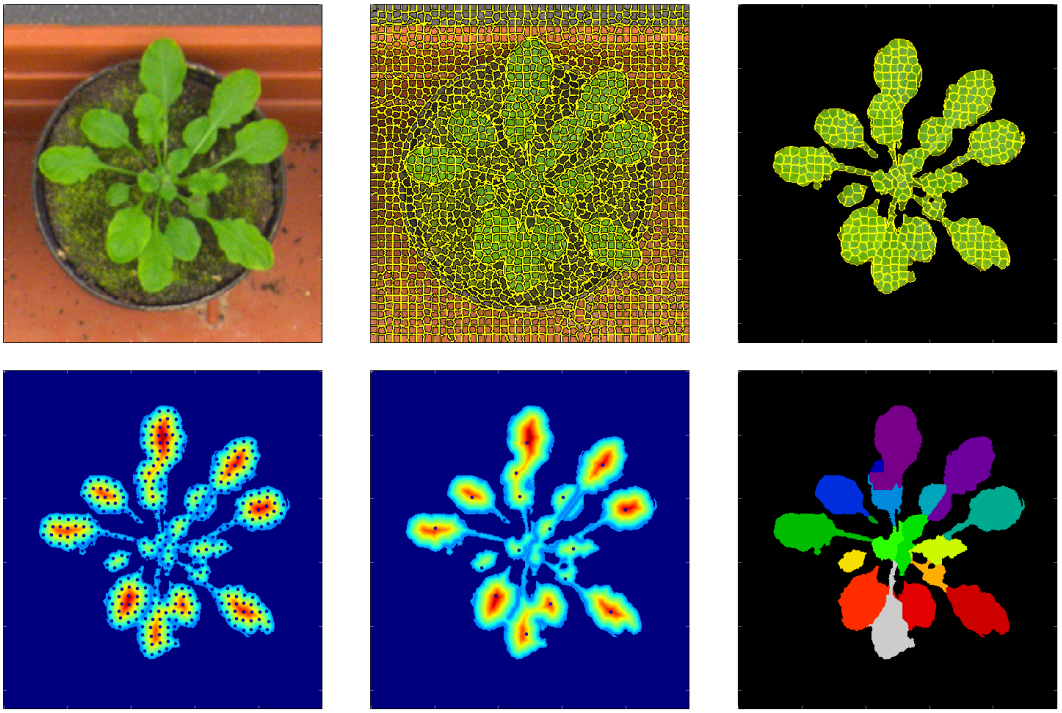


Figure C.5: Example results of our proposed pipeline. Top row, from left to right: image, superpixels and foreground extraction. Bottom row, from left to right: Distance map with superpixel centroids overlaid on it, distance map with filtered superpixel centroids and final segmentation by using watershed from the filtered centroids. It can be seen that it fails to detect some small leafs in the middle of the plant, and the brightness on the stem of the leafs make canny remove important boundaries and thus, watershed fails to properly segment leafs in the top section.

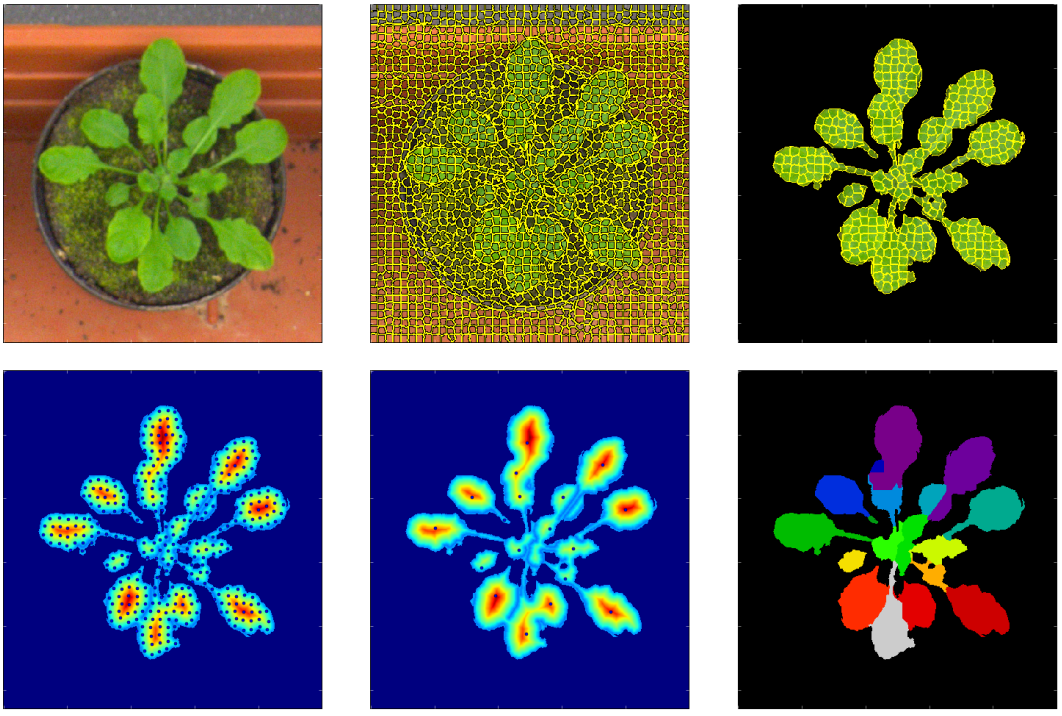


Figure C.6: Results for the individual leaf segmentation challenge. IPK, MSU and Wageningen show results for other three methods, detailed in the collaborative study (Scharr et al., 2016). Columns show Symmetric Best Dice score (SBD) meassuring both leaf detection and segmentation, Foreground-Background Dice (FBD) measuring how well the plant is segmented and the Difference in Count —DiC— and DiC shows the (absolute) difference on the total of objects count and the ground truth objects present.

C | Bibliography

. SuRVoS v1.0.

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

Adams, R. and L. Bischof
1994. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647.

Al-Kofahi, Y., W. Lassoued, W. Lee, and B. Roysam
2010. Improved automatic detection and segmentation of cell nuclei in histopathology images. *IEEE Transactions on Biomedical Engineering*, 57(4):841–852.

Apostol, B. L., A. Kazantsev, S. Raffioni, K. Illes, J. Pallos, L. Bodai, N. Slepko, J. E. Bear, F. B. Gertler, S. Hersch, and Others
2003. A cell-based assay for aggregation inhibitors as therapeutics of polyglutamine-repeat disease and validation in Drosophila. *Proceedings of the National Academy of Sciences*, 100(10):5950–5955.

Arandjelović, R. and A. Zisserman
2012. Three things everyone should know to improve object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, Pp. 2911–2918. IEEE.

Arbelaez, P., M. Maire, C. Fowlkes, and J. Malik
2011. Contour detection and hierarchical image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):898–916.

Arthur, D. and S. Vassilvitskii
2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth*

annual ACM-SIAM symposium on Discrete algorithms, Pp. 1027–1035. Society for Industrial and Applied Mathematics.

B24

. B24: Cryo Transmission X-ray Microscopy (cryo-TXM).

Bay, H., T. Tuytelaars, and L. Van Gool

2006. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, Pp. 404–417.

Beier, T., C. Pape, N. Rahaman, T. Prange, S. Berg, D. D. Bock, A. Cardona, G. W. Knott, S. M. Plaza, L. K. Scheffer, et al.

2017. Multicut brings automated neurite segmentation closer to human performance. *Nature Methods*, 14(2):101–102.

Besag, J.

1974. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, Pp. 192–236.

Black, M. J., G. Sapiro, D. H. Marimont, and D. Heeger

1998. Robust anisotropic diffusion. *IEEE Transactions on image processing*, 7(3):421–432.

Blake, A., P. Kohli, and C. Rother

2011. *Markov random fields for vision and image processing*. Mit Press.

Borne, K. and Z. Team

2011. The zooniverse: A framework for knowledge discovery from citizen science data. In *AGU Fall Meeting Abstracts*.

Boykov, Y. and G. Funka-Lea

2006. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131.

Boykov, Y. and V. Kolmogorov

2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137.

Boykov, Y., O. Veksler, and R. Zabih

1998. Markov random fields with efficient approximations. In *Computer vision and pattern recognition, 1998. Proceedings. 1998 IEEE computer society conference on*, Pp. 648–655. IEEE.

Boykov, Y., O. Veksler, and R. Zabih

2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239.

Boykov, Y. Y. and M.-P. Jolly

2001. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, Pp. 105–112. IEEE.

Breiman, L.

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

Breiman, L., J. Friedman, C. J. Stone, and R. A. Olshen

1984. *Classification and regression trees*. CRC press.

Carzaniga, R., M.-C. Domart, L. M. Collinson, and E. Duke

2014. Cryo-soft x-ray tomography: a journey into the world of the native-state cell. *Protoplasma*, 251(2):449–458.

Caselles, V., R. Kimmel, and G. Sapiro

1995. Geodesic active contours. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, Pp. 694–699. IEEE.

Caselles, V., R. Kimmel, and G. Sapiro

1997. Geodesic active contours. *International journal of computer vision*, 22(1):61–79.

Celebi, M. E., H. A. Kingravi, and P. A. Vela

2013. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 40(1):200–210.

Chambolle, A.

2004. An Algorithm for Total Variation Minimization and Applications. *Journal of Mathematical Imaging and Vision*.

Chan, T. F. and L. A. Vese

2001. Active contours without edges. *IEEE Transactions on image processing*, 10(2):266–277.

Chen, L.-C., G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille

2014. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*.

Chen, T. C. T. and Z. S. Z. Shen

2010. An adaptive image segmentation method using region growing. *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on*, 7.

Cheng, Y.

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

Cheriton, D. and R. E. Tarjan

1976. Finding minimum spanning trees. *SIAM Journal on Computing*, 5(4):724–742.

Çiçek, Ö., A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger

2016. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Pp. 424–432. Springer.

Ciresan, D., A. Giusti, L. M. Gambardella, and J. Schmidhuber

2012. Deep neural networks segment neuronal membranes in electron microscopy images. In *Advances in neural information processing systems*, Pp. 2843–2851.

Comaniciu, D. and P. Meer

2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619.

Cormen, T., C. Leiserson, R. Rivest, and C. Stein

2009. 21. data structures for disjoint sets. *Introduction to Algorithms,*, Pp. 498–524.

Cortes, C. and V. Vapnik

1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Cover, T. M. and J. A. Thomas

2012. *Elements of information theory*. John Wiley & Sons.

Criminisi, A., J. Shotton, and S. Bucciarelli

2009. Decision forests with long-range spatial context for organ localization in ct volumes. In *MICCAI Workshop on Probabilistic Models for Medical Image Analysis*, volume 1, Pp. 146–158.

Criminisi, A., J. Shotton, E. Konukoglu, et al.

2012. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 7(2–3):81–227.

Criminisi, A., J. Shotton, D. P. Robertson, and E. Konukoglu

2010. Regression forests for efficient anatomy detection and localization in ct studies. *MCV*, 2010:106–117.

Dalal, N. and B. Triggs

2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, Pp. 886–893. IEEE.

Darrow, M., W. Dai, P. Mitchell, I. Luengo, C. Wah, and E. Duke

2016. Organelle changes in a huntingtons disease model using cryogenic soft x-ray tomography. In *The 16th European Microscopy Congress (EMC, **Oral Presentation**)*. http://emc-proceedings.com/abstract/organelle-changes-in-a-huntingtons-disease-model-using-cryogenic-soft-x-ray-tomography/.

Darrow, M. C., I. Luengo, M. Basham, M. C. Spink, S. Irvine, A. P. French, A. W. Ashton, and E. M. Duke

2017. Volume segmentation and analysis of biological materials using survos (super-region volume segmentation) workbench. *JoVE: Journal of Visual Experiments*.

De Chaumont, F., S. Dallongeville, N. Chenouard, N. Hervé, S. Pop, T. Provoost, V. Meas-Yedid, P. Pankajakshan, T. Lecomte, Y. Le Montagner, et al.

2012. Icy: an open bioimage informatics platform for extended reproducible research. *Nature methods*, 9(7):690–696.

Dijkstra, E. W.

1959. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

Dueck, D. and B. J. Frey

2007. Non-metric affinity propagation for unsupervised image categorization. In *ICCV 2007*.

Duke, E., K. Dent, M. Razi, and L. M. Collinson

2014a. Biological applications of cryo-soft X-ray tomography. *Journal of microscopy*, 255(2):65–70.

Duke, E. M., M. Razi, A. Weston, P. Guttmann, S. Werner, K. Henzler, G. Schneider, S. A. Tooze, and L. M. Collinson

2014b. Imaging endosomes and autophagosomes in whole mammalian cells using correlative cryo-fluorescence and cryo-soft x-ray microscopy (cryo-clxm). *Ultramicroscopy*, 143:77–87.

Edmonds, J. and R. M. Karp

1972. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264.

Ester, M., H.-P. Kriegel, J. Sander, X. Xu, et al.
1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, Pp. 226–231.

Falcão, A. X., J. K. Udupa, and F. K. Miyazawa
2000. An ultra-fast user-steered image segmentation paradigm: live wire on the fly. *IEEE transactions on medical imaging*, 19(1):55–62.

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

Flake, G. W., R. E. Tarjan, and K. Tsioutsiouliklis
2004. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408.

Ford, L. R. and D. Fulkerson
1962. Flows in networks.

Ford, L. R. and D. R. Fulkerson
1956. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404.

Frangi, A. F., W. J. Niessen, K. L. Vincken, and M. A. Viergever
1998. Multiscale vessel enhancement filtering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Pp. 130–137. Springer.

Freedman, D. and T. Zhang
2005. Interactive graph cut based segmentation with shape priors. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, Pp. 755–762. IEEE.

Freund, Y. and R. E. Schapire
1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, Pp. 23–37. Springer.

Friedman, J., T. Hastie, and R. Tibshirani
2001. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin.

Friedman, J. H.
2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, Pp. 1189–1232.

Friedman, J. H.

2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378.

Fulkerson, B., A. Vedaldi, and S. Soatto

2009. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, Pp. 670–677. IEEE.

Garbay, C., J. Chassery, and G. Brugal

1986. An iterative region-growing process for cell image segmentation based on local color similarity and global shape criteria. *Analytical and quantitative cytology and histology*, 8(1):25–34.

Geremia, E., B. H. Menze, O. Clatz, E. Konukoglu, A. Criminisi, and N. Ayache

2010. Spatial decision forests for ms lesion segmentation in multi-channel mr images. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Pp. 111–118. Springer.

Geurts, P., D. Ernst, and L. Wehenkel

2006. Extremely randomized trees. *Machine learning*, 63(1):3–42.

Goldstein, T. and S. Osher

2009. The Split Bregman Method for L1-Regularized Problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343.

Gomory, R. E. and T. C. Hu

1961. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9(4):551–570.

Grady, L., T. Schiwietz, S. Aharon, and R. Westermann

2005. Random walks for interactive organ segmentation in two and three dimensions: Implementation and validation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2005*, Pp. 773–780.

Hariharan, B., P. Arbeláez, R. Girshick, and J. Malik

2014. Simultaneous detection and segmentation. In *European Conference on Computer Vision*, Pp. 297–312. Springer.

Hecksel, C. W., M. C. Darrow, W. Dai, et al.

2016a. Quantifying variability of manual annotation in cryo-electron tomograms. *Microscopy and Microanalysis*, 22(03):487–496.

Hecksel, C. W., M. C. Darrow, W. Dai, J. G. Galaz-Montoya, J. A. Chin, P. G. Mitchell, S. Chen, J. Jakana, M. F. Schmid, and W. Chiu

2016b. Quantifying Variability of Manual Annotation in Cryo-Electron Tomograms.

Microscopy and microanalysis: the official journal of Microscopy Society of America, Microbeam Analysis Society, Microscopical Society of Canada, Pp. 1–10.

Hong, X., H. Chang, S. Shan, X. Chen, and W. Gao

2009. Sigma set: A small second order statistical region descriptor. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Pp. 1802–1809. IEEE.

Hong, X., H. Chang, S. Shan, B. Zhong, X. Chen, and W. Gao

2010. Sigma set based implicit online learning for object tracking. *IEEE Signal Processing Letters*, 17(9):807–810.

Ishikawa, H.

2003. Exact optimization for markov random fields with convex priors. *IEEE transactions on pattern analysis and machine intelligence*, 25(10):1333–1336.

Jain, A. K. and F. Farrokhnia

1991. Unsupervised texture segmentation using gabor filters. *Pattern recognition*, 24(12):1167–1186.

Kainz, P., M. Urschler, S. Schulter, P. Wohlhart, and V. Lepetit

2015. You should use regression to detect cells. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Pp. 276–283. Springer.

Kass, M., A. Witkin, and D. Terzopoulos

1988. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331.

Kaufman, L.

1987. Clustering by means of medoids. *Statistical data analysis based on the L1-norm and related methods*.

Kocak, A., K. Cizmeciler, A. Erdem, and E. Erdem

2014. Top down saliency estimation via superpixel-based discriminative dictionaries. In *BMVC*.

Kohli, P., P. H. Torr, et al.

2009. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324.

Kolmogorov, V., Y. Boykov, and C. Rother

2007. Applications of parametric maxflow in computer vision. In *2007 IEEE 11th International Conference on Computer Vision*, Pp. 1–8. IEEE.

Komodakis, N. and G. Tziritas

2007. Approximate labeling via graph cuts based on linear programming. *IEEE transactions on pattern analysis and machine intelligence*, 29(8):1436–1453.

Komodakis, N., G. Tziritas, and N. Paragios

2008. Performance vs computational efficiency for optimizing single and dynamic mrfs: Setting the state of the art with primal-dual strategies. *Computer Vision and Image Understanding*, 112(1):14–29.

Kontschieder, P., P. Kohli, J. Shotton, and A. Criminisi

2013. Geof: Geodesic forests for learning coupled predictors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 65–72.

Kremer, J. R., D. N. Mastronarde, and J. McIntosh

1996. Computer Visualization of Three-Dimensional Image Data Using IMOD. *Journal of Structural Biology*, 116(1):71–76.

Kruskal, J. B.

1956. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50.

Ladickỳ, L., C. Russell, P. Kohli, and P. H. Torr

2014. Associative hierarchical random fields. *IEEE transactions on pattern analysis and machine intelligence*, 36(6):1056–1077.

Lempitsky, V., M. Verhoek, J. A. Noble, and A. Blake

2009. Random forest classification for automatic delineation of myocardium in real-time 3d echocardiography. In *International Conference on Functional Imaging and Modeling of the Heart*, Pp. 447–456. Springer.

Leskovec, J., A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance

2007. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, Pp. 420–429. ACM.

Levinshtein, A., A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi

2009. Turbopixels: Fast superpixels using geometric flows. *IEEE transactions on pattern analysis and machine intelligence*, 31(12):2290–2297.

Li, C., C. Xu, C. Gui, and M. D. Fox

2005. Level set evolution without re-initialization: a new variational formulation. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, Pp. 430–436. IEEE.

Li, C., C. Xu, C. Gui, and M. D. Fox

2010. Distance regularized level set evolution and its application to image segmentation. *IEEE Transactions on image processing*, 19(12):3243–3254.

Li, L.-J., R. Socher, and L. Fei-Fei

2009. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Pp. 2036–2043. IEEE.

Li, S. Z.

1994. Markov random field models in computer vision. In *European Conference on Computer Vision*, Pp. 361–370. Springer.

Li, Y., J. Sun, C.-K. Tang, and H.-Y. Shum

2004. Lazy snapping. In *ACM Transactions on Graphics (ToG)*, volume 23, Pp. 303–308. ACM.

Li, Z. and J. Chen

2015. Superpixel segmentation using linear spectral clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Pp. 1356–1363. IEEE.

Lin, G., C. Shen, A. van den Hengel, and I. Reid

2016. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 3194–3203.

Liu, M.-Y., O. Tuzel, S. Ramalingam, and R. Chellappa

2011. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, Pp. 2097–2104. IEEE.

Lowe, D. G.

1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, Pp. 1150–1157. Ieee.

Luc, P., C. Couprie, S. Chintala, and J. Verbeek

2016. Semantic segmentation using adversarial networks. *arXiv preprint arXiv:1611.08408*.

Lucchi, A., K. Smith, R. Achanta, G. Knott, and P. Fua

2012a. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE transactions on medical imaging*, 31(2):474–486.

Lucchi, A., K. Smith, R. Achanta, G. Knott, and P. Fua

2012b. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE transactions on medical imaging*, 31(2):474–86.

Luengo, I., M. Basham, and A. P. French

2016a. Fast global interactive volume segmentation with regional supervoxel descriptors. In *SPIE Medical Imaging*, Pp. 97842D—–97842D. International Society for Optics and Photonics.

Luengo, I., M. Basham, and A. P. French

2016b. Hierarchical piecewise-constant super-regions. *arXiv preprint arXiv:1605.05937*.

Luengo, I., M. Basham, and A. P. French

2016c. Selective labeling: Identifying representative sub-volumes for interactive segmentation. In *MICCAI 2016: International Workshop on Patch-based Techniques in Medical Imaging*, Pp. 17–24. Springer.

Luengo, I., M. Basham, and A. P. French

2016d. SMURFS: superpixels from multi-scale refinement of super-regions. In *British Machine Vision Conference (BMVC)*.

Luengo, I., M. Basham, and A. P. French

2017a. Extremely contextual forests for multi-scale n-d image segmentation. In *To be submitted.*

Luengo, I., M. C. Darrow, M. Basham, and A. P. French

2016e. Hierarchical regions for fast biomedical volume segmentation. In *Tomography for Scientific Advancement 2016 (TOSCA,* ***Oral Presentation****)*. http://www.nhm.ac.uk/content/dam/nhmwww/our-science/dpts-facilities-staff/Coreresearchlabs/ToScA-Programme-2016-FINAL-sept.pdf.

Luengo, I., M. C. Darrow, M. C. Spink, Y. Sun, W. Dai, C. Y. He, W. Chiu, T. Pridmore, A. W. Ashton, E. M. Duke, et al.

2017b. Survos: Super-region volume segmentation workbench. *Journal of Structural Biology*, 198(1):43–53.

Lukinavičius, G., K. Umezawa, N. Olivier, A. Honigmann, G. Yang, T. Plass, V. Mueller, L. Reymond, I. R. Corrêa Jr, Z.-G. Luo, et al.

2013. A near-infrared fluorophore for live-cell super-resolution microscopy of cellular proteins. *Nature chemistry*, 5(2):132–139.

Machairas, V., M. Faessel, D. Cárdenas-Peña, T. Chabardes, T. Walter, and E. Decencière

2015. Waterpixels. *IEEE Transactions on Image Processing*, 24(11):3707–3716.

Malcolm, J., Y. Rathi, and A. Tannenbaum

2007. Graph cut segmentation with nonlinear shape priors. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 4, Pp. IV–365. IEEE.

Malladi, R., J. A. Sethian, and B. C. Vemuri

1995. Shape modeling with front propagation: A level set approach. *IEEE transactions on pattern analysis and machine intelligence*, 17(2):158–175.

Markram, H.

2012. The human brain project. *Scientific American*, 306(EPFL-ARTICLE-183377):50–5.

Mastronarde, D. N.

1997. Dual-axis tomography: an approach with alignment methods that preserve resolution. *Journal of structural biology*, 120(3):343–352.

Menze, B., B. Kelm, D. Splitthoff, U. Koethe, and F. Hamprecht

2011. On oblique random forests. *Machine Learning and Knowledge Discovery in Databases*, Pp. 453–469.

Montillo, A., J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi

2011. Entangled decision forests and their application for semantic segmentation of ct images. In *Biennial International Conference on Information Processing in Medical Imaging*, Pp. 184–196. Springer.

Moore, A. P., S. J. Prince, J. Warrell, U. Mohammed, and G. Jones

2008. Superpixel lattices. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, Pp. 1–8. IEEE.

Mortensen, E. N. and W. A. Barrett

1998. Interactive segmentation with intelligent scissors. *Graphical models and image processing*, 60(5):349–384.

Müller, W. G., J. B. Heymann, K. Nagashima, P. Guttmann, S. Werner, S. Rehbein, G. Schneider, and J. G. McNally

2012. Towards an atlas of mammalian cell ultrastructure by cryo soft x-ray tomography. *Journal of structural biology*, 177(2):179–192.

Neubert, P. and P. Protzel

2014. Compact watershed and preemptive slic: On improving trade-offs of superpixel segmentation algorithms. In *ICPR*, Pp. 996–1001.

Nieuwenhuis, C. and D. Cremers

2013. Spatially varying color distributions for interactive multilabel segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1234–1247.

Norouzi, M., M. D. Collins, D. J. Fleet, and P. Kohli

2015. Co2 forest: Improved random forest by continuous optimization of oblique splits. *arXiv preprint arXiv:1506.06155*.

Ozuysal, M., M. Calonder, V. Lepetit, and P. Fua

2010. Fast keypoint recognition using random ferns. *IEEE transactions on pattern analysis and machine intelligence*, 32(3):448–461.

Paganin, D., S. Mayo, T. E. Gureyev, P. R. Miller, and S. W. Wilkins

2002. Simultaneous phase and amplitude extraction from a single defocused image of a homogeneous object. *Journal of microscopy*, 206(1):33–40.

Palenstijn, W., K. Batenburg, and J. Sijbers

2011. Performance improvements for iterative electron tomography reconstruction using graphics processing units (gpus). *Journal of structural biology*, 176(2):250–253.

Pauly, O., B. Glocker, A. Criminisi, D. Mateus, A. Möller, S. Nekolla, and N. Navab

2011. Fast multiple organ detection and localization in whole-body mr dixon sequences. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*, Pp. 239–247.

Pavlidis, T. and Y.-T. Liow

1990. Integrating region growing and edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):225–233.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and Others

2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Perazzi, F., P. Krähenbühl, Y. Pritch, and A. Hornung

2012. Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, Pp. 733–740. IEEE.

Pettersen, E. F., T. D. Goddard, C. C. Huang, G. S. Couch, D. M. Greenblatt, E. C. Meng, and T. E. Ferrin

2004. UCSF Chimeraa visualization system for exploratory research and analysis. *Journal of computational chemistry*, 25(13):1605–1612.

Phan, T., S. Sohoni, D. M. Chandler, and E. C. Larson

2012. Performance-analysis-based acceleration of image quality assessment. In *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*, Pp. 81–84. IEEE.

Pock, T., A. Chambolle, D. Cremers, and H. Bischof

2009. A convex relaxation approach for computing minimal partitions. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, Pp. 810–817. IEEE.

Pohle, R. and K. D. Toennies

2001. Segmentation of medical images using adaptive region growing. In *Proc. SPIE Medical Imaging*, volume 4322, Pp. 1337–1346.

Potts, R. B.

1952. Some generalized order-disorder transformations. In *Mathematical proceedings of the cambridge philosophical society*, volume 48, Pp. 106–109. Cambridge University Press.

Price, B. L., B. Morse, and S. Cohen

2010. Geodesic graph cut for interactive image segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, Pp. 3161–3168. IEEE.

Ren, C. Y. and I. Reid

2011. gslic: a real-time implementation of slic superpixel segmentation. *University of Oxford, Department of Engineering, Technical Report*.

Ren, S., K. He, R. Girshick, and J. Sun

2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, Pp. 91–99.

Ren, X. and J. Malik

2003. Learning a classification model for segmentation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, Pp. 10–17. IEEE.

Ren, Z. and G. Shakhnarovich

2013. Image segmentation by cascaded region agglomeration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Pp. 2011–2018.

Ronneberger, O., P. Fischer, and T. Brox

2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Pp. 234–241. Springer.

Rother, C., V. Kolmogorov, and A. Blake

2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, volume 23, Pp. 309–314. ACM.

Rother, C., V. Kolmogorov, V. Lempitsky, and M. Szummer

2007. Optimizing binary mrfs via extended roof duality. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, Pp. 1–8. IEEE.

Rousson, M. and N. Paragios

2002. Shape priors for level set representations. *Computer VisionECCV 2002*, Pp. 416–418.

Rudin, L. I., S. Osher, and E. Fatemi

1992. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268.

Santner, J., T. Pock, and H. Bischof

2010. Interactive multi-label segmentation. In *Asian Conference on Computer Vision*, Pp. 397–410. Springer.

Santner, J., T. Pock, and H. Bischof

2011. Interactive multi-label segmentation. *Computer Vision–ACCV 2010*, Pp. 397–410.

Santner, J., M. Unger, T. Pock, C. Leistner, A. Saffari, and H. Bischof

2009. Interactive texture segmentation using random forests and total variation. In *BMVC*, Pp. 1–12. Citeseer.

Saran, H. and V. V. Vazirani

1995. Finding k cuts within twice the optimal. *SIAM Journal on Computing*, 24(1):101–108.

Schaeffer, S. E.

2007. Graph clustering. *Computer science review*, 1(1):27–64.

Scharr, H., M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J.-M. Pape, G. Polder, D. Vukadinovic, et al.

2016. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications*, 27(4):585–606.

Schindelin, J., I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, and Others

2012. Fiji: an open-source platform for biological-image analysis. *Nature methods*, 9(7):676–682.

Sethian, J. A.

1999. *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge university press.

Shi, J. and J. Malik

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

Shotton, J., J. Winn, C. Rother, and A. Criminisi

2006. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision*, Pp. 1–15. Springer.

Sijbers, J., P. Scheunders, M. Verhoye, A. Van der Linden, D. Van Dyck, and E. Raman

1997. Watershed-based segmentation of 3d mr data for volume quantization. *Magnetic Resonance Imaging*, 15(6):679–688.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al.

2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484.

Simpson, R., K. R. Page, and D. De Roure

2014. Zooniverse: observing the world's largest citizen science platform. In *Proceedings of the 23rd international conference on world wide web*, Pp. 1049–1054. ACM.

Slabaugh, G. and G. Unal

2005. Graph cuts segmentation using an elliptical shape prior. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, Pp. II–1222. IEEE.

Sommer, C., C. Straehle, U. Koethe, and F. A. Hamprecht

2011. Ilastik: Interactive learning and segmentation toolkit. In *Biomedical Imaging: From Nano to Macro, 2011 IEEE International Symposium on*, Pp. 230–233. IEEE.

Stella, X. Y. and J. Shi

2003. Multiclass spectral clustering. In *ICCV*, Pp. 313–319.

Su, T. and J. G. Dy

2007. In search of deterministic methods for initializing k-means and gaussian mixture clustering. *Intelligent Data Analysis*, 11(4):319–338.

Tang, D., H. Fu, and X. Cao

2012. Topology preserved regular superpixel. In *2012 IEEE International Conference on Multimedia and Expo*, Pp. 765–768. IEEE.

Tang, M., L. Gorelick, O. Veksler, and Y. Boykov

2013. Grabcut in one cut. In *Proceedings of the IEEE International Conference on Computer Vision*, Pp. 1769–1776.

Tapia, E.

2011. A note on the computation of high-dimensional integral images. *Pattern Recognition Letters*, 32(2):197–201.

Tarjan, R. E.

1975. Efficiency of a good but not linear set union algorithm. *Journal of the ACM (JACM)*, 22(2):215–225.

The HDF Group

. Hierarchical Data Format, version 5.

Tighe, J. and S. Lazebnik

2010. Superparsing: scalable nonparametric image parsing with superpixels. In *European conference on computer vision*, Pp. 352–365. Springer.

Top, A., G. Hamarneh, and R. Abugharbieh

2010. Spotlight: Automated confidence-based user guidance for increasing efficiency in interactive 3d image segmentation. In *MVC*.

Top, A., G. Hamarneh, and R. Abugharbieh

2011. Active learning for interactive 3d image segmentation. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2011*, Pp. 603–610.

Truc, P. T., M. A. Khan, Y.-K. Lee, S. Lee, and T.-S. Kim

2009. Vessel enhancement filter using directional filter bank. *Computer Vision and Image Understanding*, 113(1):101–112.

Tu, Z.

2005. Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 2, Pp. 1589–1596. IEEE.

Tu, Z. and X. Bai

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

Uijlings, J. R., K. E. van de Sande, T. Gevers, and A. W. Smeulders

2013. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.

Unger, M., T. Pock, W. Trobin, D. Cremers, and H. Bischof
2008. Tvseg-interactive total variation based image segmentation. In *BMVC*, volume 31, Pp. 44–46. Citeseer.

van Aarle, W., W. J. Palenstijn, J. Cant, E. Janssens, F. Bleichrodt, A. Dabravolski, J. De Beenhouwer, K. J. Batenburg, and J. Sijbers
2016. Fast and flexible x-ray tomography using the astra toolbox. *Optics express*, 24(22):25129–25147.

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

Vazquez-Reina, A., M. Gelbart, D. Huang, J. Lichtman, E. Miller, and H. Pfister
2011. Segmentation fusion for connectomics. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Pp. 177–184. IEEE.

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

Veksler, O.
2008. Star shape prior for graph-cut image segmentation. In *European Conference on Computer Vision*, Pp. 454–467. Springer.

Veksler, O., Y. Boykov, and P. Mehrani
2010. Superpixels and supervoxels in an energy optimization framework. In *European conference on Computer vision*, Pp. 211–224. Springer.

Vese, L. A. and T. F. Chan
2002. A multiphase level set framework for image segmentation using the mumford and shah model. *International journal of computer vision*, 50(3):271–293.

Vicente, S., V. Kolmogorov, and C. Rother
2008. Graph cut based image segmentation with connectivity priors. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, Pp. 1–8. IEEE.

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

Viola, P., M. J. Jones, and D. Snow
2003. Detecting pedestrians using patterns of motion and appearance. In *null*, P. 734. IEEE.

Wadeson, N. and M. Basham

2016. Savu: A python-based, mpi framework for simultaneous processing of multiple, n-dimensional, large tomography datasets. *arXiv preprint arXiv:1610.08015*.

Wang, R., R. L. Stone, J. T. Kaelber, R. H. Rochat, A. M. Nick, K. V. Vijayan, V. Afshar-Kharghan, M. F. Schmid, J.-F. Dong, A. K. Sood, et al.

2015. Electron cryotomography reveals ultrastructure alterations in platelets from patients with ovarian cancer. *Proceedings of the National Academy of Sciences*, 112(46):14266–14271.

Wang, S., H. Lu, F. Yang, and M.-H. Yang

2011. Superpixel tracking. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Pp. 1323–1330. IEEE.

Westenberger, P.

2008. Avizo-3d visualization framework. In *Geoinformatics Conference*, Pp. 1–11.

Weston, J. and C. Watkins

1998. Multi-class support vector machines. Technical report, Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, May.

Weston, J., C. Watkins, et al.

1999. Support vector machines for multi-class pattern recognition. In *ESANN*, volume 99, Pp. 219–224.

Wienert, S., D. Heim, K. Saeger, A. Stenzinger, M. Beil, P. Hufnagl, M. Dietel, C. Denkert, and F. Klauschen

2012. Detection and segmentation of cell nuclei in virtual microscopy images: a minimum-model approach. *Scientific reports*, 2:srep00503.

Wirtz, E., S. Leal, C. Ochatt, and G. M. Cross

1999. A tightly regulated inducible expression system for conditional gene knock-outs and dominant-negative genetics in Trypanosoma brucei. *Molecular and biochemical parasitology*, 99(1):89–101.

Zeng, T., B. Wu, and S. Ji

2017. Deepem3d: approaching human-level performance on 3d anisotropic em image segmentation. *Bioinformatics*, P. btx188.

Zhang, C., J. Yarkony, and F. A. Hamprecht

2014. Cell detection and segmentation using correlation clustering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, Pp. 9–16. Springer.

Zhang, T., R. Ramakrishnan, and M. Livny

1996. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, Pp. 103–114. ACM.

Zhang, Y., R. Hartley, J. Mashford, and S. Burn

2011. Superpixels via pseudo-boolean optimization. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Pp. 1387–1394. IEEE.

Zhu, J., H. Zou, S. Rosset, and T. Hastie

2009. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.