EPFL

Thèse n° 8853

Object Priors for Volumetric Image Segmentation

Présentée le 27 juin 2022

Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Pamuditha Udaranga WICKRAMASINGHE

Acceptée sur proposition du jury

Prof. P. Dillenbourg, président du jury
Prof. P. Fua, directeur de thèse
Dr I. Cohen, rapporteur
Prof. A. Geiger, rapporteur
Dr M. Cantoni, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

Acknowledgements

First of all, I would like to thank my advisor Prof. Pascal Fua for giving me the opportunity to join CVLab and for his guidance through out my PhD. I'm specially grateful for the freedom and support he gave me to explore different avenues of research during the early years of my PhD. I would also like to express my gratitude to the members of my thesis committee; Prof Pierre Dillenbourg, Dr. Marco Cantoni, Dr. Isaac Cohen and Prof Andreas Geiger for their time and effort in evaluating my thesis. I'm honored to have them in my thesis committee.

During my PhD, I had the pleasure of doing an internship at Facebook Reality Labs. I would like to thank Dr. Timur Bagautdinov and Dr. Thomas Simon for giving me the opportunity to do the internship and for their support through out my stay.

It has been a pleasure to work in CVLab. Even though we didn't had much opportunity to meet during last two years due to the Covid pandemic, I would like to thank all the members of CVLab for the good times we shared and the long discussions we had about both academic and non-academic topics.

I first arrived in EPFL back in 2012 as a summer intern at LSP. I would like to thank Prof. Roger Hersch for giving me the opportunity to join LSP and for his support throughout my master's degree at EPFL. I would also like to thank Dr. Ranga Rodrigo from University of Moratuwa, Sri Lanka for introducing me into the world of computer vision and machine learning and for his guidance and encouragement to pursue higher studies.

I have being lucky to have two good friends; Dr. Dasun Perera and Dr. Kasun Fernando that made a significant impact on my academic life. I would like to thank Dr. Dasun Perera for helping me to find my way through the life in research and for all his support during my time at EPFL. I would like to thank Dr. Kasun Fernando for his guidance and support during the early days of my academic life. I would also like to thank all my friends from Lausanne, Helsinki, Oulu, and Sri Lanka that supported me in various ways during this period.

Acknowledgements

I'm eternally grateful to the free education system in Sri Lanka and the tax paying citizens from Sri Lanka and Switzerland that supported my education. Without it, I would not have been able to pursue my passion in science and engineering. I am also grateful to each and every teacher and lecturer from Richmond College, University of Moratuwa and EPFL that taught me during all these years.

My deep gratitude goes to my parents for their continuous support throughout this journey and for the guidance as well as the freedom that was given to me. Not to forget my brother Ruchiranga, who has been there for me always. I would also like thank my extended family for their support. Finally, I'm ever grateful to my loving wife Heshani who had to endure the brunt of my Phd, for always encouraging me to reach greater heights and for her immense strength, support and love.

Lausanne, February 19, 2022 Udaranga Wickramasinghe.

Abstract

Large training datasets have played a vital role in the success of modern deep learning methods in computer vision. But, obtaining sufficient amount of training data is challenging, specially when annotating volumetric images. This is because fully annotating a single image volume require annotating a whole stack of images which is costly. Moreover, majority of volumetric images originate from niche domains that require expert knowledge to annotate and it further increase the associated cost. Therefore, designing models for volumetric image segmentation that can work with small training datasets is of great importance.

One way to address this challenge is to use prior knowledge in segmentation model design. There are many forms of prior knowledge and in this thesis, we focus on using object priors which defines prior knowledge related to the properties of the object being segmented. We begin by proposing an end-to-end differentiable architecture that produce surface meshes from input image volumes which enable easy integration of object priors related to topology and surface properties. With that we demonstrate how the priors help achieve better accuracy and quality in predictions compared to state-of-the-art methods that do not use them. Then we focus on surface priors and propose a methodology based on Active Surface models to achieve better trade-offs between the accuracy of the surfaces produced by the segmentation models and their quality. Afterwards, we apply the Active Surface model proposed earlier to develop an interactive annotation tool that significantly reduce the effort necessary to annotate image volumes. Along with that, a volumetric image segmentation model that best utilize annotations produced with the tool is also proposed. Finally we focus on object priors related to overall shape. We use Probabilistic Atlases (PAs) to introduce this prior knowledge into the segmentation model and demonstrate its ability to improve the accuracy in predictions.

Overall, we use object prior to improve the accuracy, quality and the speed of annotation in volumetric image segmentation. To demonstrate their effectiveness, we use volumetric images produced by Magnetic Resonance Imaging (MRI), Computed Tomography (CT) and Electron Microscopy (EMs).

Abstract

Keywords: Volumetric Images, Segmentation, Deep Learning, Object Priors, Prior Knowledge, Surface Meshes, Computer Vision

Résumé

Les grandes bases de données d'entraînement ont joué un rôle essentiel dans le succès des méthodes modernes d'apprentissage profond en vision par ordinateur. Cependant, il est difficile d'obtenir une quantité suffisante de données d'entraînement, en particulier lorsqu'elles requièrent l'annotation d'images volumétriques. En effet, l'annotation complète d'un seul volume d'images nécessite l'annotation de toute une pile d'images, ce qui est coûteux. De plus, la majorité des images volumétriques proviennent de domaines de niche qui nécessitent des connaissances d'experts pour être annotées, ce qui augmente encore le coût associé. Par conséquent, la conception de modèles de segmentation d'images volumétriques pouvant fonctionner avec de petites bases de données d'apprentissage est d'une grande importance.

Une façon de relever ce défi consiste à utiliser les connaissances préalables (a priori) dans la conception de modèles de segmentation. Il existe de nombreuses formes de connaissances a priori et dans cette thèse, nous nous concentrons sur l'utilisation des a priori relatifs aux propriétés de l'objet segmenté. Nous commençons par proposer une architecture différentiable de bout en bout qui produit des maillages de surface à partir de volumes d'images, et qui permet une intégration facile des a priori liés à la topologie et aux propriétés de surface de l'objet reconstruit. Avec cela, nous démontrons comment les a priori aident à obtenir une meilleure précision et qualité dans les prédictions par rapport aux méthodes de pointe les plus récentes qui ne les utilisent pas. Ensuite, nous nous concentrons sur les a priori portant sur la surface de l'objet, et proposons une méthodologie basée sur les modèles de surface active pour obtenir de meilleurs compromis entre la précision des surfaces produites par les modèles de segmentation et leur qualité. Puis nous appliquons le modèle de surface active proposé précédemment pour développer un outil d'annotation interactif qui réduit considérablement l'effort nécessaire pour annoter des volumes d'images. Parallèlement à cela, un modèle de segmentation d'image volumétrique qui utilise au mieux les annotations produites avec l'outil est également proposé. Enfin, nous nous concentrons sur les a priori des objets liés à la forme globale. Nous utilisons des atlas probabilistes (AP) pour introduire ces connaissances préalables dans le modèle de segmentation et démontrer sa capacité à améliorer la précision des prédictions.

Résumé

Dans l'ensemble, nous utilisons des connaissances a priori portant sur les objets reconstruits pour en améliorer la précision, la qualité et la vitesse d'annotation dans la segmentation d'images volumétriques. Pour démontrer leur efficacité, nous utilisons des images volumétriques produites par imagerie par résonance magnétique (IRM), tomodensitométrie (CT) et microscopie électronique (EM).
Mots-clés : Images volumétriques, Segmentation, Apprentissage profond, A priori de forme, Connaissances préalables, Maillages surfaciques, Vision par ordinateur

Contents

Acknowledgements i

Abstract (English) iii

Abstract (French) v

List of Figures xi

List of Tables xv

1 Introduction 1

1.1 Prior Knowledge for Volumetric Image Segmentation 1

1.2 Shape Representations and Object Priors 3

1.2.1 Contribution 4

2 Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data 7

2.1 Introduction 7

2.2 Related Work 8

2.2.1 Importance of Surface Models 8

2.2.2 Deformable Models 9

2.2.3 Deep Surfaces 9

2.3 Method 9

2.3.1 Mesh Decoder 10

2.3.2 Loss function 12

2.4 Experiments 12

2.4.1 Datasets 12

2.4.2 Baselines 13

2.4.3 Comparative Results 14

2.4.4 Ablation Study 15

2.5 Conclusion 15

3 Deep Active Surface Models **17**
3.1 Introduction 17
3.2 Related Work 19
3.2.1 Active Contour and Surface Models 19
3.2.2 Deep Surface Models 19
3.2.3 Deep Contour Models 20
3.2.4 Active Contours/Surfaces and Neural Networks 20
3.3 Method 21
3.3.1 Active Surface Model (ASM) 21
3.3.2 Deep Active Surface Model (DASM) 24
3.4 Experiments 26
3.4.1 From 2D Images to 3D Surfaces 27
3.4.2 From 3D Image stacks to 3D Surfaces 29
3.5 Conclusion 32

4 Weakly Supervised Volumetric Image Segmentation with Deformed Templates **33**
4.1 Introduction 33
4.2 Related Work 34
4.2.1 Weakly-Supervised Image Segmentation 35
4.2.2 Template Based Approaches 36
4.2.3 Image Reconstruction 36
4.3 Method 37
4.3.1 Network Architecture and Losses 37
4.3.2 Template Deformation 39
4.4 Experiments 40
4.4.1 Datasets 40
4.4.2 Metrics and Baseline 41
4.4.3 Providing Annotations 41
4.4.4 Limitations 46
4.4.5 Ablation Study 46
4.5 Conclusion 48

5 Probabilistic Atlases to Enforce Topological Constraints **49**
5.1 Introduction 49
5.2 Related Work 50

5.3 Approach 51
5.3.1 Network Architecture 51
5.3.2 Atlas Design 52
5.3.3 Atlas Registration 52
5.3.4 Loss Functions 53
5.4 Results and Discussion 53
5.4.1 Datasets 53
5.4.2 Quantitative and Qualitative Results 54
5.5 Conclusion 55

6 Conclusion **57**
6.1 Summary 57
6.2 Future Work 58

A Appendix - Deep Active Surface Models **61**
A.1 Appendix 61
A.1.1 Active Surface Model: Continuous Formulation 61
A.1.2 Active Surface Model: Discrete Formulation 61
A.1.3 Matrix Inversion using Neumann Series 63
A.1.4 Quantitatively Measuring Mesh Regularization with Consistency Metrics 64

Bibliography **65**

Curriculum Vitae **75**

List of Figures

2.1 **Architectures** (a) The **Pixel2Mesh-3D** architecture, a straightforward extension of [98], uses a surface decoder but no voxel decoder. (b) By contrast, our **Voxel2Mesh** architecture takes as input an image and spherical mesh. They are jointly encoded and then decoded into cubes and meshes of increasing resolution. At each mesh decoding stage, the decoder first receives as input the current mesh and a set of features sampled from the cube of corresponding resolution. Then the mesh is deformed and refined non-uniformly by adding vertices only where they are needed. 8

2.2 **Approach.** (a) Learned Neighborhood Sampling. (b) Adaptive Mesh Unpooling. . . . 10

2.3 **Qualitative results.** (a) Input volumes. EM (row 1,2), CT(row 3), MRI(row 4) (b) Ground truth (c) CNN baseline (d) CNN baseline + post processing (e) **Voxel2Mesh**. The orange boxes highlight false positive regions. 13

2.4 **Levels of resolution.** (a) **Pixel2Mesh-3D** result (10422 vertices). (b) **Voxel2Mesh** result (7498 vertices). With our adaptative unpooling, we obtain better results with fewer vertices. 14

3.1 **Smoothness and Accuracy.** (Top) 3D surface meshes of a couch modeled from an RGB image and of a synaptic connection segmented from an electron microscopy stack by Mesh R-CNN [38] and by Voxel2Mesh [103], two state-of-the-art mesh-generating methods. (Bottom) Results using the same backbones augmented by our DASM smoothing layers. The meshes have far fewer artifacts and we will show that they are also more accurate. 18

3.2 **Derivatives. Top.** The surface is represented by a differentiable mapping v from $\mathbb{R}^2$ to $\mathbb{R}^3$. **Bottom.** After discretization, let $v(s,r)$ be a vertex. To approximate derivatives with respect to s using finite differences, we need to estimate quantities such as $v(s+\delta s, r)$ where δs is small. To this end, we estimate the barycentric coordinates λ, λ_1, and λ_2 of $v(s+\delta s, r)$ in the facet it belongs to and take $v(s+\delta s, r)$ to be $\lambda v(s,r) + \lambda_1 v(s_1, r_1) + \lambda_2 v(s_2 + r_2)$, where $v(s_1, r_1)$ and $v(s_2, r_2)$ are the other two vertices of the facet. The same operation can be performed for derivatives with respect to r. 22

3.3 **Uniform vs Adaptative Smoothing.** (a) Mesh at time t. In general, $|\mathbf{B}\Gamma^t|_q < |\mathbf{B}\Gamma^t|_p$. (b) When using enough uniform smoothing to remove the irregularity at point p, the mesh will typically be oversmoothed at q. (c) When using adaptative smoothing, the mesh is smoothed around p but not oversmoothed around q. 25

3.4 **Performing multiple smoothing steps.** (a) Ground truth (b) Input (c) ASM 1-step (d) ASM 3-steps (e) ASM 5-steps (f) ASM 7 steps. We start with a very noisy mesh produced by the algorithm of [38] run without regularization. We have colored front-faces and back-faces of the meshes in blue and pink respectively for better visualization. After running one step of surface evolution given by Eq. 3.6, we obtain the mesh of Fig. 3.4 (c). Subsequent meshes shown in Fig. 3.4 (d,e,f) are obtained by continuing the surface evolution for 3, 5, and 7 steps respectively. In these subsequent steps we set $F(\Phi^{t-1})$ to zero. 26

3.5 **ShapeNet Results.** (a) Input images (b) Mesh R-CNN results from two different viewpoints. The orange arrows highlight commonly seen Mesh R-CNN artifacts. (c) DASM results in the same two views. The meshes are much smoother and most artifacts have disappeared, except for a few highlighted by blue arrows. 29

3.6 **Influence of the regularization term.** As λ_{reg} increases, the output of both methods becomes smoother but only DASM completely eliminates the artifacts. Note that in the case $\lambda_{reg} = 0$, the output of Mesh R-CNN is an extremely irregular mesh that nevertheless scores well on the Chamfer metric. 30

3.7 Modeling Synapses from Electron Microscopy Image Stacks (a) Representative slice from the input volume. (b) **Voxel2Mesh** results seen from two different views. (c) DASM results seen from the same two views. The pre-synaptic region, post-synaptic region, and synaptic cleft are shown in blue, green, and red, respectively. The DASM results are much smoother and without artifacts, while also being more accurate. . . . 31

4.1 **Weak-Net** architecture. A first U-Net takes the image $\mathbf{X}$ as input and outputs a segmentation $\hat{\mathbf{Y}}$, which is in turn fed to a second U-Net that outputs a reconstructed image $\hat{\mathbf{X}}$. Training is achieved by jointly minimizing L_{mse} and L_{ce}. This encourages $\hat{\mathbf{X}}$ to resemble the original image and $\hat{\mathbf{Y}}$ to be similar to $\mathbf{Y}$, a roughly aligned version of a template. 34

4.2 **Deformed templates.** (a,b,c,d,e) Deformed template using $N = 25, 50, 125, 250$, and 3661 user-supplied 3D points when segmenting a liver. $N = 3661$ corresponds to full annotations in all slices. 34

4.3 **Using a secondary task to improve segmentations**. (a) **X** is the input image and $\mathbf{Y}_{true}$ is the ground-truth segmentation. **Y** is the atlas that is roughly correct but in which the boundaries of the target object are inaccurate. (b) When λ is too large, $\mathscr{L}_{mse}$ dominates and the resulting segmentation features a spurious region shown in red that corresponds to image boundaries that are *not* those of the target structure. When λ is zero, only $\mathscr{L}_{ce}$ is minimized and minimizing produces boundaries that are also those of the atlas, which are not at the right place as the denoted by the red lines. For appropriate values of λ, the boundaries are correct and the spurious region is eliminated. 37

4.4 Impact of the λ parameter in the loss of Eq. 4.1 and the thresholding parameter γ. (Col. 2-4) When $\lambda = 0$, the segmentation is very similar to the template. When λ is large, there are many spurious values. (Col. 5-8) We set λ to 10^{-4} and vary γ. 38

4.5 **Iterative annotation strategy**. The annotator provides a few 3D points, which are used to deform the template. The result is rasterized and overlaid on the images. The annotator can then add more points and deform again until the deformed template is close enough to the desired shape. For the hippocampus, shown here, this takes about 150 points to achieve 80% of the maximum accuracy that could be obtained using a fully annotated ground truth which consist of 1509 points on average. 39

4.6 **Annotation GUI** A user can place points by clicking in the 2D image cross sections. A template is then deformed to fit the points and the user can place corrective clicks. The fitting takes less than one second so the process can be done interactively. 40

4.7 **Annotation simulation on the Liver dataset** (a) Reconstruction accuracy (IoU) variation for different values of K and P. (b) Reconstruction accuracy (IoU) variation as a function of the number of annotated points. The black-dashed line indicates the accuracy that would be achieve by annotating all points in each sample, that is 3661 points per sample on average for the liver dataset. 43

4.8 **Segmentation results.** (a+f) A slice from the input volume from top to bottom (b+g) Ground truth (c+h) U-Net with full-supervision (d+i) Corresponding Baseline (e) **Weak-Net** trained with 25 points per sample. (f) **Weak-Net** trained with 125 points per sample. 43

4.9 **Results on the hippocampus dataset.** (a) **Baseline** (b) **Weak-Net** (c) Annotation effort required to achieve 65% and 70% IoU. The blue and red arrows indicate the effort reduction our approach delivers. 44

4.10 **IoU as function of the number of points used (N).** (a) Synaptic Junction dataset. (b) Liver dataset. The dashed line denotes fully supervised results. 45

4.11 **Template accuracy** as a function of the number of human or simulated point annotations (N). (a) Hippocampus. (b) Liver. 46
4.12 **Influence of λ and γ**. IoU (%) as a function of λ in (a), γ in (b) on the Synaptic Junction dataset with $N = 25$. 47

5.1 Eliminating topological mistakes. (a) A small FIBSEM image stack featuring a synaptic junction and a retinal fundus image. (b) Ground-truth segmentations. The pre-synaptic, synaptic-cleft, and post-synaptic regions shown in green, red, and blue respectively. Similarly, the optic disk and cup shown in green and red, respectively. (c) U-Net segmentations with orange arrows pointing to topological mistakes. (d) Our error-free result (best viewed in color). 50
5.2 **PA-Net** architecture. The network takes as input an image and a list of PAs. Given the decoder output, the fully connected layers estimate the parameters of the affine transformation that registers the PAs to the input. The registered PAs are then concatenated with the features computed by the convolutional decoder at each scale. 51
5.3 Probabilistic atlases. (a) For synaptic junction segmentation, they are cubes and we show a single face. (b) For optic optic disk and cup segmentation, they are regular 2D arrays. The color hues denote the same areas as those in Fig. 5.1. The color saturation is proportional to the probability at a given location (best viewed in color). 52
5.4 U-Net vs **PA-Net**. Top three rows depict results from synaptic junction segmentation. The bottom three rows depict results from retinal fundus image segmentation. (a) Input images, (b) Ground-truth, (c) U-Net results (d) **PA-Net** results. The color hues denote the same areas as those in Fig. 5.1 (best viewed in color). 56

A.1 **Finite Differences.** Relative positions of $(s + k_1\delta, r + k_2\delta)$ terms w.r.t (s, r) which is at the center and its 1-ring neighbors from degree 4 to 10. 62
A.2 **Neumann Series Approximation.** RMSE between the approximated inverse and the true one in blue and computation time in red as function of K. $K = 4$ gives an acceptable trade-off between the two. 63

List of Tables

2.1 Comparative results on three datasets using the IoU metric. 15
2.2 Comparative results against CNN based mesh deforming baselines. 15

3.1 **Comparative results on ShapeNet.** 28
3.2 **Results on ShapeNet** as a function of λ_{edge}. 28
3.3 **Ablation study on ShapeNet.** 28
3.4 **Comparative results on CortexEM.** We use the IoU metric to compare volumetric segmentations. 31
3.5 **Comparative results on CortexEM.** We use the Chamfer distance ($\times 10^{-2}$) 31
3.6 **Run times.** We report the time required to perform a forward and backward pass. $|V|$ is the number of mesh vertices. 32

4.1 **Human annotation results.** Values are given as mean±std. Times are in minutes. 45
4.2 Comparative results 47

5.1 Comparative results for synaptic junction segmentation. 53
5.2 Comparative results for fetinal fundus image segmentation. 54

A.1 **Results on ShapeNet** as a function of λ_{edge}. Note that this is an extension of Table 3.2. 64

1 Introduction

Volumetric image segmentation is one of the sub-domains in computer vision which has the potential to make significant real world impact. Its objective is to segment 3D structures in volumetric image data. Even though the problem landscape and the approaches to solve it has many similarities with 2D image segmentation, it has one unique challenge; that is the difficulty in obtaining large training datasets. Compared to other computer vision tasks like image classification or object detection, annotating images for image segmentation is time consuming and difficult, even in 2D images. When it comes to volumetric images, the problem become more difficult since we have to annotate a whole image stack to obtain a single annotated sample. Furthermore, most of the modern day volumetric image data originate from medical imaging techniques such as Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) or laboratories that use imaging devices such as Electron Microscopes (EMs). As a result, annotating volumetric image data requires export knowledge and it adds another layer of difficulty to the process. Therefore, it is vital to develop methods for volumetric image segmentation that can work with small training datasets and use of prior knowledge in model design is one of the common approaches to address this challenge.

1.1 Prior Knowledge for Volumetric Image Segmentation

Within the context of machine learning, *prior knowledge* refers to all the information available about a problem in addition to the training data [87]. Based on this definition, we can identify many forms of prior knowledge. Here we list some of the mostly used forms of prior knowledge and how they are used in state-of-the-art volumetric image segmentation model design.

- **Object Properties**: Prior knowledge about the object being segmented is introduced by identifying properties of the object. They can be divided in to two categories as global and local properties. Information about overall shape and topology of the structure is considered as global properties where as information about local regions is considered as local properties. Together, we refer to them as object priors.

  Prior knowledge on overall shape is commonly introduced via atlases [112, 60, 109, 22, 94]. They are made by averaging multiple segmented objects to obtain a mean shape. They can take the form of a binary atlas [51] or a probabilistic atlas (PAs) [92, 93, 112, 60]. Atlases are commonly used in volumetric image segmentation because they are mostly acquired through non-projective imaging techniques such as MRI, CT or EM. If images are captured using a device like the standard camera that employs projective techniques, application of the prior information become challenging. The atlases are commonly incorporated by using them as inputs similar to the input image volumes [92, 93] or learning to deform the atlases to fit the target object in the input image volume [22].

  Topology is not as descriptive as shape priors, but still contains useful information about an object. It is possible to introduce both geometric and network topology of a structure as prior knowledge. Genus of an object defines the geometric topology while connectivity of parts in an object which consist of multiple parts (i.e part image segmentation) defines the network topology. All the atlas based approaches that introduce prior knowledge related to the shape also introduce prior knowledge about the topology.

  Local properties define prior information about a local region which is almost consistent throughout the object. This include volume based prior knowledge such as voxels with similar intensity belonging to the same class and surface based prior knowledge such as surface smoothness, tension, and skewness. Volume related priors are commonly introduced in to CNNs by incooperating probabilistic graphical models with CNNs [58, 111] or introducing more complex loss terms [12]. Surface related priors are commonly introduced using shape regularizing loss terms [98].

- **Transformation Invariance**: In image segmentation, transformation invariance refers to the detection of low-to-high level features of objects of interest irrespective of its position, orientation, scale and color/intensity variations in the input image. Incorporation of these forms of prior knowledge can be widely seen in start-of-the-art image segmentation methods. For instance, in standard Convolutional Neural Networks (CNNs) used for segmentation like the U-Net [19], convolutional layers are used because of the prior knowledge on translation invariance of the features. The choice of convolutional layers over fully connected layers based on this prior knowledge significantly reduce the computations

needed and make the modern day CNNs a reality. Similarly, pooling layers enable the model to learn scale invariance, even though the primary objective of the layer is feature aggregating across space. Rotational invariance is not incorporated in standard CNNs. But various attempts have been made to introduce rotational invariance as well in to convolutional layers [16].

Similarly, prior knowledge on transformation invariance is used during training when deciding data augmentation methods. This enable the network to learn these invariances. Random translation and scaling is used as a common data augmentation technique. When performing rotations, 90^o rotations and mirroring is commonly used. To introduce invariance to color transformations, various types color augmentations are performed.

- **Data Distribution**: Data distribution captures the prior knowledge on the nature of the data distribution. Bayesian learning brings prior knowledge on data distribution into modeling and it has been adapted in to deep learning as Bayesian deep learning [35].

All these types of prior knowledge opens up countless opportunities to use prior knowledge in model design. In this thesis we focus on the use of object priors for volumetric image segmentation. We use them to design state-of-the-art deep learning models and use them to improve the accuracy, quality and the speed of annotation in volumetric image segmentation.

1.2 Shape Representations and Object Priors

One of the critical factors that determine the ease of which object priors can be introduced is the type of shape representation used in the segmentation model. Therefore, it is vital to choose the appropriate shape representation depending on the form of prior knowledge that is being incorporated. Here we list most widely used types of shape representations and how they support the incorporation of different forms of object priors.

- **Voxel Representations:** Since input image volumes use voxel representation to represent data, voxel representation is the most widely used shape representation. Typical models based on this representation use 3D CNNs and since there are no changes in the representation within the model, the architectures are relatively simpler [19, 73, 49]. Atlases are commonly used to introduce object priors related to shape and topology when using voxel representations [112, 60, 109, 22, 94]. But introducing priors on surface properties is not trivial with voxel representation. Additionally, introduction of new layers

in to 3D CNNs [41], complex loss functions [12, 107] and integration with graphical models [4, 5] can be identified as attempts to incorporate different forms of object priors when working with voxel representation.

- **Surface Mesh Representation:** Surface meshes explicitly models the surface of the target object. Therefore, it can be easily used to introduce priors related to topology, shape and surface properties. One of the key challenges is performing the shift from voxel representation into surface mesh representation. It could be addressed using differentiable marching cube algorithms [84] or a mesh deformation based approach [98, 38].
- **Implicit Surface Representations:** Recent work has proposed the use of implicit surfaces for representing 3D shapes. While one class of implicit surfaces learn the occupancy fields [72], the other learns distance fields [80, 97]. Even though occupancy fields has the same limitation with introducing surface priors as with voxel representations, distance fields has demonstrated its ability to incorporate surface priors similar to surface meshes [90]. But, one disadvantage with implicit surfaces is that, in applications that require fast extraction of an explicit surface (eg: interactive annotation tool), extraction of a the surface from the implicit field adds an additional computational overhead compared to using surface mesh representation.
- **Other Representations:** They include point clouds, octrees, and volumetric meshes. Point clouds are not used in volumetric image segmentation since it fails to represent the surface of an object fully. Octrees and volumetric meshes do not have this drawback, but still they are not used since the models that use them fails to achieve superior performance compared to other competing shape representations.

1.2.1 Contribution

As mentioned above, in this thesis we focus on introducing object priors for volumetric image segmentation. Depending on the type of object priors we want to introduce, we choose appropriate shape representation and propose an ensemble of methods that address different aspects in volumetric image segmentation. They are,

- **Voxel2Mesh** (see Chapter 2) that introduce priors on geometric topology and surface properties such as smoothness and tension to produce more accurate segmentations,
- **DASM** (see Chapter 3) that introduce an alternative mechanism to enforce surface properties such as tension, smoothness, and skewness on surface meshes produced by

graph convolutional neural networks to achieve better trade-offs between the accuracy and the quality of the meshes produced by them,

- **Weak-Net** (see Chapter 4) that use prior knowledge on surface properties and topology to obtain weak annotations of target objects and use them to train CNNs for volumetric image segmentation, and
- **PA-Net** (see Chapter 5) that introduce prior knowledge on approximate overall shape using PAs to produce more accurate segmentations,

Collectively, these approaches improve the accuracy, quality and the speed of annotation in volumetric image segmentation.

2 Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data

2.1 Introduction

State-of-the-Art volumetric segmentation techniques rely on Convolutional Neural Networks (CNNs) that use voxel representation [19, 73, 88]. However in clinical and research practice, a mesh representation is often required to model the surface morphology and to compute area-based statistics. Furthermore, introducing surface priors with voxel representation is not trivial and priors related to shape and topology can be easily introduced using a mesh deforming approach.

We therefore introduce an end-to-end trainable architecture that goes from volumetric images to 3D surface meshes by learning to deform a template mesh to fit the target object. Our **Voxel2Mesh** architecture is depicted in Fig. 2.1 (b). It comprises a voxel encoder, voxel decoder and a mesh decoder. The two decoders communicate at all resolution levels and our approach incorporates two innovative features that are key to performance.

- **Learned Neighborhood Sampling.** The mesh decoder learns to sample the output of the volume decoder only where needed, that is, in the neighbor- hood of output vertices.
- **Adaptive Mesh Unpooling.** Accurately representing the surface requires densely sampled mesh vertices in high-curvature areas but not elsewhere. We introduce an adaptative mesh unpooling scheme that achieves this results and eliminates the need for exponentially large amounts of memory that uniform unpooling requires.

Our contribution therefore is a novel architecture that takes a 3D volume as input and yields an accurate 3D surface without any post-processing. We evaluate it on Electron Microscopy and

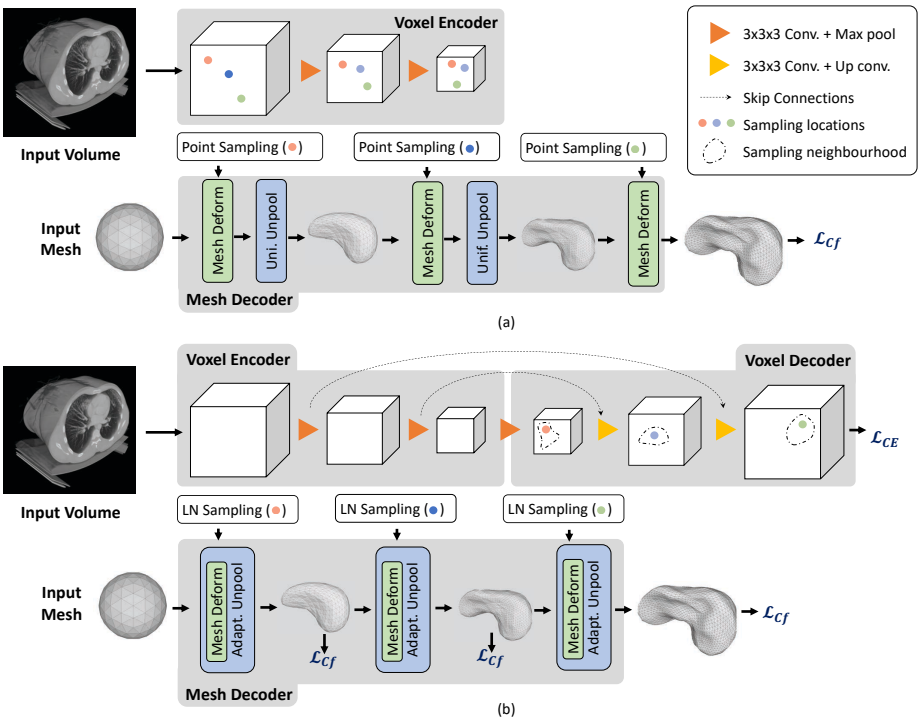


Figure 2.1: **Architectures** (a) The **Pixel2Mesh-3D** architecture, a straightforward extension of [98], uses a surface decoder but no voxel decoder. (b) By contrast, our **Voxel2Mesh** architecture takes as input an image and spherical mesh. They are jointly encoded and then decoded into cubes and meshes of increasing resolution. At each mesh decoding stage, the decoder first receives as input the current mesh and a set of features sampled from the cube of corresponding resolution. Then the mesh is deformed and refined non-uniformly by adding vertices only where they are needed.

MRI brain images as well as CT liver scans. We will show that it outperforms state-of-the-art segmentation methods, especially when the training set is relatively small.

2.2 Related Work

CNN-based volumetric methods such as U-Net and its variants [19, 49, 73, 88] now dominate biomedical image segmentation. This is evident from the CHAOS challenge [55] and Medical Segmentation Decathlon [50] results. The winners of both competitions used ensembles of methods relying on volumetric CNNs.

2.2.1 Importance of Surface Models

Many biological imaging tools are designed to study the morphology of structures such as cells, organs, or tissues. Researchers usually prefer to visualize them as 3D surfaces at the required

level of detail everywhere and are not limited by the voxels resolution. Even though volumes and distances can be estimated using voxels, analyzing surfaces is best accomplished using meshes.

But, state-of-the-art methods produce volumetric descriptions and therefore must be converted to surface meshes. This conversion typically relies on algo- rithms such as Marching Cubes [67] followed by mesh smoothing. This introduces artifacts and prevents end-to-end training. We now turn to existing approaches that mitigate these difficulties.

2.2.2 Deformable Models

Deformable surface models became popular in the 1990s to model biological structures in volumetric data [43, 71] and are still being developped [52, 63, 82]. They are now used in conjunction with deep networks [69] trained to return the energy function the deformed models should minimize.

While they can remove some of the artifacts introduced by converting volumes to surfaces, their use makes the processing pipeline more complex and still prevents end-to-end training. Furthermore, they are not well-suited to segmenting structures that exhibit high inter-sample shape variations, such as the synaptic junctions used in our experiments.

2.2.3 Deep Surfaces

In this context, the Pixel2Mesh [98] approach and its more recent variants [79, 100] are of particular interest. They are among the very few approaches that go *directly* from 2D images to 3D surface meshes without resorting to an intermediate stage. They take an image and encodes it into a set of progressively smaller feature maps. This set is then used at each stage of the decoding process to produce an increasingly accurate mesh. This approach is easily extended to handle 3D image volumes using the architecture depicted by Fig. 2.1(a) and we will refer to this extension as **Pixel2Mesh-3D**. Unfortunately, as we will see, it was conceived for a different purpose and its design choices makes it suboptimal for handling 3D image volumes.

2.3 Method

Our **Voxel2Mesh** architecture is depicted by Fig. 2.1(b). It takes an image volume as input and returns a 3D surface mesh. The image volume is first encoded into smaller latent volumes that serves as input to the *voxel decoder*. Then the voxel decoder generates a pyramid of cubes of

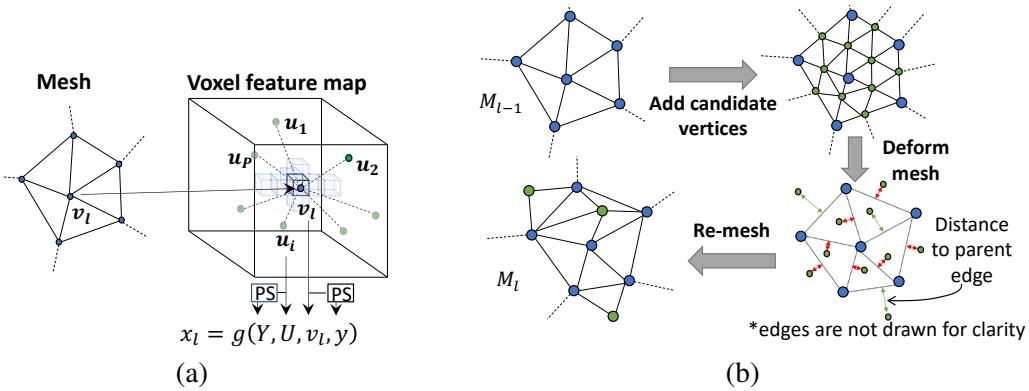


Figure 2.2: **Approach.** (a) Learned Neighborhood Sampling. (b) Adaptive Mesh Unpooling.

increasing resolution whose voxels contain feature vectors. Finally the *mesh decoder* generates increasingly precise deformations of an initial spherical mesh using feature vectors extracted from voxel decoder. The voxel encoder and voxel decoder pictured at the top of Fig. 2.1(b) are based on a standard U-Net [19] architecture.

A key specificity of our approach is that both the sampling of the voxel features and the location and number of the new vertices needed to refine the mesh are adaptive so that the final mesh is refined where it needs to be, and only there.

2.3.1 Mesh Decoder

The input to the mesh decoder is a sphere mesh with 3D vertices forming facets whose edges we use to perform graph convolutions. It learns to iteratively refine the sphere mesh to match the target object.

Let us denote by $l = 0$ the input to the decoders and by $1 \leq l \leq L$ the output of subsequent blocks. For each mesh vertex, we write

$$\mathbf{z}_l = h_l(\mathbf{x}_l, \mathbf{z}_{l-1}, \mathbf{v}_{l-1}) \text{ and } \mathbf{v}_l = \mathbf{v}_{l-1} + \Delta_l(\mathbf{z}_l)\,, \tag{2.1}$$

where $\mathbf{v}_l$ are the 3D vertex coordinates after block l; $\mathbf{x}_l$ and $\mathbf{z}_{l-1}$ are the feature vectors produced by blocks l and $l-1$ in the voxel and mesh decoder, respectively; h_l and Δ_l are two functions implemented by 4 graph convolution layers each, whose weight we learn during training. By convention, we take $\mathbf{z}_0$ to be an empty feature vector. We write the graph convolutions as

$$\mathbf{f}' = w_1\mathbf{f} + \frac{1}{|\mathcal{N}(\mathbf{v}_l)|}\sum_{\mathbf{v}_l^i \in \mathcal{N}(\mathbf{v}_l)} \mathbf{f}^{\,i} w_2 e^{-d_i^2/\sigma^2}\,, \tag{2.2}$$

where $\mathbf{f}'$ and $\mathbf{f}$ are the feature vector associated with the vertex $\mathbf{v}_l$ before and after the convolution. $\mathcal{N}(\mathbf{v}_l)$ is the set of neighborhood vertices of $\mathbf{v}_l$ and $\mathbf{y}^{\,i}$ is the feature vector corresponding to the neighboring vertex $\mathbf{v}_l^i$. $d_i = \sqrt{||\mathbf{v}_l^i - \mathbf{v}_l||}$. $\mathbf{w}_1, \mathbf{w}_2$ and σ are weights learned during training.

Learned Neighborhood Sampling (LNS)

The feature vector x_l in Eq. 2.1 is extracted from voxel features by feature sampling at locations that are functions of the mesh vertices. Since voxel features lie on a discrete grid, we use tri-linear interpolation to sample features and refer to this as *point sampling*. Current approaches only sample at exact vertex locations [98] or in pre-determined neighborhoods around the vertex [100]. This restricts the sampler's ability to pool information from its neighborhood.

Instead, we introduce our LNS strategy that learns optimum sampling loca- tions. It first samples feature vector $\mathbf{y}$ at a given vertex $\mathbf{v}_l$ using point sampling. We then train a neural function to return the set of neighborhood points to sample

$$\mathbf{U} = \{u_i\}_{i=1}^{P} = f(\mathbf{y}, \mathbf{v}_l)\,. \tag{2.3}$$

Next, the set of features $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{P}$ are sampled at $\{u_i\}_{i=1}^{P}$, again using point sampling. Finally, the feature vector $\mathbf{x}_l$ corresponding to vertex $\mathbf{v}_l$ is given by another neural function

$$\mathbf{x}_l = g(\mathbf{Y}, \mathbf{U}, \mathbf{y}, \mathbf{v}_l)\,. \tag{2.4}$$

As shown in Fig. 2.2 (a), LNS only samples from the voxel feature map corres- ponding to the mesh deformation stage. This is in contrast to earlier sampling strategies, that samples from all feature maps at each stage which yields graph convolution networks with many more weights in their mesh deforming module and thus over-fitting when trained with smaller datasets.

Adaptive Mesh Unpooling

High accuracy requires enough vertices to pro- perly fit the underlying surface. We could start with a sphere with many vertices but this is neither computationally nor memory efficient. Therefore, **Pixel2Mesh-3D** and its variants use an uniform unpooling strategy that gradually increase the vertex count. Unfortunately, the vertex count still increases exponentially.

To prevent this, we introduce the adaptive unpooling strategy depicted by Fig. 2.2 (b). First, we add candidate vertices using uniform unpooling strategy. Then the mesh is deformed and we compute the shortest distance from each candidate vertex to its parent edge as indicated by red and green arrows in Fig. 2.2 (b). If the distance is greater than a threshold, we keep them, otherwise we discard them. This looses edge connectivity making re-meshing necessary. To this end, we exploit the fact that the mesh decoder learns a continuous mapping from the surface points on the input sphere to those on the object surface. This relationship enables us to find the corresponding points on the sphere's surface for each mesh vertex. We compute the convex hull to restore edge connectivity $\mathscr{E}'$ between the points on the sphere. $\mathscr{E}'$ can then be directly transferred to the target mesh because the mapping is continuous. Since our remeshing only recomputes the neighbors of each vertex and does not perform any non-differentiable operations on vertex variables, it preserves overall differen- tiability.

2.3.2 Loss function

We use the cross entropy loss $\mathscr{L}_{ce}$ with ground-truth volumes and the Chamfer distance $\mathscr{L}_{cf}$ to points at the boundary of the same ground-truth volumes to train the voxel and mesh decoders, respectively. Instead of using mesh vertices when evaluating $\mathscr{L}_{cf}$, we randomly sample points from the 3D mesh [78]. We also introduce three regularization terms normal loss $\mathscr{L}_n$, Laplacian loss $\mathscr{L}_{lap}$, and edge length loss $\mathscr{L}_{el}$ to improve convergence and smooth the output mesh [98]. We write the complete loss as

$$\mathscr{L} = \sum_{l=1}^{L} \mathscr{L}_{cf}^{l} + \lambda_1 \mathscr{L}_{ce} + \lambda_2 \mathscr{L}_n + \lambda_3 \mathscr{L}_{lap} + \lambda_4 \mathscr{L}_{el} \,, \tag{2.5}$$

where L is the number of stages in the mesh decoder.

2.4 Experiments

2.4.1 Datasets

We tested our approach on 3 datasets. We describe them below briefly and provide more details on the training and testing splits in the supplementary material.

Synaptic Junction Dataset. åIt comprises a $500 \times 500 \times 200$ FIB-SEM image stack of a mouse cortex. We extracted 13 volumes roughly centered around a synapse for training and 13 for testing. The task is to segment the pre-synaptic region, post-synaptic region, and synaptic cleft.

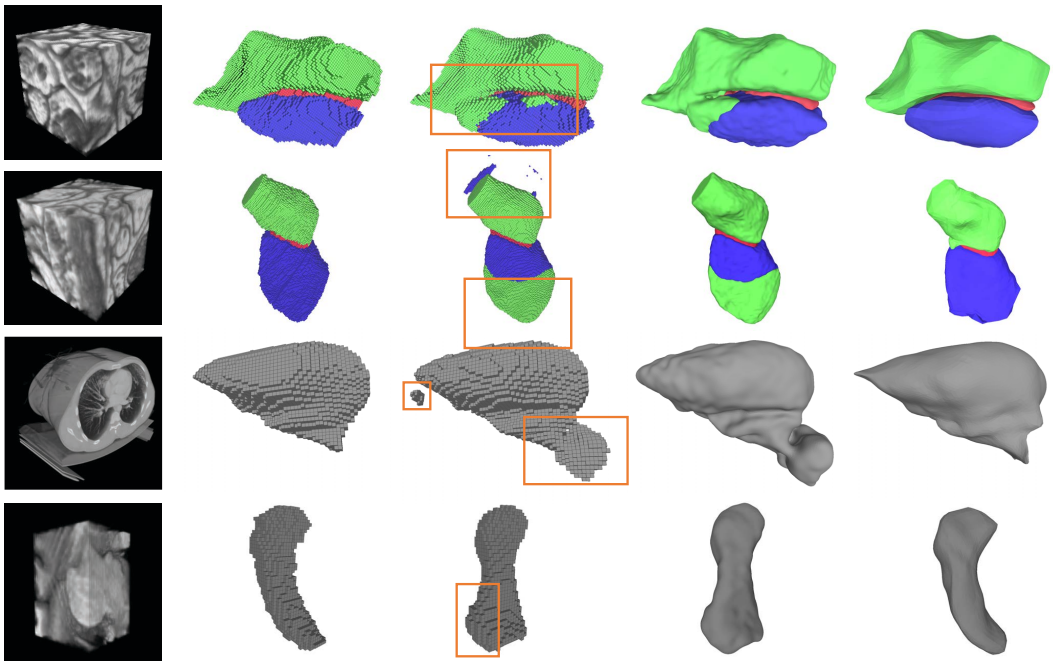


Figure 2.3: **Qualitative results.** (a) Input volumes. EM (row 1,2), CT(row 3), MRI(row 4) (b) Ground truth (c) CNN baseline (d) CNN baseline + post processing (e) **Voxel2Mesh**. The orange boxes highlight false positive regions.

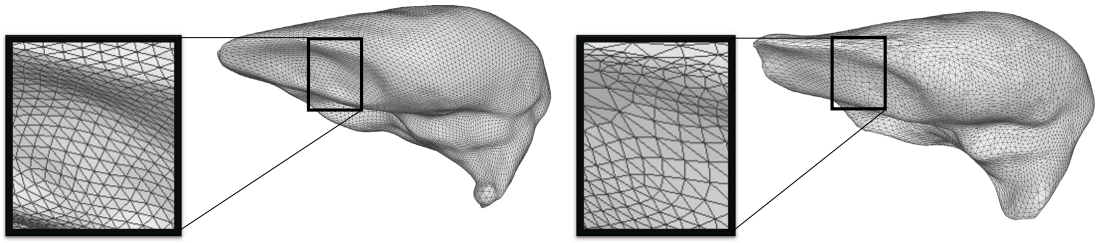
They are shown in blue, green, and red respectively in the first two rows of Fig. 2.3.

Hippocampus Dataset. It consists of 260 labeled MRI image cubes from the Medical Segmentation Decathlon [89]. The task is to segment the hippocampus, as depicted by the fourth row of Fig. 2.3.

Liver dataset. It consists of 20 labeled CT image cubes from the CHAOS challenge [55]. The task is to segment the liver as shown in the third row of Fig. 2.3.

2.4.2 Baselines

As the architecture of **Voxel2Mesh** borrows from **U-NET** and **Pixel2Mesh-3D**, they both constitute natural baselines. As state-of-the-art CNN based appro- aches, we use **TernausNet**, **LinkNet34**, **ResNet50** and **ResNet50-SE**, which belong to the ensemble of architectures that won the CHAOS challenge. **U-NET** was used as the base architecture by the winner of Medical Segmentation Decathlon. We also use **V-NET**, a widely used variant of **U-NET**, as a baseline. Since we are working with volumetric data, we use 3D variants of all these architectures.



(a) (b)

Figure 2.4: **Levels of resolution.** (a) **Pixel2Mesh-3D** result (10422 vertices). (b) **Voxel2Mesh** result (7498 vertices). With our adaptative unpooling, we obtain better results with fewer vertices.

2.4.3 Comparative Results

Fig. 2.3 and Fig. 2.4 depict our results qualitatively and we report quantitative ones in Tab. 2.1. As all the baselines shown above the first thick line return volumetric descriptions, we rasterize the mesh that **Voxel2Mesh** returns and use intersection over union (IoU) score to compare all the results against the ground truth. In all cases, we trained the networks three times with different initializations and we report the mean IoU and its standard deviation.

For completeness, we simulated a post-processing pipeline by selecting the best performing baseline for each of the three datasets, removing small false positive regions outside the object by connected component analysis, running Marching Cubes followed by smoothing using Algebraic Point Set Surfaces [39]. We refer to the result obtained by only removing the false positives as *Best CNN* + **CLN** and the one with full post-processing as *Best CNN* + **FPP**.

Voxel2Mesh performs best in all cases except the synaptic cleft, where it comes second. This can ascribed to the fact that synaptic clefts sometimes have holes in them, which a spherical mesh cannot capture. The improvement is most significant in the two datasets—liver and synaptic junction—with fewer training samples compared to the hippocampus dataset that features more training data.

The key challenge when training **Voxel2Mesh** is finding the correct amount of regularization. If the regularization is not sufficient, it can result in vertices with large movements and faces with large area. If it is too much, it can result in meshes that are over-smoothed. Therefore, the regularization coefficients should be found using hyperparameter grid search.

Table 2.1: Comparative results on three datasets using the IoU metric.

	Liver	Hippo.	Synaptic Junction		
			Pre-Synap.	Synapse	Post-Synap.
TernausNet [49]	*84.4 ± 1.3*	78.4 ± 1.2	73.5 ± 1.3	64.4 ± 0.5	*78.4 ± 1.3*
LinkNet34 [88]	82.8 ± 1.4	79.4 ± 0.8	72.3 ± 0.5	63.2 ± 1.2	78.2 ± 1.1
ResNet50 [55]	82.1 ± 0.7	80.7 ± 0.2	70.3 ± 0.8	63.3 ± 0.6	76.2 ± 1.4
ResNet50-SE [55]	82.6 ± 1.2	80.5 ± 1.3	71.3 ± 0.6	63.6 ± 0.7	76.3 ± 0.9
V-NET [73]	81.5 ± 1.4	75.3 ± 1.4	64.3 ± 0.7	65.2 ± 1.3	74.1 ± 0.7
U-NET [19]	84.2 ± 1.6	*80.9 ± 1.5*	*73.6 ± 1.3*	*67.2 ± 0.8*	78.2 ± 0.9
Best CNN + **CLN**	84.6 ± 1.7	81.1 ± 1.5	74.5 ± 1.2	67.6 ± 0.8	79.5 ± 0.9
Best CNN + **FPP**	84.3 ± 1.7	80.8 ± 1.5	74.2 ± 1.2	**67.4 ± 0.8**	79.3 ± 0.9
Voxel2Mesh	**86.9 ± 1.1**	**82.3 ± 0.9**	**77.3 ± 1.2**	65.3 ± 1.2	**83.2 ± 1.6**

2.4.4 Ablation Study

Pixel2Mesh-3D is the baseline closest to our approach because it directly outputs a 3D surface. Point Sampling (**PS**) and Uniform Mesh Unpooling (**UMU**) are the two modules in that play the same role as our Learned Neighborhood Sampling (**LNS**) and adoptive mesh unpooling (**AMU**). To check the importance of these strategies, we replaced them in our **Voxel2Mesh** pipeline by the equi- valent ones in **Pixel2Mesh-3D**. We also evaluate the performance of Hypothesis Sampling (**HS**), the sampling strategy of [100] that relies on a fixed neighborhood sampling. We report the results in Table 2.2 in Chamfer distance terms. They demonstrate that our adaptative strategies **LNS+AMU** deliver a clear benefit.

Table 2.2: Comparative results against CNN based mesh deforming baselines.

	Liver		Hippocampus	
	IoU	Cf.	IoU	Cf.
PS + UMU	83.3 ± 0.8	3.3×10^{-3}	78.8 ± 1.1	2.9×10^{-3}
HS + UMU	84.2 ± 0.6	2.8×10^{-3}	79.9 ± 0.9	2.3×10^{-3}
LNS + UMU	85.6 ± 0.9	2.1×10^{-3}	81.2 ± 1.2	1.8×10^{-3}
LNS + AMU (Voxel2Mesh)	**86.9 ± 1.1**	$\mathbf{1.3 \times 10^{-3}}$	**82.3 ± 0.9**	$\mathbf{1.1 \times 10^{-3}}$

2.5 Conclusion

We have proposed an end-to-end trainable architecture that takes an image volume as input and outputs a 3D surface mesh. This makes the post processing steps usually required to obtain such a mesh from a volumetric representation unnecessary, while preserving accuracy. Our adaptative sampling and unpooling strategies are key to this result. Not only does our architecture deliver good results, it also bridges the gap between voxel-based and surface-based representations. In

future work, we plan to extend our approach to structures with more complex topologies, such as the synaptic cleft and its potential holes.

3 Deep Active Surface Models

3.1 Introduction

Triangulated meshes are one of the most popular and versatile kind of 3D surface representation. In recent years, one of the most popular approaches to inferring such representations from images has been to use deep networks to produce a volumetric representation and then running a marching-cube style algorithm to create the mesh. However, using marching-cubes tends to introduce artifacts and introduces additional complexities when trying to make the process end-to-end differentiable. Hence, deep-learning methods that go *directly* and without resorting to an intermediate stage from 2D images [98, 79, 100] and 3D image stacks [103] to 3D triangulated surfaces have recently been proposed.

Unfortunately, these direct methods are also prone to generating unwanted artifacts such as those shown at the top of Fig. 3.1. State-of-the-art methods handle them by introducing additional regularizing loss terms such as the edge length loss, the normal consistency loss, or the Laplacian loss during training [38]. To be effective without sacrificing reconstruction accuracy, these terms must be carefully weighted, which is typically difficult to achieve.

In this paper, we solve this problem by introducing into the surface generating architecture a special-purpose layer that regularize the meshes using a semi-implicit scheme that involves recursively solving sparse linear systems of linear equations. It propagates smoothness constraints much faster and more reliably than traditional gradient descent-based energy minimization without requiring much computational power and yields surface meshes that fit the data while remaining smooth, such as those shown at the bottom of Fig. 3.1. Furthermore, this scheme enables us to

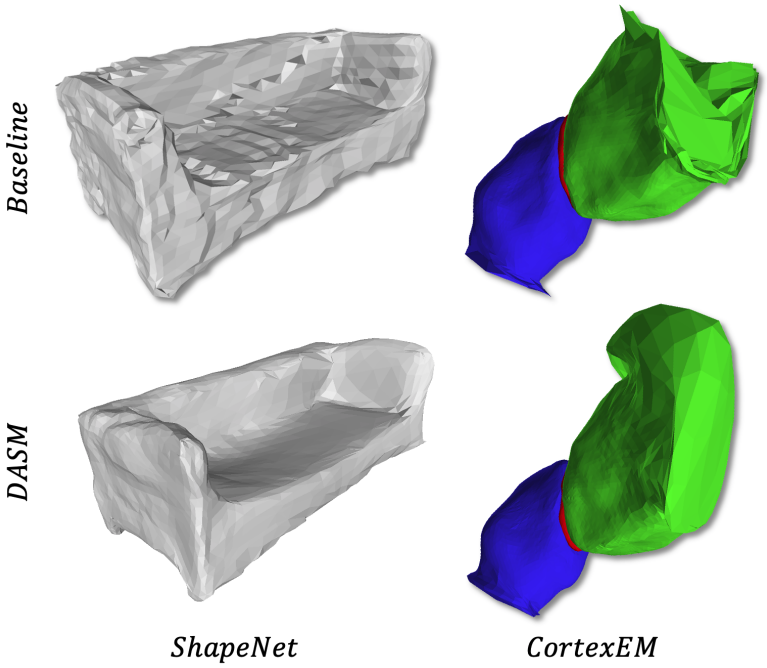


Figure 3.1: **Smoothness and Accuracy.** (Top) 3D surface meshes of a couch modeled from an RGB image and of a synaptic connection segmented from an electron microscopy stack by Mesh R-CNN [38] and by Voxel2Mesh [103], two state-of-the-art mesh-generating methods. (Bottom) Results using the same backbones augmented by our DASM smoothing layers. The meshes have far fewer artifacts and we will show that they are also more accurate.

- modulate locally the amount of regularization we impose so that we regularize only where it is needed and, hence, preserve accuracy;
- use meshes consisting of vertices with arbitrary degrees which is not commonly seen in majority of Active Shape Models.

Both of these are important to model complex 3D objects that can be smooth in some places and very curvy elsewhere.

We took our inspiration from the Active Surface Models (ASMs) idea [95, 96], which were first introduced over 30 years ago and also used a semi-implicit optimization scheme to model complex 3D shapes from images. Today, they are mostly used in conjunction with deep networks that are used to compute the data term that is minimized when deforming the models and meta-parameter maps that controls its behavior [69, 24]. Even though these methods are end-to-end trainable, they do not embed the ASMs within the contour deforming graph convolution networks as we do. Furthermore, they are limited to 2D contours whereas we handle irregular 3D surface meshes, that is, meshes whose vertices can be of arbitrary degrees. To this end, we propose an original

method to compute the derivatives required for back-propagation on such a mesh.

Our contribution therefore is *Deep Active Surface Models* (DASMs) that outperform equivalent architectures in which the smoothness constraints are imposed by minimizing a traditional loss function. We will demonstrate this for 3D surface reconstruction from 2D images and for 3D volume segmentation.

3.2 Related Work

3.2.1 Active Contour and Surface Models

Active contour models allow contours to be refined to account for local image properties while preserving global geometric primitives. They were first introduced in [53] for interactive delineation and then extended for many different purposes [33]. Active surface models operate on the same principle [95, 96] but replace the contours by triangulated meshes to model 3D surfaces. They have proved very successful for medical [70, 43] and cartographic applications [34], among others, and are still being improved [63, 82, 52].

Both active contours and surfaces operate by minimizing an objective function that is a weighted sum of data term derived from the images and a quadratic term that enforces global smoothness. They owe part of their success to the ability to perform the optimization using a semi-implicit scheme that propagates smoothness constraints much faster than gradient descent energy minimization would and gives them superior convergence properties. In our work, we integrate this scheme into our deep architecture.

3.2.2 Deep Surface Models

Before the advent of deep learning, mesh representations used to be dominant in the field of 3D surface reconstruction [70, 33]. Since then, they have been eclipsed by methods that rely on continuous deep implicit-fields. They represent 3D shapes as level sets of deep networks that map 3D coordinates to a signed distance function [80, 106] or an occupancy field [72, 13]. This mapping yields a continuous shape representation that is lightweight but not limited in resolution. This representation has been successfully used for single-view reconstruction [72, 13, 106] and 3D shape-completion [17].

However, for applications requiring explicit surface parameterizations, the non-differentiability of standard approaches to iso-surface extraction, such as the many variants of the Marching Cubes

algorithm [67, 76], as well as their tendency to produce artifacts, remain an obstacle to exploiting the advantages of implicit representations. The non-differentiability is addressed in [84] but the artifacts remain. Pixel2Mesh [98] and its newer variants [79, 100] represent attempts to overcome this difficulty by going *directly* from 2D images to 3D surface meshes without resorting to an intermediate stage. This approach has recently been extended to handle 3D image volumes [103]. These methods rely on graph-convolution layers to iteratively deform an initial mesh to match the target. While effective, they tend to produce large artifacts that detract both from their accuracy and their usability for further processing. This can be mitigated by introducing regularizing cost terms into the training loss function. However, it is difficult to weigh them properly to remove the artifacts without compromising the accuracy. Our method is designed to address this very issue.

3.2.3 Deep Contour Models

Similar to Pixel2Mesh and its variants for surface extraction, there exist its 2D counter parts. [64, 81] use graph-convolution networks to perform instance segmentation by deforming a contour. The same approach is used in [65] to perform interactive object annotation.

3.2.4 Active Contours/Surfaces and Neural Networks

Active contour models have been combined with deep networks by exploiting the differentiability of the active contour algorithm [69, 15, 42]. These approaches use deep networks to produce the data term as well as meta-parameter maps controlling the behavior of the active contour. This has also been exploited to correct errors in contour labels used in semantic segmentation [1]. However, we are not aware of any recent work that embeds active surfaces into deep networks. One potential reason is that most current approaches to generating 3D meshes in a deep learning context, such as those discussed above [98, 79, 100, 103], yield irregular meshes in which the vertices can have varying number of neighbors. This makes the computation of the derivatives required for back-propagation non-trivial. This has been addressed in the ASM context by introducing finite element-based computations that notably complexify the approach [61] or non-easily differentiable elements such as quadric fitting in the neighborhood of some vertices [62]. In this paper, we propose an approach that is back-propagation friendly on potentially large meshes.

3.3 Method

We first introduce the general formulation of Active Surface Models (ASMs) and then show how we can apply it to meshes whose vertices can be of arbitrary degree. Finally, we discuss the integration of ASMs and mesh-deforming Graph-Convolutional Neural Networks (GCNNs), which produces our Deep Active Surface Models (DASMs).

3.3.1 Active Surface Model (ASM)

An ASM consists of a surface $\mathscr{S}(\Phi)$ whose shape is controlled by a vector of parameters Φ and can be deformed to minimize an objective function $E(\Phi)$, often referred to as an *energy*. We first introduce a continuous formulation and then its discretization, which is the one used in practice.

Continuous Formulation. $\mathscr{S}$ is represented by the mapping from $\mathbb{R}^2$ to $\mathbb{R}^3$

$$v : (s, r; \Phi) \mapsto (v_x(s, r; \Phi), v_y(s, r; \Phi), v_z(s, r; \Phi)) , \tag{3.1}$$

where $(s, r) \in \Omega = [0, 1] \times [0, 1]$. Fig. 3.2 depicts this mapping and its derivatives. Φ is typically taken to be

$$\begin{aligned} \Phi^* &= \arg\min_{\Phi} E(\Phi) , \\ E(\Phi) &= E_{\text{dat}}(\Phi) + E_{\text{def}}(\Phi), \end{aligned} \tag{3.2}$$

where E_{dat} is a data term that measures how well the surface matches the images and E_{def} is a deformation energy that is smallest when the surface is smooth. E_{def} is often written as

$$\begin{aligned} E_{\text{def}} = \int_{\Omega} & w_{10} \left\| \frac{\partial v}{\partial s} \right\|^2 + w_{01} \left\| \frac{\partial v}{\partial r} \right\|^2 + 2 w_{11} \left\| \frac{\partial^2 v}{\partial s \partial r} \right\|^2 \\ & + w_{20} \left\| \frac{\partial^2 v}{\partial s^2} \right\|^2 + w_{02} \left\| \frac{\partial^2 v}{\partial r^2} \right\|^2 dr ds . \end{aligned} \tag{3.3}$$

The surface Φ^* that minimizes the energy $E(\Phi)$satisfies the associated Euler-Lagrange equation [20, 53]

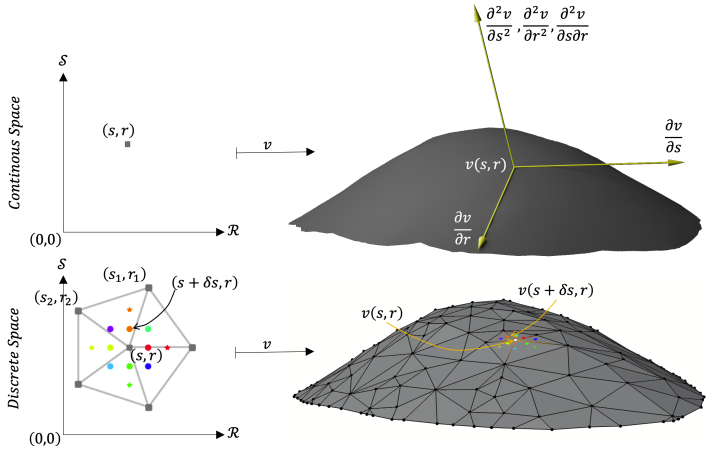


Figure 3.2: **Derivatives. Top.** The surface is represented by a differentiable mapping v from $\mathbb{R}^2$ to $\mathbb{R}^3$. **Bottom.** After discretization, let $v(s,r)$ be a vertex. To approximate derivatives with respect to s using finite differences, we need to estimate quantities such as $v(s+\delta s, r)$ where δs is small. To this end, we estimate the barycentric coordinates λ, λ_1, and λ_2 of $v(s+\delta s, r)$ in the facet it belongs to and take $v(s+\delta s, r)$ to be $\lambda v(s,r)+\lambda_1 v(s_1,r_1)+\lambda_2 v(s_2+r_2)$, where $v(s_1,r_1)$ and $v(s_2,r_2)$ are the other two vertices of the facet. The same operation can be performed for derivatives with respect to r.

$$F(v) = -\frac{\partial}{\partial s}\left(w_{10}\frac{\partial v}{\partial s}\right) - \frac{\partial}{\partial r}\left(w_{01}\frac{\partial v}{\partial r}\right) \\ +2\frac{\partial^2}{\partial s \partial r}\left(w_{11}\frac{\partial^2 v}{\partial s \partial r}\right) + \frac{\partial}{\partial^2 s}\left(w_{20}\frac{\partial^2 v}{\partial s^2}\right) + \frac{\partial}{\partial^2 r}\left(w_{02}\frac{\partial^2 v}{\partial r^2}\right), \tag{3.4}$$

where $F = -\nabla E_{\text{dat}}$.

Discrete Formulation. When $\mathscr{S}(\Phi)$ is discretized and represented by a triangulated mesh $\mathbf{M}(\Phi)$, Φ becomes the $3N$-vector built by concatenating the 3D coordinates of its N vertices. Using the finite-difference approximation described in the appendix, Eq 3.4 can be written in matrix form as

$$\mathbf{A}\Phi^* = F(\Phi^*)\,, \tag{3.5}$$

where F is the negative gradient of E_{dat} with respect to Φ. Because $\mathbf{A}$ is usually non-invertible, given an initial value Φ^0, a solution to this equation can be found by iteratively solving

$$\begin{aligned} \alpha(\Phi^t - \Phi^{t-1}) + \mathbf{A}\Phi^t &= F(\Phi^{t-1}) \,, \\ \Rightarrow (\mathbf{A} + \alpha\mathbf{I})\Phi^t &= \alpha\Phi^{t-1} + F(\Phi^{t-1}) \,, \end{aligned} \tag{3.6}$$

where $\mathbf{I}$ is the identity matrix. When the process stabilizes, $\Phi^t = \Phi^{t-1}$ and is a solution of Eq. 3.5.

The strength of this semi-implicit optimization scheme is that it propagates smoothness constraints much faster than traditional gradient descent that minimizes energy $E(\Phi)$ and at a low computational cost because $\mathbf{A}$ is sparse, which means that the linear system of Eq. 3.6 can be solved efficiently. In this scheme α plays the role of the inverse of a step size: When α is large enough for the Froebinius norm of $\alpha\mathbf{I}$ to be much larger than that of $\mathbf{A}$, the optimizer performs a steepest gradient step given by $F(\Phi^{t-1})$ with learning rate $\frac{1}{\alpha}$ at each iteration. Conversely, when α is small, $\mathbf{A}$ dominates and much larger steps can be taken.

In the original deformable contour models [53], the matrix $\mathbf{A} + \alpha\mathbf{I}$ was never inverted. Instead Eq. 3.6 was solved by LU decomposition. Instead, to implement this effectively on a GPU using sparse tensors and to speed up the computations of the losses and their derivatives, we approximate the inverse of $(\mathbf{A} + \alpha\mathbf{I})$ using the Neumann series

$$(\mathbf{A} + \alpha\mathbf{I})^{-1} \approx \sum_{n=0}^{K} (-1)^n \left(\frac{1}{\alpha}\right)^{n+1} A^n \,. \tag{3.7}$$

and use it to solve Eq. 3.6. We use $K = 4$, which yields a sufficiently good approximation of actually solving Eq. 3.6.

Computing the Regularization Matrix A. In most traditional ASMs, the meshes are either square or hexagonal and regular, which makes the computation of the derivatives of the mesh vertices possible using finite-differences and, hence, the regularization matrix $\mathbf{A}$ of Eq. 3.5 easy to populate.

When the mesh is triangular and irregular, vertices can have any number of neighbors and the computation becomes more complex. Nevertheless the required derivatives, of order 2 and 4, can still be expressed as finite differences of weighted sums of vertex coordinates where the weights are barycentric coordinates of small perturbations of the original vertices. This is explained in more details in the appendix.

3.3.2 Deep Active Surface Model (DASM)

The update equation in a typical mesh-deforming graph-convolutional neural network (GCNN) that plays the same role as that of Eq. 3.6 is

$$\Phi^t = \Phi^{t-1} + \frac{1}{\alpha} F(\Phi^{t-1}, \mathbf{X}^{t-1}) \,, \tag{3.8}$$

where F denotes the negative gradient of the loss function calculated using the feature vector $\mathbf{X}^{t-1}$ associated with the mesh parameters Φ^{t-1}. In the case of our deep active surface models, it becomes

$$(\mathbf{A} + \alpha \mathbf{I})\Phi^t = \alpha \Phi^{t-1} + F(\Phi^{t-1}, \mathbf{X}^{t-1}) \,, \tag{3.9}$$

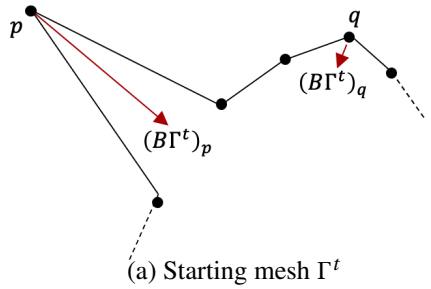
as in Eq. 3.6. In Eq. 3.8, the loss function typically includes a regularization term to keep the mesh smooth, whereas in Eq. 3.9 our semi-implicit scheme enforces smoothness by solving the linear equation.

Uniform vs Adaptive DASMs Eq. 3.9 forms the basis of the simplest version of our DASMs, which we will refer to as Uniform DASMs because the same amount of smoothing is applied across the whole mesh. This may result in under- or over-smoothing because some parts of the objects require more smoothing while some parts do not.

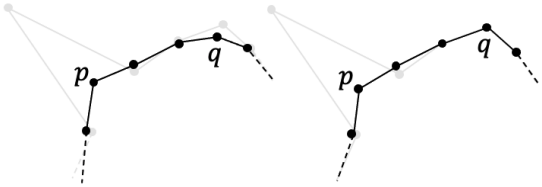
To account for this, we also introduce Adaptive DASMs that are designed to smooth only where necessary, as indicated by an auxiliary metric. Experimentally, adaptive smoothing is required when the GCNN produces particularly large deformations but only in a very specific part of the mesh or fails to smooth-out artifacts produced by mesh initialization algorithms. This could be eliminated by strongly smoothing everywhere but would degrade accuracy in high-curvature areas.

To solve this problem, we begin by using the approximation of $(\mathbf{A} + \alpha \mathbf{I})^{-1}$ from Eq. 3.7 to rewrite the evolution equation of Eq. 3.9 as

$$\begin{aligned} \Gamma^t &= \Phi^{t-1} + \frac{1}{\alpha} F(\Phi^{t-1}, \mathbf{X}^{t-1}) \,, \\ \Phi^t &= (\mathbf{I} + \sum_{n=1}^{K} (-1)^n \left(\frac{1}{\alpha}\right)^n \mathbf{A}^n)\Gamma^t \,, \\ &= \Gamma^t + \mathbf{B}\Gamma^t \text{ withă ă} \mathbf{B} = \sum_{n=1}^{K} (-1)^n \left(\frac{1}{\alpha}\right)^n \mathbf{A}^n \,. \end{aligned} \tag{3.10}$$



(a) Starting mesh Γ^t



(b) Uniform DASM (c) Adaptive DASM

Figure 3.3: **Uniform vs Adaptative Smoothing.** (a) Mesh at time t. In general, $|\mathbf{B}\Gamma^t|_q < |\mathbf{B}\Gamma^t|_p$. (b) When using enough uniform smoothing to remove the irregularity at point p, the mesh will typically be oversmoothed at q. (c) When using adaptative smoothing, the mesh is smoothed around p but not oversmoothed around q.

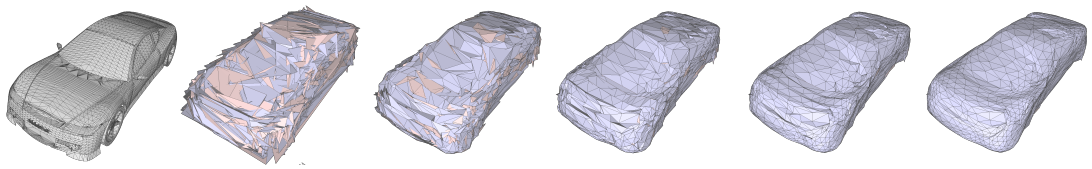
Γ^t represents Φ^{t-1} incremented by the negative gradient of the loss function $F(\Phi^{t-1})$ but not yet smoothed. In other words, we have rewritten the smoothing operation that transforms Γ^t into Φ^t as simply adding $\mathbf{B}\Gamma^t$ to Γ^t . This gives us the freedom to decide where we want to smooth and where we do not by introducing a diagonal matrix Λ and rewriting the update rule of Eq. 3.10 as

$$\Phi^t = \Gamma^t + \Lambda\mathbf{B}\Gamma^t \,. \tag{3.11}$$

This update rule is similar to the one of the Adagrad algorithm [26]. Here, each diagonal component $\lambda_{i,i}$ of Λ rescales the corresponding component $(\mathbf{B}\Gamma)_i$ of $\mathbf{B}\Gamma$. In Adagrad, adaptive re-scaling is a function of past gradients. Here we take it to be a function of current surface gradients because we have observed that $|\mathbf{B}\Gamma^t|_i$ tends to grow large when the facets increase in size and smoothing is required, and remains small otherwise. We therefore take the diagonal values of Λ to be

$$\lambda_{ii} = \sigma(|\mathbf{B}\Gamma^t|_i; \beta, \gamma) \,, \tag{3.12}$$

where σ is the Sigmoid function and β, γ are its steepness and midpoint. In this way, for small values of $|\mathbf{B}\Gamma^t|_i$, there is almost no smoothing, but for larger ones there is. Fig. 3.3 illustrates this behavior.



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

Figure 3.4: **Performing multiple smoothing steps.** (a) Ground truth (b) Input (c) ASM 1-step (d) ASM 3-steps (e) ASM 5-steps (f) ASM 7 steps. We start with a very noisy mesh produced by the algorithm of [38] run without regularization. We have colored front-faces and back-faces of the meshes in blue and pink respectively for better visualization. After running one step of surface evolution given by Eq. 3.6, we obtain the mesh of Fig. 3.4 (c). Subsequent meshes shown in Fig. 3.4 (d,e,f) are obtained by continuing the surface evolution for 3, 5, and 7 steps respectively. In these subsequent steps we set $F(\Phi^{t-1})$ to zero.

Recursive smoothing Any single DASM step given by Eq. 3.11 can only rectify a finite amount of deformations. To mitigate this, we perform more than one adaptive-smoothing step in-between gradient updates. During these additional smoothing steps no gradient update is done and we use $F(\Phi^{t-1}, \mathbf{X}^{t-1}) = 0$. In practice we perform these steps until $\|\Phi^t - \Phi^{t-1}\| < \epsilon$, where ϵ is a preset constant. Fig. 3.4 illustrates this process.

Loss terms In architectures such as Mesh R-CNN [38] and Voxel2Mesh [103], a loss term is used to supervise the output of each mesh-refinement stage. We follow the same approach and add a loss term at the end of each DASM module. We write it as

$$\begin{aligned} \mathscr{L} &= \mathscr{L}_{data} + \lambda \mathscr{L}_{reg.}\,, \\ \mathscr{L}_{data} &= \lambda_{cf.}\mathscr{L}_{cf.} + \lambda_{n.dist.}\mathscr{L}_{n.dist.}\,, \\ \mathscr{L}_{reg.} &= \lambda_{edge}\mathscr{L}_{edge} + \lambda_{Lap.}\mathscr{L}_{Lap.} + \lambda_{n.cons.}\mathscr{L}_{n.cons.}\,. \end{aligned} \tag{3.13}$$

Here $\mathscr{L}_{Cf.}, \mathscr{L}_{n.dist.}$ are Chamfer and Normal distances [38] and $\mathscr{L}_{edge}, \mathscr{L}_{Lap.}, \mathscr{L}_{n.cons.}$ are edge length loss, Laplacian loss and normal consistency loss, respectively [98]. All these loss terms are used in Voxel2Mesh [103] except $\mathscr{L}_{Norm.}$. Similarly, they are all used in Mesh R-CNN [38] except $\mathscr{L}_{Lap.}$ and $\mathscr{L}_{Norm.}$.

3.4 Experiments

In this section, we test DASM's ability to predict 3D surfaces from 2D images on Shapenet [10] and to extract 3D surfaces from electron microscopy image stacks.

3.4.1 From 2D Images to 3D Surfaces

For prediction of 3D surfaces from 2D images, we benchmark our Adaptative DASM, which we will refer to as Ad.-DASM, on the ShapeNet dataset [10].

Baselines. We use Mesh R-CNN [38] both as a baseline and as the backbone of our network because, among methods that use explicit surface representations, it is currently reported as yielding the best results on ShapeNet. We also compare against **Pixel2Mesh-3D** [98].

Dataset. ShapeNet is a collection of 3D textured CAD models split into semantic categories. As in the Mesh R-CNN experiments, we use ShapeNetCore.v1 and corresponding rendered images from [18]. They are of size 137×137 and have been captured from 24 random viewpoints. We use the train / test splits of [38], that is, 35,011 models seen in 840,189 images for training and 8,757 models seen in 210,051 images for testing. We use 5% of the training data for validation purposes.

Metrics. We use the same metrics as in Mesh R-CNN. They are the Chamfer distance, Normal distance, and $F1^\tau$ at $\tau = 0.1, 0.3$ and 0.5. For the Chamfer distance a lower value is better while a higher value is better for the others.

Implementation. We use the publicly available Pytorch implementation of Mesh R-CNN and incorporate Adaptative DASM layers after each mesh-refinement stage. We also add a Uniform DASM layer after the cubify operation to make the input to mesh refinement stages smooth. We train the networks for 12 epochs using Adam optimizer [57] with a learning rate 10^{-4}. We set $\alpha = 1, \beta = 6000$ and $\gamma = 15$.

Mesh R-CNN only uses the $\mathscr{L}_{edge}$ term of Eq. 3.13 for regularization purposes when training on ShapeNet and turns off the term $\mathscr{L}_{Lap.}$ because, according to remarks on Github by the authors, it has not helped to improve the results. For a fair comparison, we therefore do the same. In this setup, $\mathscr{L}_{edge}$, which penalizes increases in edge-length, is the only other source of geometric regularization besides the one we provide with our DASM layers. We will therefore experiment with different values of λ_{edge}, the weight parameter in Eq. 3.13 that controls how much influence it is given.

Results. We provide qualitative results in Fig. 3.5 and report quantitative results in Table 3.1 for $\lambda_{edge} = 0.2$. Ad.-DASM outperforms **Pixel2Mesh-3D** and boosts the performance of Mesh R-CNN. Furthermore, the meshes it produces are of much a better visual quality.

Table 3.1: **Comparative results on ShapeNet.**

	Chf. (↓)	Normal	$F^{0.1}$	$F^{0.3}$	$F^{0.5}$
Pixel2Mesh-3D	0.241	0.701	31.6	77.3	91.3
Mesh R-CNN	0.189	0.691	32.8	80.4	**92.6**
Ad.-DASM	**0.183**	**0.727**	**33.5**	**81.0**	**92.6**

Table 3.2: **Results on ShapeNet** as a function of λ_{edge}.

$\lambda_{e.}$		Chf. (↓)	Norm.	$F^{0.1}$	$F^{0.3}$	$F^{0.5}$
1.0	Mesh R-CNN	0.232	0.684	**29.7**	**76.6**	**89.4**
	Ad.-DASM	**0.231**	**0.691**	29.3	**76.6**	89.3
0.6	Mesh R-CNN	0.212	0.681	30.6	79.2	**91.4**
	Ad.-DASM	**0.206**	**0.693**	**31.4**	**79.5**	**91.4**
0.2	Mesh R-CNN	0.189	0.691	32.8	80.4	**92.6**
	Ad.-DASM	**0.183**	**0.727**	**33.5**	**81.0**	**92.6**
0.0	Mesh R-CNN	**0.144**	0.713	**35.8**	**85.1**	**94.2**
	Ad.-DASM	0.167	**0.718**	34.3	84.3	93.9

Table 3.3: **Ablation study on ShapeNet.**

	Chf. (↓)	Normal	$F^{0.1}$	$F^{0.3}$	$F^{0.5}$
PostProc-ASM	0.249	0.673	28.5	75.3	88.1
Uniform-DASM	0.201	0.699	31.4	78.3	91.3
Ad.-DASM	**0.183**	**0.727**	**33.5**	**81.0**	**92.6**

In Table 3.2, we report similar results for different values of λ_{edge}, which are depicted qualitatively by Fig. 3.6. The trend is the same for $\lambda_{edge} = 0.6$ and 1.0. However, for $\lambda_{edge} = 0.0$, the Chamfer distance for Mesh R-CNN is lowest even though the resulting meshes are extremely noisy, as can be seen in the leftmost column of Fig. 3.6. Our interpretation for this somewhat surprising result is that, in this regime, the meshes produced by Mesh R-CNN are so noisy that DASM smoothing takes them away from the data they are trying to fit and degrades the Chamfer distance. In any event, even though the Chamfer distance is low, this can hardly be considered as a good results, hence confirming the observation made in [38, 98] that this metric might not be the best to evaluate the quality of a mesh. When $\lambda_{edge} = 1.0$, the difference between DASM and Mesh R-CNN performance is not statistically significant and this is because, when using higher λ_{edge}, there are not many anomalies for the DASM to fix.

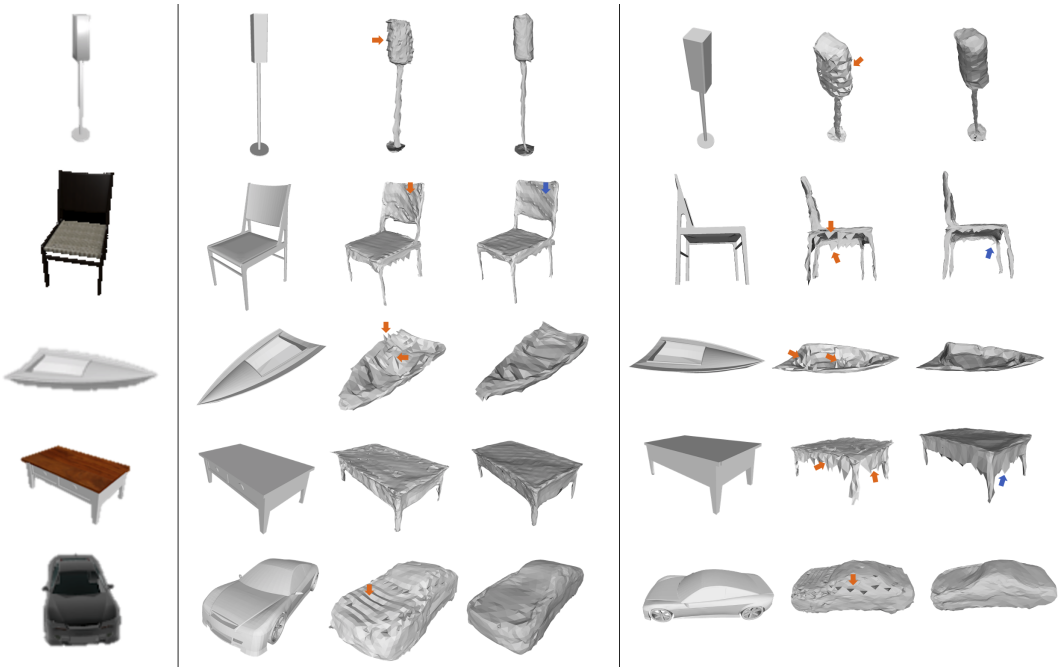


Figure 3.5: **ShapeNet Results.** (a) Input images (b) Mesh R-CNN results from two different viewpoints. The orange arrows highlight commonly seen Mesh R-CNN artifacts. (c) DASM results in the same two views. The meshes are much smoother and most artifacts have disappeared, except for a few highlighted by blue arrows.

Ablation Study It could be argued that we would have gotten similar results by simply smoothing our meshes as a post-processing step. To demonstrate this is not the case, we implemented PostProc-ASM that starts with Mesh R-CNN model trained with $\lambda_{edge} = 0.2$ that is then adaptively smoothed by running several times the surface evolution update of Eq. 3.11. In Table. 3.3, we compare PostProc-ASM against Ad.-DASM and the results are clearly worse. We also compare Ad.-DASM against Uniform-DASM, which clearly shows the benefit of the adaptive scheme of Section 3.3.2.

Failure modes The main source of failures are anomalies produced by the GCNN that are too large to be rectified. Remaining ones are denoted by blue arrows in Fig 3.5.

3.4.2 From 3D Image stacks to 3D Surfaces

In this section we benchmark our approach on CortexEM dataset and compare DASM against **Voxel2Mesh** [103] and several other baselines.

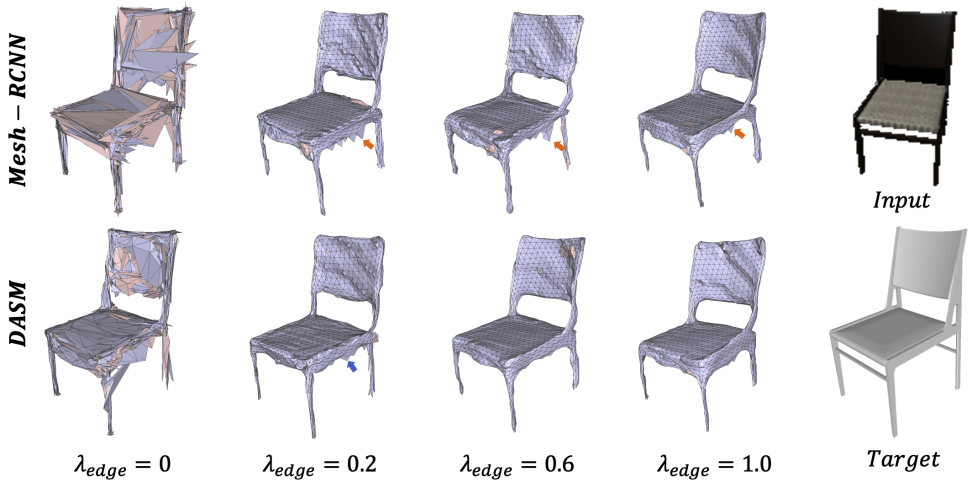


Figure 3.6: **Influence of the regularization term.** As λ_{reg} increases, the output of both methods becomes smoother but only DASM completely eliminates the artifacts. Note that in the case $\lambda_{reg} = 0$, the output of Mesh R-CNN is an extremely irregular mesh that nevertheless scores well on the Chamfer metric.

Baselines. We use **Voxel2Mesh** both as a baseline and as the backbone of our network. We also compare our performance against several architectures popular in the biomedical imaging community [19, 73, 49, 88, 55].

Dataset. CortexEM is a $500 \times 500 \times 200$ FIB-SEM image stack of a mouse cortex. From this 26 sub-volumes with dimension $96 \times 96 \times 96$ were extracted so that each one contains a synaptic junction that is roughly centered. 14 sub-volumes in the first 100 slices in the image stack are used for training and the remaining 12 in the next 100 slice are used for testing. The task is to segment the pre-synaptic region, post-synaptic region, and synaptic cleft as shown in Fig. 3.7.

Metrics. As in [103] and many other papers, we use the intersection-over-union (IoU) as a measure of quality for volumetric segmentation. To compare the meshes, we use the Chamfer distance as in [103]. We repeat our experiments 3 times for each model and report the mean and the standard deviation.

Implementation. As we did with Mesh R-CNN in Section 3.4.1, we incorporate Ad.-DASM layers into a **Voxel2Mesh** backbone. We train the networks for 150000 iterations using Adam optimizer [57] with a learning rate of 10^{-4}. We set $\alpha = 1, \beta = 6000$ and $\gamma = 45$.

To match the conditions in [103], Ad.-DASM was trained using $\lambda_{edge} = \lambda_{Lap.} = \lambda_{n.cons.} = 0.25$ in Eq. 3.13. For comparison purposes, we also use $\lambda_{edge} = \lambda_{Lap.} = \lambda_{n.cons.} = 0.025$. To differentiate two versions, we add the regularization coefficients as a subscript to model names. This gives us

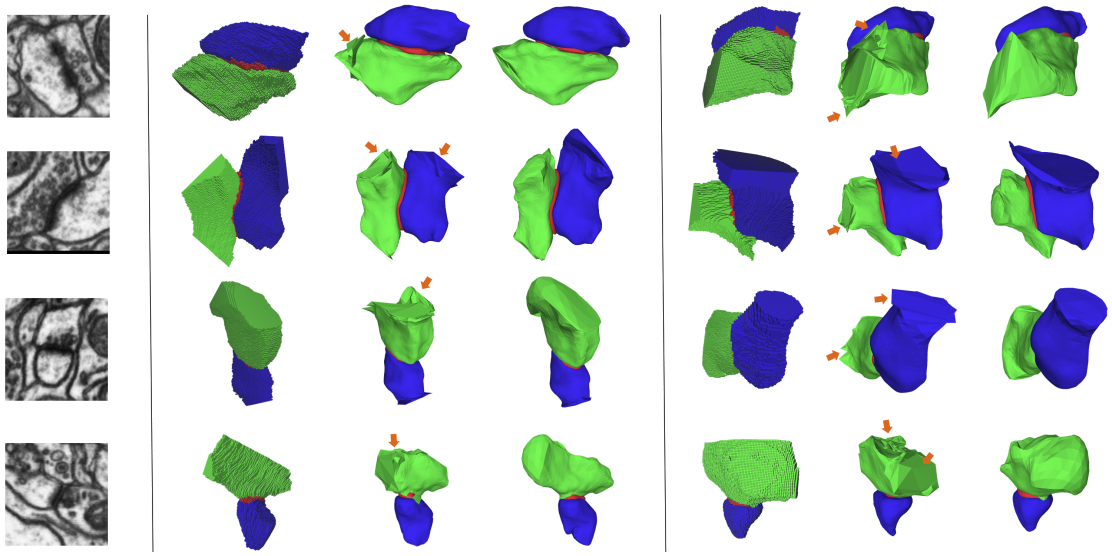


Figure 3.7: Modeling Synapses from Electron Microscopy Image Stacks (a) Representative slice from the input volume. (b) **Voxel2Mesh** results seen from two different views. (c) DASM results seen from the same two views. The pre-synaptic region, post-synaptic region, and synaptic cleft are shown in blue, green, and red, respectively. The DASM results are much smoother and without artifacts, while also being more accurate.

Table 3.4: **Comparative results on CortexEM.** We use the IoU metric to compare volumetric segmentations.

	Pre-Syn.	Synapse	Post-Syn.
TernausNet [49]	73.5 ±1.3	64.4 ±0.5	78.4 ±1.3
LinkNet34 [88]	72.3 ±0.5	63.2 ±1.2	78.2 ±1.1
ResNet50 [55]	70.3 ±0.8	63.3 ±0.6	76.2 ±1.4
ResNet50-SE [55]	71.3 ±0.6	63.6 ±0.7	76.3 ±0.9
V-NET [73]	64.3 ±0.7	65.2 ±1.3	74.1 ±0.7
U-NET [19]	73.6 ±1.3	**67.2 ±0.8**	78.2 ±0.9
Voxel2Mesh$_{0.25}$	77.3 ± 1.2	65.3 ±1.2	83.2 ±1.6
Voxel2Mesh$_{0.025}$	76.8 ± 0.9	65.4 ±1.6	81.2 ±1.4
Ad.-DASM$_{0.25}$	77.7 ±0.8	65.3 ±0.7	83.3 ±0.8
Ad.-DASM$_{0.025}$	**79.4 ±1.4**	65.3 ±0.9	**85.5 ± 1.2**

Table 3.5: **Comparative results on CortexEM.** We use the Chamfer distance ($\times 10^{-2}$)

	Pre-Syn.	Synapse	Post-Syn.
Voxel2Mesh$_{0.25}$	1.62 ±0.4	0.19 ±0.2	2.41 ±0.7
Voxel2Mesh$_{0.025}$	1.53 ±0.6	**0.18 ± 0.4**	2.35 ±0.4
Ad.-DASM$_{0.25}$	1.59 ±0.5	0.19 ±0.3	2.39 ±0.8
Ad.-DASM$_{0.025}$	**1.47 ± 0.6**	0.18 ±0.5	**2.31 ± 0.6**

Voxel2Mesh$_{0.25}$, Ad.-DASM$_{0.25}$, **Voxel2Mesh**$_{0.025}$, and Ad.-DASM$_{0.025}$.

Results We report quantitative results in Table 3.4 in IoU terms and in Table 3.5 in Chamfer distance terms. Fig. 3.7 depicts qualitative results for Ad.-DASM$_{0.25}$, and **Voxel2Mesh**$_{0.025}$. Ad.-DASM$_{0.025}$ easily outperforms **Voxel2Mesh** when segmenting pre and post synaptic regions. For the smaller synaptic junction, **Voxel2Mesh** and DASM are statistically equivalent because, unlike for the other two regions, their shapes are simple and there is not much improvement for DASM to make. In fact, the best result is obtained by a vanilla U-Net.

Table 3.6: **Run times.** We report the time required to perform a forward and backward pass. $|V|$ is the number of mesh vertices.

	Time (sec.)	$\|V\|$
Voxel2Mesh	1.06 ±0.04	1534 ±4
Ad.-DASM	1.47 ± 0.06	1535 ±2

Computation time We report average execution time for a single forward and backward pass for **Voxel2Mesh** and DASM in Table 3.6. We run this test on a single Tesla V100 GPU. We have implemented the ASM module using custom CUDA kernels and uses sparse tensors. Assembling the regularization matrix and performing the update of Eq. 3.11 adds a 40% overhead, which is reasonable giving how large the matrices we deal with are.

3.5 Conclusion

We have developed an approach to incorporating Active Shape Models into layers that can be integrated seamlessly into Graph Convolutional Networks to enforce sophisticated smoothness priors at an acceptable computational cost. By embedding the smoothing directly into the update equations, we can deliver smoother and more accurate meshes than methods that rely solely on a regularization loss term.

In future work, we will further improve our adaptive DASM scheme. Currently, it relies on a hand-designed metric to detect where to smooth and where not to. We will replace it by an auxiliary network that predicts where smoothing is required, given the current state of the mesh and the input data.

4 Weakly Supervised Volumetric Image Segmentation with Deformed Templates

4.1 Introduction

State-of-the-Art volumetric segmentation techniques rely on Convolutional Neural Networks (CNNs) operating on image volumes [19, 88, 73]. However, their performance depends critically on obtaining enough annotated data, which itself requires expert knowledge and is both tedious and expensive.

Weakly-supervised image segmentation techniques can be used to mitigate this problem. They typically rely on tag annotations [46, 37, 36, 28] or coarse object annotations in the form of point annotations [3, 83, 110], bounding box annotations [45, 59, 91], scribbles [110] or approximate target shapes [56]. However, these techniques have been mostly demonstrated in 2D. For 3D volume segmentation, the dominant approach is to fully label a subset of 2D slices [19, 25]. This is referred to as *weak supervision* in the literature, even though it requires full supervision within each slice.

By contrast, we propose an approach to 3D segmentation that is truly weakly-supervised in the sense that we only need to provide a sparse set of 3D points on the surface of the target objects, which can be done quickly and easily. To this end, we introduce the **Weak-Net** architecture depicted by Fig. 5.2. It comprises two U-Net-like networks. The first one is the *core network* that takes an image volume $\mathbf{X}$ as input and produces a segmentation map $\hat{\mathbf{Y}}$. The second one is the *reconstruction network* that takes $\hat{\mathbf{Y}}$ as input and outputs $\hat{\mathbf{X}}$, which should reproduce the original image $\mathbf{X}$ as exactly as possible. During training, we jointly minimize the mean squared error (MSE) from $\mathbf{X}$ to $\hat{\mathbf{X}}$ and the binary cross entropy between $\hat{\mathbf{Y}}$ and the mask volume $\mathbf{Y}$ defined by the annotations. We obtain $\mathbf{Y}$ by iteratively deforming a simple template, such as a sphere, to fit

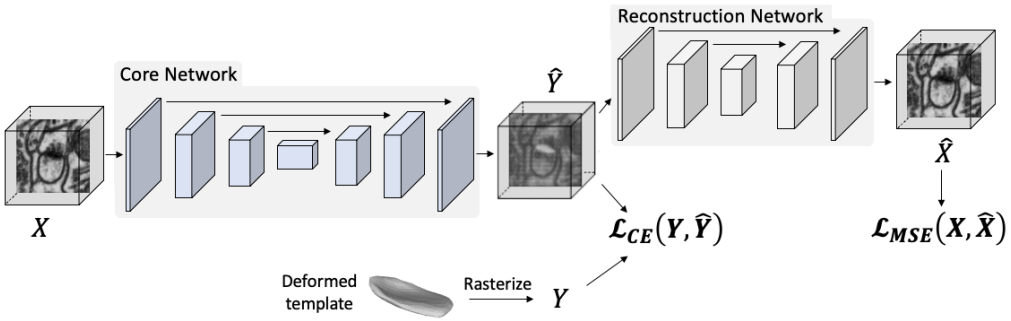
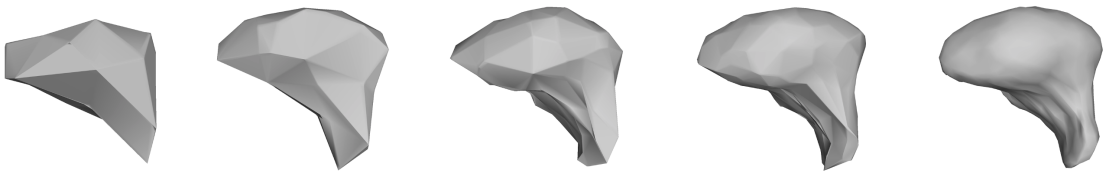


Figure 4.1: **Weak-Net** architecture. A first U-Net takes the image $\mathbf{X}$ as input and outputs a segmentation $\hat{\mathbf{Y}}$, which is in turn fed to a second U-Net that outputs a reconstructed image $\hat{\mathbf{X}}$. Training is achieved by jointly minimizing L_{mse} and L_{ce}. This encourages $\hat{\mathbf{X}}$ to resemble the original image and $\hat{\mathbf{Y}}$ to be similar to $\mathbf{Y}$, a roughly aligned version of a template.

point annotations provided by an annotator, as shown in Fig. 4.2. The mesh is then rasterized to obtain the voxel ground truth. When the user only provides few annotations, the deformed template does not match the target object closely. Hence, minimizing the binary cross-entropy only provides coarse supervision. However, minimizing the MSE loss so that the reconstructed image resembles the original one provides the additional supervision required for good accuracy.

We evaluate the performance of **Weak-Net** on Computed Tomography (CT), Magnetic Resonance Imagery (MRI) and Electron Microscopy (EM) image datasets. We will show that it outperforms the standard approach to weak-supervision in 3D at a reduced supervision cost. More specifically, we can deliver the same accuracy as when fully annotating 2D slices for less than a third of the annotation effort. This is important because annotators typically are experts whose time is both scarce and valuable. Furthermore, it creates the basis for interactive annotating strategies that deliver the full accuracy at a lower cost than full supervision.



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

Figure 4.2: **Deformed templates.** (a,b,c,d,e) Deformed template using $N = 25, 50, 125, 250$, and 3661 user-supplied 3D points when segmenting a liver. $N = 3661$ corresponds to full annotations in all slices.

4.2 Related Work

We first briefly review current approaches to weak-supervision for segmentation in 2D images and 3D volumetric images. We then discuss the use of atlases for 3D segmentation in a biomedical

context.

4.2.1 Weakly-Supervised Image Segmentation

Segmenting 2D Images. Tag annotations are among the weakest forms of annotations that have been used to segment natural images [46, 37, 36, 28] and medical images [30]. However, they rarely provide enough supervision for accurate results. Furthermore, they can only be used when segmentation can be formulated in terms of an object classification problem, which is not the case when attempting to segment a hippocampus or a liver that is present in every image. Box annotations have also been used [45, 59, 91]. In terms of localization, they are more effective than tags but still provide little information about the precise shape of the target objects. By contrast, point annotations [3, 83, 110], scribbles [110], and approximate shape annotation [56] can be used to provide useful shape information.

The annotation process can be sped-up using dynamic programming [75] or deformable contours [53], also known as *snakes*. This makes it possible to mark only a subset of points along a contour and have the system refine their location while following the target object boundary. Unfortunately, these algorithms are hard to deploy effectively in medical imagery because there are many contours besides those of interest and they can easily confuse the semi-automated algorithms, which greatly reduces their usefulness. This is addressed in [81] by introducing *deep snakes* that only need a simple approximate contour for initialization purposes. In [2, 65], the annotator is brought into the loop by giving corrective clicks when necessary, which results in an effective approach to annotation. However, these deep snakes require fully labelled data for their own training and have only been demonstrated in the 2D case.

Segmenting 3D Volumetric Images. By far the most prevalent approach to weak-supervision for 3D segmentation, is to *fully* annotate subsets of slices from the training volumes [19]. Even though this reduces the segmentation effort, the annotator still has to carefully trace the object boundary in those slices. The effort can be reduced by using scribbles in individual slices [25] or any of the semi-automated techniques described above. Unfortunately, the effectiveness of those that do not require training [75, 53] is limited while the others [2, 65, 81] require full supervision and are not applicable in our scenario.

Part of the problem is that these annotation techniques operate purely in 2D without exploiting the 3D nature of the data. We will show that by so doing we only need a comparatively small number of point annotations for effective training.

4.2.2 Template Based Approaches

Shape priors have long been used for image segmentation [32, 21], well before the rise to prominence of deep networks, and more recent work use shape priors in conjunction with deep nets [103, 74] In the field of medical imaging, these priors are usually supplied in the form of sophisticated templates known as Probabilistic Atlases (PAs) that assign to each pixel or voxel a probability of belonging to a specific class. They are typically built by fusing multiple manually annotated images and incorporated in to CNN architectures as auxiliary inputs to provide localization priors that help the network find structures of interest. The PAs can be either very detailed, as in [92, 93, 112, 60], to model structures that are known in detail or very rough, as in [101], to deal with 3D structures whose shape can vary significantly.

PAs designed by point annotations have also been used in medical imaging [51, 86] as well as in natural image segmentation as a source of prior information [108, 8, 68, 99]. They are often referred to as seed layers and come in two main flavors, Gaussian priors [108, 8, 68, 99, 86] or binary seed layers [51]. These seed layers either indicate points inside the object [108, 51] or points on the boundary of the object [68, 99].

However, within the field of deep learning, PAs have been mostly used in fully-supervised approaches. An exception is the work of [93], but here they are only used for pre-training purposes.

One-shot and Few-shot learning based segmentation algorithms [109, 22, 94] can also be considered as template-based and semi-supervised approaches, as they still require at least one or a few fully annotated target objects as the atlas(es). In this work, we demonstrate another use for templates, that is, supplying rough annotations we use to train a weakly-supervised network.

4.2.3 Image Reconstruction

Image reconstruction is used for semi-supervised [27, 14, 66, 31] and unsupervised [105, 11, 9] image segmentation as an auxiliary task to improve the results. In this work, we demonstrate that this idea also applies in a weakly-supervised setting to improve the segmentations produced by the core-network trained with rough annotations. In our framework, the image reconstruction network helps refine the rough initial shapes we obtain from the annotations. There are alternative approaches to boundary refinement [1, 12, 40] but they are designed for 2D segmentation and extending them to 3D would be non-trivial.

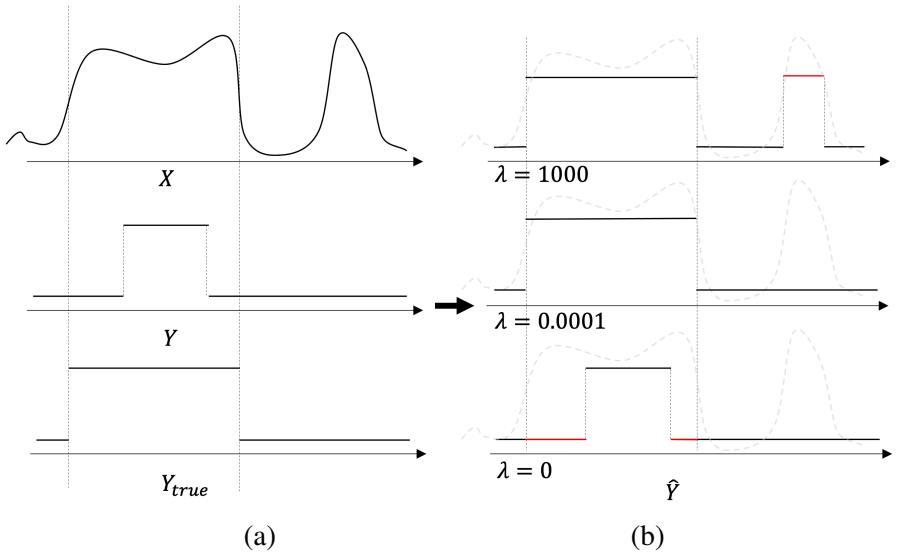


Figure 4.3: **Using a secondary task to improve segmentations**. (a) **X** is the input image and $\mathbf{Y}_{true}$ is the ground-truth segmentation. **Y** is the atlas that is roughly correct but in which the boundaries of the target object are inaccurate. (b) When λ is too large, $\mathscr{L}_{mse}$ dominates and the resulting segmentation features a spurious region shown in red that corresponds to image boundaries that are *not* those of the target structure. When λ is zero, only $\mathscr{L}_{ce}$ is minimized and minimizing produces boundaries that are also those of the atlas, which are not at the right place as the denoted by the red lines. For appropriate values of λ, the boundaries are correct and the spurious region is eliminated.

4.3 Method

Our approach to 3D segmentation is weakly-supervised in the sense that, for training, we only need to provide a sparse set of 3D points on the surface of target objects for each image volume, which we use for training purposes. These points are used to deform a template, such as the one depicted by Fig. 4.2, to provide a rough indication of where the target object boundary is. It is exploited by our network to learn weights that yield accurate object boundaries. In short, we provide minimal human input at training time so that, at inference time, the trained network can be used without human intervention.

In this section, we first describe our **Weak-Net** architecture and the loss functions that allow it to effectively exploit the weak supervision provided by our rough templates. We then discuss our approach to deforming simple spherical templates given only a few user-supplied 3D points.

4.3.1 Network Architecture and Losses

Fig. 5.2 depicts the the **Weak-Net** architecture. It relies on two U-Net networks [19]. The first takes as input an image volume **X** of $D \times H \times W$, where D, H, W stand for depth, height and

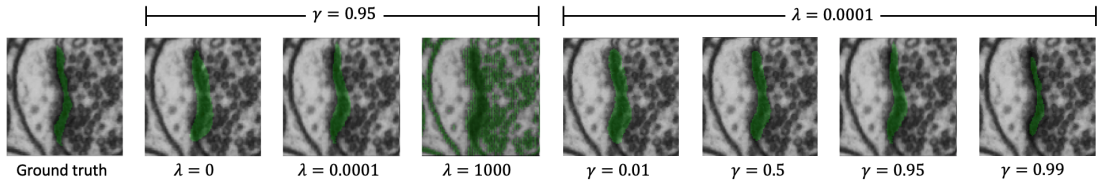


Figure 4.4: Impact of the λ parameter in the loss of Eq. 4.1 and the thresholding parameter γ. (Col. 2-4) When $\lambda = 0$, the segmentation is very similar to the template. When λ is large, there are many spurious values. (Col. 5-8) We set λ to 10^{-4} and vary γ.

width. It outputs a tensor $\hat{\mathbf{Y}}$ of dimension $D \times H \times W$ that stores the probabilities of each voxel belonging to the foreground. The second takes $\hat{\mathbf{Y}}$ as input and yields $\hat{\mathbf{X}}$, which is of the same dimension as $\mathbf{X}$. Ideally, $\hat{\mathbf{Y}}$ should be the desired segmentation and $\hat{\mathbf{X}}$ should be equal to $\mathbf{X}$.

To train these two U-Nets we minimize a weighted sum of two losses

$$\begin{aligned} \mathscr{L} &= \mathscr{L}_{ce} + \lambda \mathscr{L}_{mse} \,, \\ \mathscr{L}_{ce} &= -\sum_{i,j,k} \mathbf{Y}_{i,j,k} \log(\hat{\mathbf{Y}}_{i,j,k}) \,, \\ \mathscr{L}_{mse} &= \sum_{i,j,k} \mathbf{W}^{mse}_{i,j,k} (\mathbf{X}_{i,j,k} - \hat{\mathbf{X}}_{i,j,k})^2 \,, \end{aligned} \tag{4.1}$$

where $\mathbf{Y}$ is the rasterized template that has been fitted to the target object and λ is a scalar that controls the influence of the second loss. $\mathscr{L}_{ce}$ is a standard cross entropy loss whose minimization promotes similarity between the rasterized template $\mathbf{Y}$ and the segmentation $\hat{\mathbf{Y}}$. $\mathscr{L}_{mse}$ is a voxel-wise mean squared error in which the individual voxels are given weights $W^{mse}_{i,j,k}$ whose value is high within a distance d from boundaries in $\mathbf{Y}$ and low elsewhere.

In this scheme, the primary task is to minimize $\mathscr{L}_{ce}$ to guarantee that the segmentation is roughly correct and, at inference time, we only use the first U-Net, which we will refer to as our *core network*. To obtain the final segmentation, we threshold $\hat{\mathbf{Y}}$ using a threshold γ.

However, because the template can only be expected to provide a coarse depiction of the object, the $\mathscr{L}_{ce}$ loss does not suffice on its own. To train this core network to produce accurate boundaries, we also minimize $\mathscr{L}_{mse}$, which serves as a secondary task. As depicted by the bottom and top rows of Fig. 4.3, minimizing $\mathscr{L}_{ce}$ alone (λ=0) would result in boundaries in $\hat{\mathbf{Y}}$ that are exactly those of the template while minimizing $\mathscr{L}_{mse}$ alone (λ=1000) would produce boundaries that exist in the image but are not necessarily those we are looking for. Minimizing a properly weighted sum of the two yields segmentations that conform to the template while matching actual image boundaries, as shown in the middle row of Fig. 4.3 (λ=0.0001).

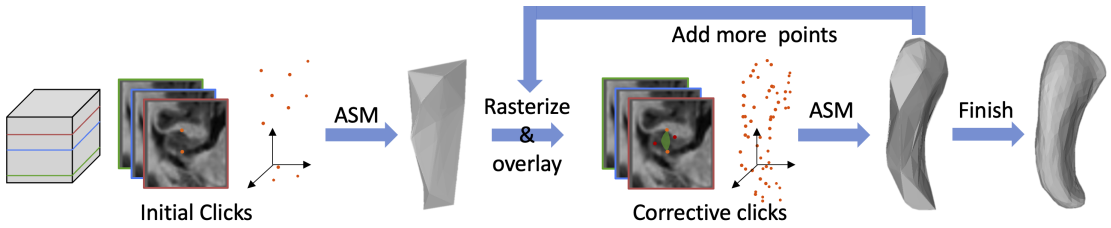


Figure 4.5: **Iterative annotation strategy**. The annotator provides a few 3D points, which are used to deform the template. The result is rasterized and overlaid on the images. The annotator can then add more points and deform again until the deformed template is close enough to the desired shape. For the hippocampus, shown here, this takes about 150 points to achieve 80% of the maximum accuracy that could be obtained using a fully annotated ground truth which consist of 1509 points on average.

Fig. 4.4 illustrates the influence of the γ and λ parameters on a real image. In practice, the results are insensitive over a wide range to how they are chosen, which we will demonstrate in the results section.

4.3.2 Template Deformation

Recall from Section 4.3.1, that the template $\mathbf{Y}$ of Eq. 4.1 should approximately match the target structure. Hence, the annotator should supply points that are distributed across the object surface. These points can then be used to deform the template so that its boundaries roughly correspond to those of the target structure. In practice, structures of genus 0 are the most common. For these, we start from a simple spherical template but more complex ones are possible, *eg* those discussed in Section 4.2.2. We do this because creating complex atlases would require fully annotated data and it would go against the objectives of this work. As we increase the number of points, we get increasingly refined templates, as shown in Fig. 4.2.

To perform this deformation interactively and in real-time, we developed the GUI interface depicted by Fig. 4.6, which relies on a recent approach to Active Surface Models (ASMs) [102] that makes full-use of GPUs and is therefore fast. It is implemented as an MITK [104] plugin. It lets the annotator supply a few points by clicking on 2D cross sections of the input image volume. The ASM then deforms the template, which is overlaid on the image data, both as 2D cross sections and a 3D surface rendering. The annotator can then add more points wherever the deformed template is too far from the target organ's boundary and iterate as often as necessary. This effectively puts the human in the loop in a painless and practical way. We illustrate this in a video that can be found in the supplementary material.

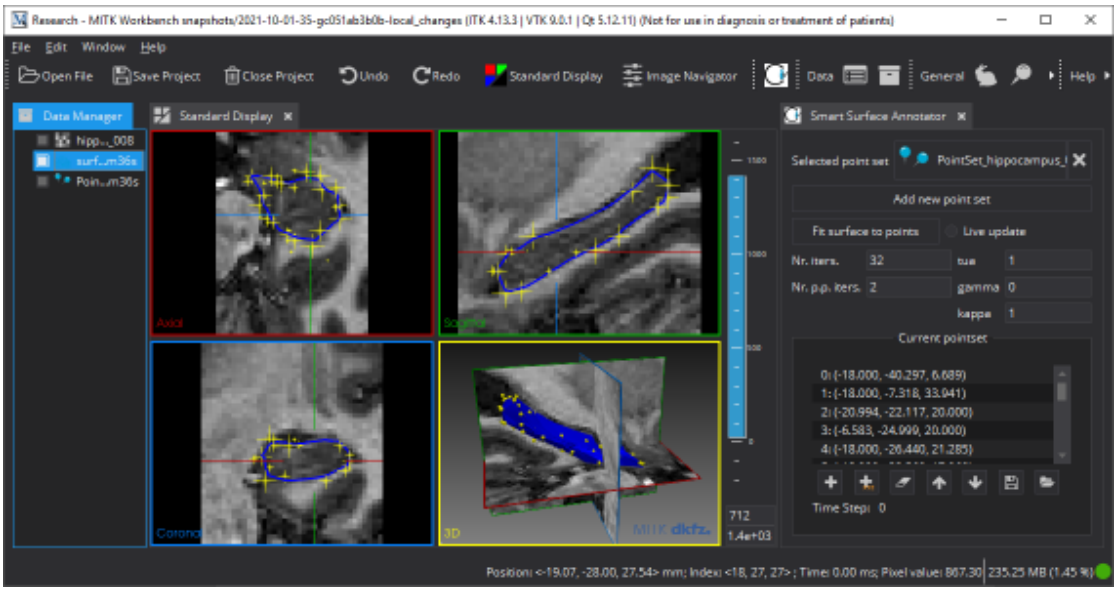


Figure 4.6: **Annotation GUI** A user can place points by clicking in the 2D image cross sections. A template is then deformed to fit the points and the user can place corrective clicks. The fitting takes less than one second so the process can be done interactively.

4.4 Experiments

4.4.1 Datasets

In our experiments, we use three datasets acquired using different modalities to highlight the generality of our approach.

Hippocampus Dataset The dataset consists of 260 labeled MRI image volumes from the Medical Segmentation Decathlon [89]. The task is to segment the hippocampus and we re-sample and pad them to obtain $64 \times 64 \times 64$ volumes. We use 130 volumes for training and the other 130 for testing. We use half of the test set as the validation set which we use for our hyper-parameter tuning and the remaining half to evaluate the final performance.

Synaptic Junction Dataset. We extract 25 $96 \times 96 \times 96$ sub-volumes centered around a synapse from a $500 \times 500 \times 200$ FIB-SEM image stack of a mouse cortex. We use the first 12 for training and the remaining 13 for testing, the task being to segment the synaptic cleft that separates the pre- and post-synaptic regions. In our experiments, we use the same hyper-parameters as for the hippocampus dataset.

Liver dataset. The dataset comprises 20 labeled CT image volumes from the CHAOS challenge [55]. The task is to segment the liver and we re-sample and pad them to obtain $64 \times 64 \times 64$ volumes. We use 10 image volumes for training and the other 10 for testing. In our experiments, we also use the same hyper-parameters as for the hippocampus dataset.

4.4.2 Metrics and Baseline

We aim to produce segmentations that are above a given quality threshold using as few annotations as possible. To formalize this goal, we use two metrics, one for quality and the other for annotation effort.

Intersection over Union. We use the standard IoU metric [19] to quantify segmentation quality.

Annotation Effort. We use the number of points provided by the annotator as a proxy for amount of effort. When the annotator provides individual points, this is clearly proportional to the time spent. When the annotator outlines contours on 2D slices, it could be argued that this overestimates the amount of effort because it is an easier task. However, in our experience it is not because precisely outlining a contour requires deliberation. The best way to do it is to use a graphics tablet and a pen, and thus requires additional hardware.

BaselineAs discussed in Section 4.2.1, the generally accepted way to provide weak-supervision for 3D image segmentation is to fully annotate a small number of 2D slices [19]. To provide a baseline, we therefore use this approach to train a 3D U-Net, as described in [19]. For a fair comparison, we use the same U-Net as in **Weak-Net**.

4.4.3 Providing Annotations

Recall from Section 4.3.2 that the annotator is expected to provide 3D points spread across the surface. As discussed in Section 4.3.2 and depicted by Fig. 4.5, we exploit the real-time performance of active shape models to allow the annotator to provide a few points, deform the template accordingly, and then add more points where the deformed shape is not satisfactory. In this section, we explore different approaches to supplying the annotations.

Randomized Annotations

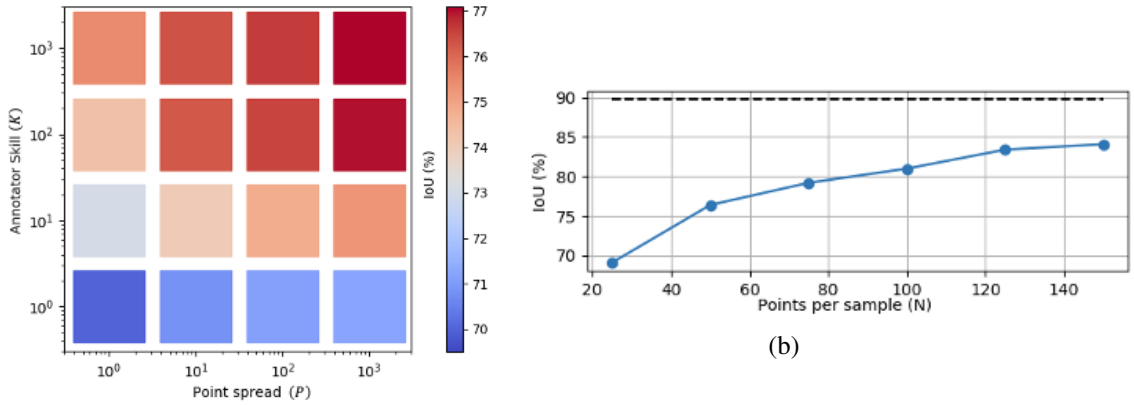
Setup. To eliminate subjectivity from our experiments, we take advantage of the fact that our datasets are fully annotated to simulate this process in the following steps.

1. **Random selection.** N points are randomly selected across the 3D surface. Ideally, they should be uniformly distributed across the surface but there is no guarantee of that. To emulate the behavior of a conscientious annotator trying to achieve this, we repeat the operation P times and select the set of N points that exhibits the largest intra-point variance. As we increase P, so does the probability that the selected points will indeed be uniformly distributed. We will refer to P as the Point Spread variable.

2. **Improved selection.** We simulate the fact that a skilled annotator will provide the most informative points possible by performing K random annotations as described above, using each one to deform the template, and selecting the one that yields the highest IoU with the ground truth. As K increases, so does the probability that the deformed template will match the ground truth well. We will refer to K as the Annotator Skill variable because it reflects the annotator's ability to select a fixed number of points to obtain the best possible result.

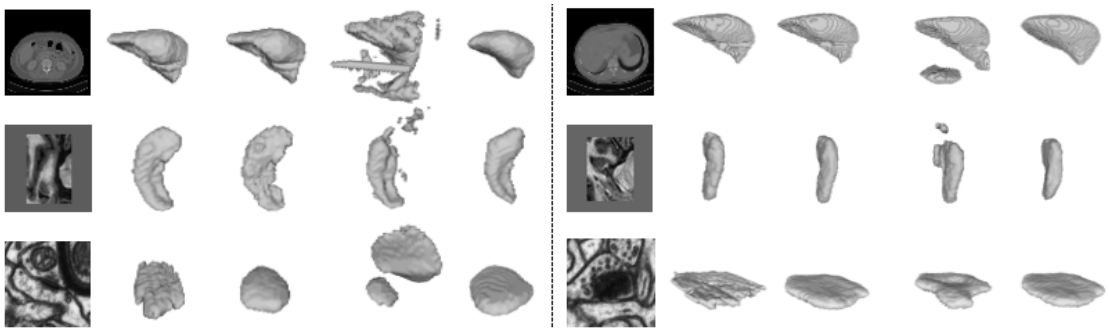
Hence our simulations are driven by three numbers, N the number of points per sample, P the number of samples we select to guarantee a good spread of the points, and K the number of attempts made to obtain a template shape that matches the ground-truth. To visualize their influence, we plot in Fig. 4.7(a) the average IoU in the liver dataset between the deformed template and the ground-truth as a function of P and K for a fixed value of $N = 50$. In Fig. 4.7(b), we plot the IoU as function of N for $P = K = 100$. Clearly, the larger P, K, and N the higher the IoU. Interestingly, even if we use *all* the ground-truth points to deform the template, the IoU does not reach 100% because more sophisticated techniques that include sub-division of facets would be required to achieve a perfect fit. In any event, with $P = K = 100$, we obtain an average IoU between 80 and 90% that represent good but far from perfect fits, which is representative of what a real-life annotator might do. We will therefore use these values in our experiments.

For the baseline, we randomly pick a number of slices from three image planes to be annotated and use the ground-truth annotations for these slices. When selecting slices, we make sure the slices contain the target object.



(a) (b)

Figure 4.7: **Annotation simulation on the Liver dataset** (a) Reconstruction accuracy (IoU) variation for different values of K and P. (b) Reconstruction accuracy (IoU) variation as a function of the number of annotated points. The black-dashed line indicates the accuracy that would be achieve by annotating all points in each sample, that is 3661 points per sample on average for the liver dataset.

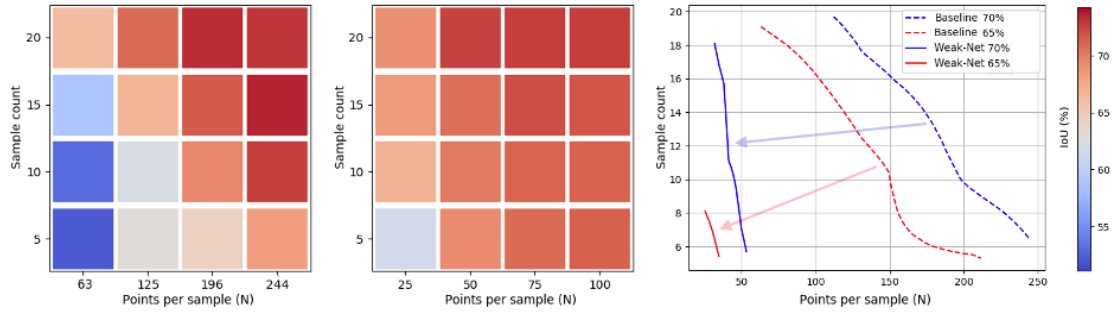


(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 4.8: **Segmentation results.** (a+f) A slice from the input volume from top to bottom (b+g) Ground truth (c+h) U-Net with full-supervision (d+i) Corresponding Baseline (e) **Weak-Net** trained with 25 points per sample. (f) **Weak-Net** trained with 125 points per sample.

Comparative Results. Given the setup described above, we now present results on the testing sets of our 3 datasets, after training with different amounts of annotations, as depicted by Fig. 4.8. When evaluating our method, we take $P = K = 100$ and vary N, the number of annotated points per sample we provide, from 25 to 100 and S the number of samples we annotate. When evaluating the baseline, we vary the number of slices we use and the number of samples we annotate. In this case, the number of annotated points per sample is the sum of the contour lengths in every slice.

In Fig. 4.9(a,b), we plot the average IoUs we obtain in this manner on the hippocampus dataset as a function of the number of points per sample and the number of annotated samples. In Fig. 4.9(c),



(a) (b) (c)

Figure 4.9: **Results on the hippocampus dataset.** (a) **Baseline** (b) **Weak-Net** (c) Annotation effort required to achieve 65% and 70% IoU. The blue and red arrows indicate the effort reduction our approach delivers.

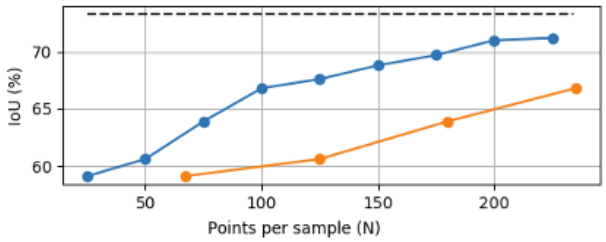
we plot the total annotation effort, that is the product of the number of points per sample and the number of samples, required to attain a target IoU, either 65% or 70%. As there are many ways to achieve a given IoU by increasing one number while decreasing the other, we draw *iso-IoU* curves. The baseline ones are dashed and ours are full and clearly to the left of the dashed ones. In other words, we need significantly less effort to achieve a similar result.

In Fig. 4.10, we report our results on the other two datasets that contain fewer samples. Hence we used all the train samples and plot the IoU as a function of the number of points per sample. For the same number of points, our approach consistently outperforms the baseline. Note also, that in the baseline method it is not possible to annotate less than one slice and therefore to reduce the annotation effort below a certain level given by the size of the contours in the annotated slice. Because we can annotate points randomly, we can actually annotate fewer, which is why the blue curves extend further to the left than the yellow ones.

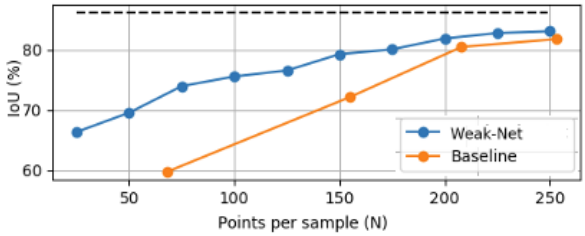
Human Annotations

In practice the ground-truth will not be available and the annotations will be provided by people. We test this more realistic scenario on the Hippocampus and Liver datasets.

Setup. We used the GUI interface of Fig. 4.6 to manually annotate a subset of images using the following protocol. The annotator first supplied an initial 4 points before fitting a template and then additional ones where the fitted model seemed most distance from the target organ. As it only takes 150 to 900 ms to perform the fit, the template was updated after each new point. For the hippocampus, at most 100 points could be specified, and the liver 250. However, the annotator could stop before entering all the allowed points. To ensure errors are only caused by



(a)



(b)

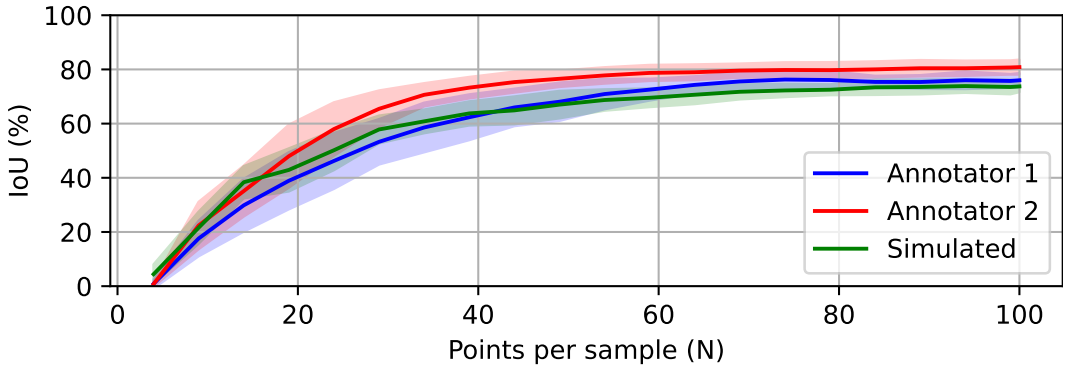
Figure 4.10: **IoU as function of the number of points used (N).** (a) Synaptic Junction dataset. (b) Liver dataset. The dashed line denotes fully supervised results.

poor template fitting and not by annotator mistakes, the ground truth labels are overlaid on the image for visual inspection while placing the points. In the end, 2 different annotators annotated 16 hippocampus images and 5 liver images. To estimate the time for full 3D annotation, xy-slices in one Hippocampus image and one Liver image were manually delineated. For *Hippocampus*, all xy-slices were annotated. For *Liver*, 5 evenly spaced slices were annotated and the average time was multiplied with the span of the liver along the z-axis. The annotations were performed at the original resolution of the datasets and the IoUs were also computed at this resolution.

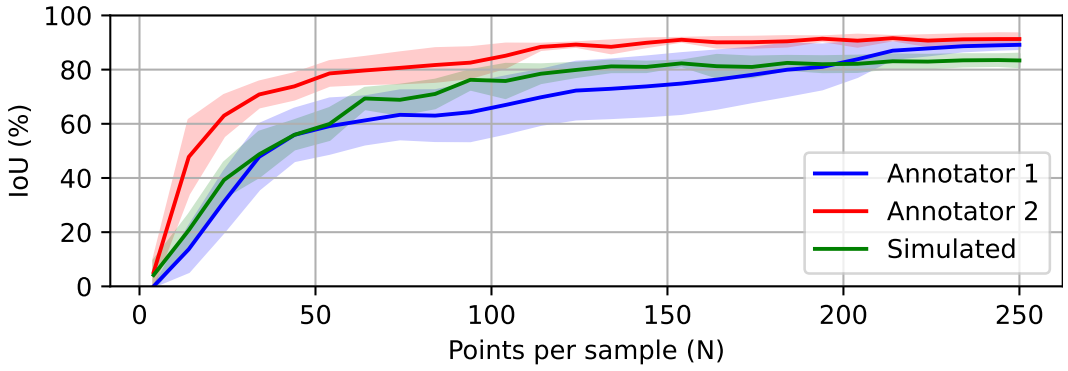
Table 4.1: **Human annotation results.** Values are given as mean±std. Times are in minutes.

		Hippocampus	Liver
Annotator 1	Max. IoU (%)	78±3.0	89±1.5
	Time	3.66±0.60	9.17±1.75
Annotator 2	Max. IoU (%)	81±2.6	92±0.7
	Time	5.72±1.31	15.5±1.47
Est. Full Annot. Time		6.65±0.65	125±19.3

Comparative Results. In Fig. 4.11 and Tab. 4.1, we report IoU as a function of the number of points supplied. The performance is roughly comparable to what we obtain with the simulated annotations of Section 4.4.3 but human-generated consistently deliver a higher IoU. However, in the early part of the annotation process the IoU achieved by human annotators varies significantly from dataset to dataset and annotator to annotator. Regarding time, as the Hippocampus volumes are small, point annotation is only ~1.5x faster than annotating the full volume. For the large



(a)



(b)

Figure 4.11: **Template accuracy** as a function of the number of human or simulated point annotations (N). (a) Hippocampus. (b) Liver.

Liver volumes however, point annotation is ~10x faster, which is significant.

4.4.4 Limitations

It could be argued that even 25 or 50 points per sample is significant annotation effort. More sophisticated templates that are parameterized by low-dimensional latent vectors can therefore be deformed by specifying even fewer 3D points than we do now and can be developed as future work.

4.4.5 Ablation Study

In this section, we first study the influence of the λ and γ hyper-parameters introduced in section 4.3 and then compare our results against those obtained using full supervision.

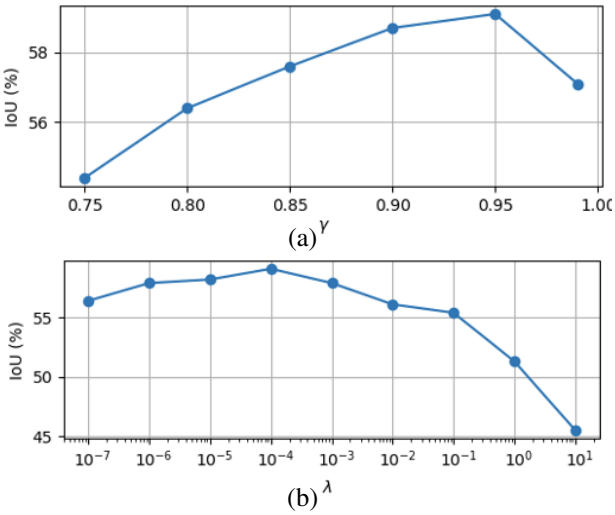


Figure 4.12: **Influence of λ and γ.** IoU (%) as a function of λ in (a), γ in (b) on the Synaptic Junction dataset with $N = 25$.

Table 4.2: Comparative results

	Hippo.	Liver	Syn. Junction
Baseline	71.2 ± 0.9	81.2 ± 0.8	68.2 ± 0.9
Weak-Net	73.4 ± 0.8	83.1 ± 1.1	71.2 ± 1.2
Full anno.	79.3 ± 0.4	87.3 ± 0.3	73.3 ± 0.6

Impact of Hyper Parameters

We plot the variation of IoU against different values of λ and γ for the Synaptic Junction dataset.

As can be seen in Fig. 4.12, λ can be chosen in the range 10^{-6} to 10^{-3} without any major change in performance. In practice, we used $\lambda = 10^{-4}$ to produce the results shown Section 4.4 unless otherwise specified. Similarly for γ, we observe a peak in the range of 0.9 to 0.99. In practice, we used $\gamma = 0.95$.

Weak vs Full Supervision

We compare **Weak-Net** and the baseline against a U-Net trained with full supervision and report the results in Table 4.2. Our approach delivers 92 to 95% of the accuracy that can be achieved with full supervision with only 2 to 10% of the annotation effort for all 3 datasets.

4.5 Conclusion

We have presented a weakly supervised approach to segmenting 3D image volumes that outperforms more traditional approaches that rely on fully annotating individual 2D slices.

It relies on deforming simple spherical templates that incorporate no shape prior. In future work, to further reduce the annotation burden, we will develop more sophisticated templates that are parameterized in terms of low-dimensional latent vectors and can therefore be deformed by specifying even fewer 3D points than we do now.

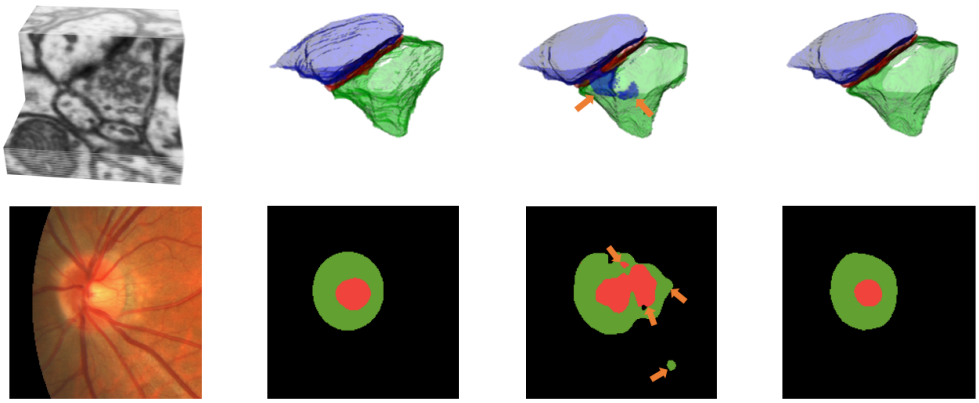
5 Probabilistic Atlases to Enforce Topological Constraints

5.1 Introduction

Probabilistic atlases (PAs) are widely used for multi-atlas based segmentation[48]. With the advent of deep learning, there has been a push to incorporate them with Convolutional Neural Networks (CNNs) as well [6, 47, 92, 93]. The published techniques relying on deep learning share a number of features: They work best for structures featuring relatively small variations in shape and position; the atlases are often created by fusing multiple manually annotated images; the atlases must also be pre-registered to the target images to align them with the structures of interest.

Thus, the PAs have been mostly used to segment structures such as the brain or heart for which the above requirements can be met. In this paper, we propose an approach to design and handle PAs that makes them usable in complex situations where the shape and position can vary dramatically, such as those shown in Fig. 5.1. Our PAs are coarse and only encode the relative position and topology of the structures we expect to find. To register them, we rely on affine transforms whose parameters are estimated at the same time as the segmentations themselves using the end-to-end trainable encoder-decoder architecture depicted by Fig. 5.1, which we will refer to as **PA-Net**. The affine transforms are used to warp the atlases and feed the features of the warped atlases to the decoder. This differs significantly from earlier approaches [92, 93] that rely on pre-registered atlases.

We validate our approach on segmentation of synaptic junctions in Electron Microscopy (EM) images and optic nerve head segmentation in retinal fundus images. A synaptic junction has an arbitrary shape but always features a synaptic cleft sandwiched between a pre-synaptic bouton



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

Figure 5.1: Eliminating topological mistakes. (a) A small FIBSEM image stack featuring a synaptic junction and a retinal fundus image. (b) Ground-truth segmentations. The pre-synaptic, synaptic-cleft, and post-synaptic regions shown in green, red, and blue respectively. Similarly, the optic disk and cup shown in green and red, respectively. (c) U-Net segmentations with orange arrows pointing to topological mistakes. (d) Our error-free result (best viewed in color).

belonging to the axon of one neuron and a post-synaptic dendritic spine belonging to another. In this case, the shape complexity and unpredictability precludes the use of standard PAs. Optic nerve head consists of an optic disk and optic cup. Even though their shape does not vary significantly, there are significant variations in the size of the optic cup and the position of the optic nerve head.

We will show that **PA-Net** eliminates most of the topological mistakes its unaided V-Net [73] or U-Net [85] backbone makes on the EM and retinal images, while being generic and potentially applicable to other segmentation tasks. Our contribution is therefore a novel approach that makes it possible to use probabili- -stic atlases even in situations where shape variability is too great for existing approaches and methods to integrate the atlas registration process directly into the network. The code is publicly available[1].

5.2 Related Work

PAs are commonly used in standard atlas-based segmentation approaches. In their piplines, CNNs are indirectly used to perform segmentation. These include using a CNN to infer deformation maps to register target images to atlas images [7] and using it to implicitly infer the segmentation masks[23, 44]; using CNNs to improve the selection of best atlases from a library [54]. As this is

[1] https://github.com/cvlab-epfl/PA-net.git

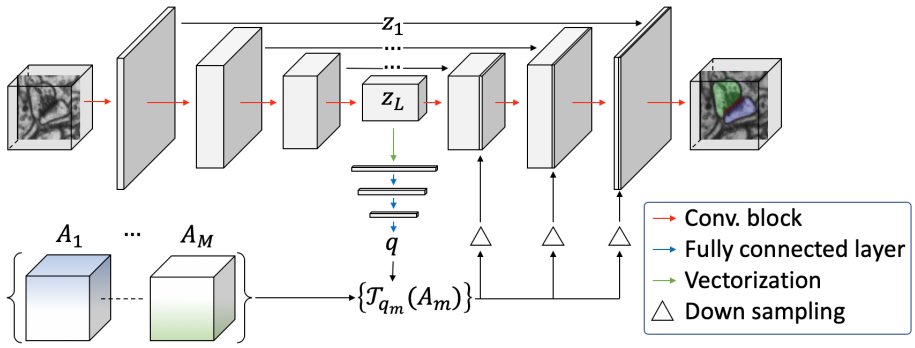


Figure 5.2: **PA-Net** architecture. The network takes as input an image and a list of PAs. Given the decoder output, the fully connected layers estimate the parameters of the affine transformation that registers the PAs to the input. The registered PAs are then concatenated with the features computed by the convolutional decoder at each scale.

not the focus of this paper, we will not discuss them further and will concentrate on methods that directly use the PAs to assist a CNN that perform segmentation.

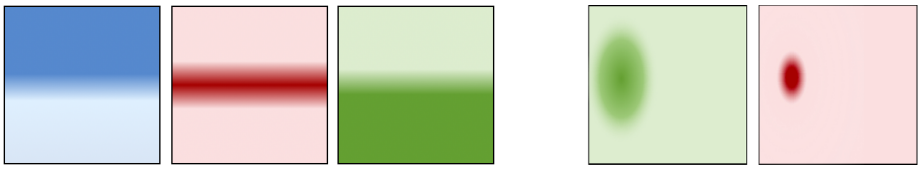
A PA can provide localization priors to help a network find the target of interest. Such PAs are often referred to as seed layers. They come in two flavors, Gaussian priors [8, 108] or binary seed layers [51]. Another popular way to focus the attention of the network is to provide an attention mask [29], which then serves the same purpose as a PA. However, the attention masks are derived from the input data itself and does not act as an external knowledge source.

PAs built by fusing multiple manually annotated images can be used to introduce even more prior knowledge about shape and topology of structures. In [92, 93], this is done to improve segmentations of human brain by having the CNN take pre-registered PAs as input. In our work, we extend the idea by demonstrating that PAs can be used to introduce topological knowledge even when the structure of interest exhibits large shape variations and without pre-registration that the whole system is end-to-end trainable.

5.3 Approach

5.3.1 Network Architecture

The architecture of the proposed **PA-Net** is depicted in Fig. 5.2. Its backbone is an encoder-decoder architecture similar to the one used in popular networks such as U-Net [85] or V-Net [73]. These networks are designed to learn to approximate the distribution $p(\mathbf{Y}|\mathbf{X})$, where $\mathbf{X}$ is the input image and $\mathbf{Y}$ is the output segmentation. We extend the idea by learning the joint distribution



(a) (b)

Figure 5.3: Probabilistic atlases. (a) For synaptic junction segmentation, they are cubes and we show a single face. (b) For optic optic disk and cup segmentation, they are regular 2D arrays. The color hues denote the same areas as those in Fig. 5.1. The color saturation is proportional to the probability at a given location (best viewed in color).

$p(\mathbf{Y}, T_{\mathbf{q}}(\mathbf{A})|\mathbf{X},\mathbf{A})$. The function $\mathscr{T}_{\mathbf{q}}(\cdot)$ applies affine transformations to the set of PAs $\mathbf{A}$ given the pose vectors $\mathbf{q}$. We factorize this joint distribution as

$$p(\mathbf{Y}, T_{\mathbf{q}}(\mathbf{A})|\mathbf{X},\mathbf{A}) = p(\mathbf{Y}|\mathbf{X}, T_{\mathbf{q}}(\mathbf{A}))\, p(\mathbf{q}|\mathbf{X},\mathbf{A}) \tag{5.1}$$

PA-Net models it using a primary stream, the encoder-decoder backbone at the top of Fig. 5.2, and a secondary stream, the fully connected layers at the bottom of Fig. 5.2. The secondary stream $\mathbf{F}(\cdot,\theta_f)$ originates from the latent vector produced by the final stage of the encoder $\mathscr{E}(\cdot,\theta_e)$, estimates the affine parameter vector $\mathbf{q}$, uses it to warp the atlases, and feed them to the various layers of the decoder $\mathscr{D}(\cdot;\theta_d)$ after rescaling them to the resolution of feature vectors at each stage. Here θ_e,θ_d and θ_f are the learned weights that control the behavior of $\mathscr{E},\mathscr{D}$ and $\mathbf{F}$. Given the encoder output, $\mathbf{F}$ and $\mathscr{D}$ model $p(\mathbf{Y}|\mathbf{X}, T_{\mathbf{q}}(A))$ and $p(\mathbf{q}|\mathbf{X}, A)$, respectively. This guarantees that these two distributions are learned from the same features. Training the encoder to produce features that are effective for both tasks yield improved performance.

5.3.2 Atlas Design

Fig 5.3. depicts the topological atlases we use to segment synaptic junctions and optic disks and cups. In the first case, they are cubes and in the second, ordinary 2D arrays. They encode the probability of any pixel of voxel to belong to one of the possible classes—pre-synaptic, cleft, or post-synaptic for synapses and optic disk or cup for retinas—given its pose.

5.3.3 Atlas Registration

Parameters necessary for PA registration is computed by estimating the pose vector q. When working with image cubes such as the one shown at the top of Fig. 5.1, we take the pose vector

Table 5.1: Comparative results for synaptic junction segmentation.

	Jaccard index (%)				TER
	Pre-synaptic	Synaptic junction	Post-synaptic	Mean	
V-Net	**63.0**	56.0	80.6	66.5	5/15
PA-VNet	59.5	**57.8**	**83.1**	**66.8**	**1/15**
U-Net	**73.6**	59.7	79.1	70.7	7/15
Naive PA-UNet	73.4	58.0	78.4	69.9	7/15
PA-UNet	72.6	**62.8**	**87.0**	**74.1**	**3/15**

to be $\mathbf{q} = [t_x, t_y, t_z, s_x, s_y, s_z, r_x, r_y, r_z]$, where $t_{\{x,y,z\}}$ represent translations and $s_{\{x,y,z\}}$ scalings in the x,y, and z directions, respectively. We represent the orientation using the angle-axis vector $[r_x, r_y, r_z]$, whose direction is the axis of rotation. When working with 2D images, we drop t_z, s_z, and r_z from $\mathbf{q}$.

5.3.4 Loss Functions

Following standard practice, we formulate the loss terms as the negative log likelihood of the joint distribution $p(\mathbf{Y}, T_{\mathbf{q}}(A)|\mathbf{X}, \mathbf{A})$ of Eq. 5.1. This yields the composite loss

$$\mathscr{L} = \mathscr{L}_{seg} + \mathscr{L}_{pose} \,. \tag{5.2}$$

$\mathscr{L}_{seg}$ is the standard cross entropy loss that evaluates segmentation performance and take $\mathscr{L}_{pose}$ is the least square error in estimating the pose $\mathbf{q}$.

5.4 Results and Discussion

5.4.1 Datasets

Synaptic Junction Dataset:

It is a $500 \times 500 \times 200$ FIB-SEM image stack of a mouse cortex. We used 50 xy slices for training, 50 for validation, and 100 for testing. From each set, we cropped $96 \times 96 \times 96$ image volumes containing a synaptic junction such as the one shown in the top row of Fig. 5.1(a) and they are zero-padded as necessary. This gave us 13, 10 and 15 volumes for training, validation and testing, respectively. The synapse is not necessarily perfectly centered and the task is to segment pre-synaptic region, post-synaptic region, and synaptic cleft.

Table 5.2: Comparative results for fetinal fundus image segmentation.

	Jaccard index (%)			TER
	Optic disk	Optic cup	Mean	
V-Net	78.4	72.5	75.5	7/200
PA-VNet	**79.0**	**73.0**	**76.0**	3/200
U-Net	81.3	73.8	77.6	18/200
Naive PA-UNet	80.8	74.1	77.3	16/200
PA-UNet	**81.6**	**74.6**	**78.1**	**5/200**

Retinal Fundus Image Dataset:

It comprises 400 2124×2056 retinal fundus images acquired using a Zeiss Visucam 500 camera [77], which we resize and pad to be 512×512. We use 100 for training, 100 for validation, and 200 for testing.

5.4.2 Quantitative and Qualitative Results

We evaluate segmentation performance in terms of the standard Jaccard Index and of an additional metric we dub the *Topological Error Ratio*(TER). We define TER as the ratio of segmentations containing topological errors to the total number of test images. A segmentation is considered to contain a topological error if it violates the expected topology of the target structure, that is, it features semantic labels appearing where they should not given the topological constraints. We compute this value by finding the connected components of the final segmentation and then finding instances that violate the topology.

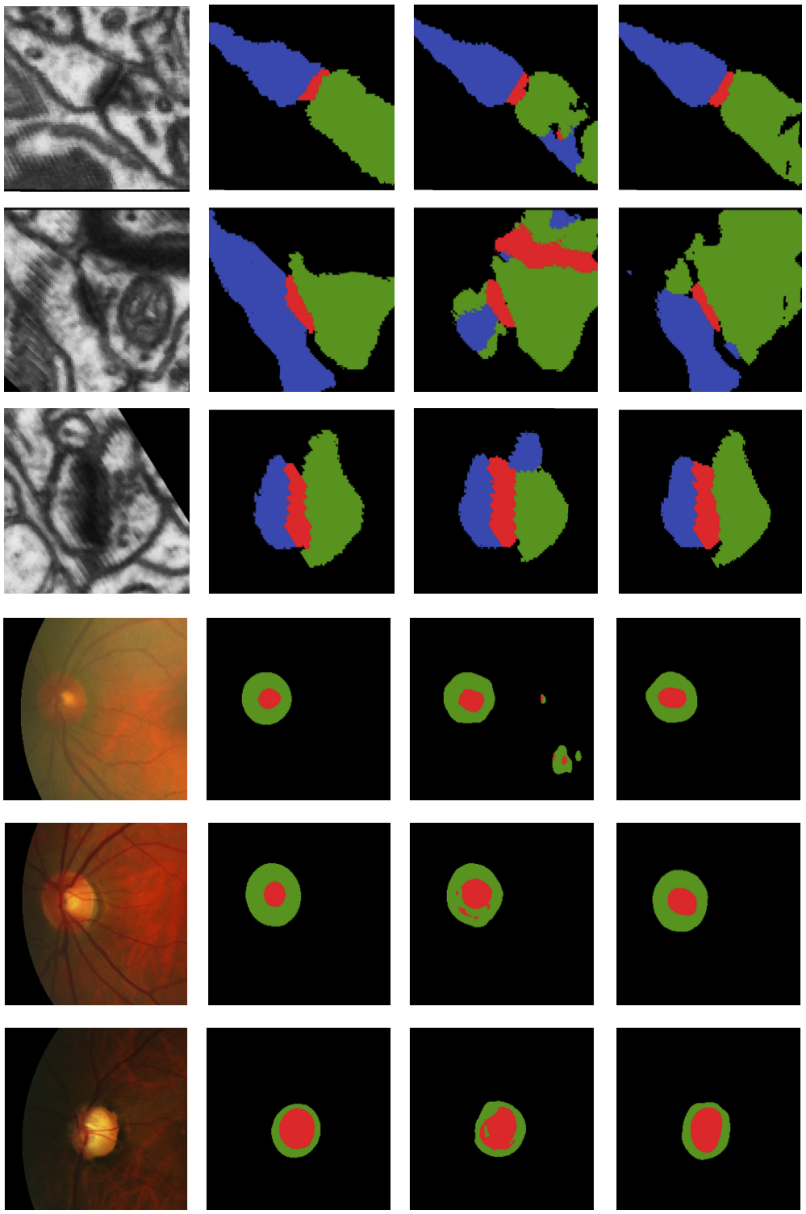
We tested two versions of **PA-Net**, one based on the U-Net [85] architecture and the other based on the more recent V-Net [73]. We compare the results against those of the standard U-Net and V-Net. We report our comparative results on our two datasets in Tables 5.1 and 5.2. Figs. 5.4 depict them qualitatively (see supplementary materials for further results). Using the atlas consistently improves over baseline performance. The gains are most visible in TER terms because our **PA-Net**s truly come into their own when the standard U-Net and V-Net fail, which is only a fraction of the time, as would be expected of architectures that are as popular as they are. Even when it fails to eliminate topological errors, it reduces the size of the errors as shown in the second row of Fig. 5.4). The one exception is the pre-synaptic region, for which the Jaccard numbers decrease. A closer inspection of the results show that the atlas sometimes encourages the segmentation to leak from weak boundary regions in few instances. We plan to address this issue by introducing an additional stream that specifically focus on predicting object boundaries

and combining its results with **PA-Net**.

To demonstrate the importance of using the same features to compute the segmentation *and* the pose parameters, we implemented a naive version of our approach that uses two *separate* streams to compute the pose and the segmentati- -on features, which we denote as Naive PA-Unet in Tables 5.1 and 5.2. The naive version fails to resolve the topological mistakes in the datasets. In the synaptic junction dataset, this occurs because the naive version has a mean orientation estimation error of 80.6^o in contrast to the **PA-Net**'s error of 10.4^o. In the retinal fundus image dataset, the naive version has a mean localization error of 26.5 pixels compared to a 3.9 pixel error for **PA-Net**. As a result, in both instances, network fails to properly register PAs.

5.5 Conclusion

We have proposed an encoding-decoding architecture that can exploit rough atlases that encode the topology of structures that can appear in any pose and have arbitrarily complex shapes to improve the segmentation results. One of its crucial components is that it relies on the output of the encoder to compute both the pose parameters used to deform the atlas and the segmentation mask itself, which makes it effective and end-to-end trainable. As future work, we plan to extend **PA-Net** by introducing an additional stream to produce edge maps and using it to address the minor loss in accaracy that occur when the structures have weak boundaries.



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

Figure 5.4: U-Net vs **PA-Net**. Top three rows depict results from synaptic junction segmentation. The bottom three rows depict results from retinal fundus image segmentation. (a) Input images, (b) Ground-truth, (c) U-Net results (d) **PA-Net** results. The color hues denote the same areas as those in Fig. 5.1 (best viewed in color).

6 Conclusion

In this thesis, we propose methods to incorporate object priors in volumetric image segmentation models and demonstrate its effectiveness in improving the accuracy, quality and the speed of annotation in volumetric image segmentation. In this chapter, we present a brief summary of this thesis and discuss future research directions.

6.1 Summary

In chapter 2, we propose **Voxel2Mesh**, an end-to-end trainable architecture that can produce surface meshes from input image volumes. We demonstrate the effectiveness of the approach to easily introduce topological priors that improve the accuracy of the predictions compared to state-of-the-art methods that only use voxel representations. Furthermore, we demonstrate that, the method can directly produce surface meshes useful for surface analyzing downstream tasks, instead of having to perform post-processing steps to obtain the surface meshes from voxel representations.

In chapter 3, we focus on enforcing object priors related local surface properties. We propose **DASM**, an alternative mechanism inspired by Active Surface Models to enforce priors on smoothness, tension and skewness which let us achieve better trade-offs between the accuracy and the quality of the surface meshes produced. We demonstrate the effectiveness of the approach with **Voxel2Mesh** which produce surface meshes from volumetric images. We also demonstrate its effectiveness with **Mesh R-CNN**[38] which is a state-of-the-art method in producing surface meshes from 2D images.

In chapter 4, we continue using surface meshes and use its ability to easily introduce object priors

on topology and smoothness to propose **Weak-Net** which consist of an interactive annotation tool for obtaining weak annotations with less effort and a 3D CNN architecture that can be trained with the produced annotations. With the proposed approach, the annotator only has to click a small subset of points on the surface of the target object and we use the Active Surface Model proposed in chapter 3 to deform a template mesh to fit to the given set of points.

In chapter 5, we focus on voxel representation and propose a new use case for PAs within the context of deep learning. Traditionally they have been used with structures that has relatively small variance in shape between samples. But, in this work we demonstrate how we can use PAs to introduce object priors even when we observe relatively larger variations in the shape and achieve better performance.

6.2 Future Work

Object priors and prior knowledge in general will continue to play an important role in volumetric image segmentation. Given the limitations in training data, prior knowledge will remain as one of the approaches to further improve the accuracy, reliability and the quality of the segmentations so that they can have a significant real world impact.

If we focus on approaches proposed in this thesis, one direct avenue of future work is in improving the proposed architectures to further improve their performance and applying them in different settings. The other avenues is to explore alternative mechanisms to introduce object priors in to volumetric image segmentation. Next, we discuss these two avenues.

Voxel2Mesh performance can be further improved by introducing better mesh-division techniques that achieve better trade-offs between accuracy and the number of vertices in the mesh which determines the memory consumption of the model. In terms of different applications, **Voxel2Mesh** could be trained with sparse partial annotations and could be developed into a model that could be trained with partial labels. Extending **Voxel2Mesh** to segment high resolution images is also a possibility since mesh representation is less memory consuming compared to voxel representation. Another direction is extending **Voxel2Mesh** to segmenting structures with varying topology and thin structures.

In **DASM**, adaptive smoothing can be further improved to achieve more accurate meshes. Instead of the current metric used to determine the amount of smoothing, it could also be learnt from data. This has been done with Active Contours [69] and it could be extended to **DASM** as well. The principal in **DASM** could also be extended to volumetric meshes.

Interactive annotation tool in **Weak-Net** can be improved by introducing mesh sub-division in between mesh deformation steps. This will enable obtaining more accurate annotations when the annotator inputs larger number of points. Furthermore, a latent code corresponding to a target object can be optimized so that the surface mesh is indirectly moved to fit to a given set of points instead of directly moving the surface mesh as it is done now. This can help to reduce the number of points user has to input to reach a given accuracy, specially when working with smaller number of points.

One of the alternative avenues that could be explored for introducing object priors is using implicit surfaces for volumetric image segmentation. Implicit surfaces can also introduce most of the object priors that could be introduced with surface meshes. At the same time, recent works by [90] demonstrate the ability of the implicit surfaces to capture very high frequency details of objects. Therefore, it could be applied for volumetric image segmentation and might be able to surpass the accuracy of the meshes produced by methods such as **Voxel2Mesh**.

A Appendix - Deep Active Surface Models

A.1 Appendix

A.1.1 Active Surface Model: Continuous Formulation

Our objective is to minimize the total energy **E** in Eq. 3.2. There is no analytical solution for the global minimum of **E**. But, as mentioned in Section 3.3.1, any local minimum must satisfy the associated Euler-Lagrange equation given in Eq. 3.4. To find a surface that does this, surface evolution is used by introducing a time t parameter into 3.4 and writing

$$\frac{\partial v(s,r,t;\Phi)}{\partial t} + L(v(s,r,t;\Phi)) = F(v(s,r,t;\Phi)) \,, \tag{A.1}$$

where $L(v(s,r,t;\Phi))$ is the R.H.S of Eq. 3.4.

Solving A.1, requires specifying an initial surface. Earlier approaches [20, 53] used a manual initialization, whereas in [69, 15] another model is used to predict the initial curve. To ensure the reached local minima corresponds to the desired curve, these approaches require the initialization to be close to the target shape. In DASM, we rely instead on the graph-convolution layers to provide a good initialization.

A.1.2 Active Surface Model: Discrete Formulation

In the continuous formulation of Section 3.3.1, computing the solution to Eq. 3.4 requires computing the derivatives of order 2 and 4 for the mapping v of Eq. 3.1. To compute them in practice, we discretize the surface and use finite difference equations to estimate the derivatives.

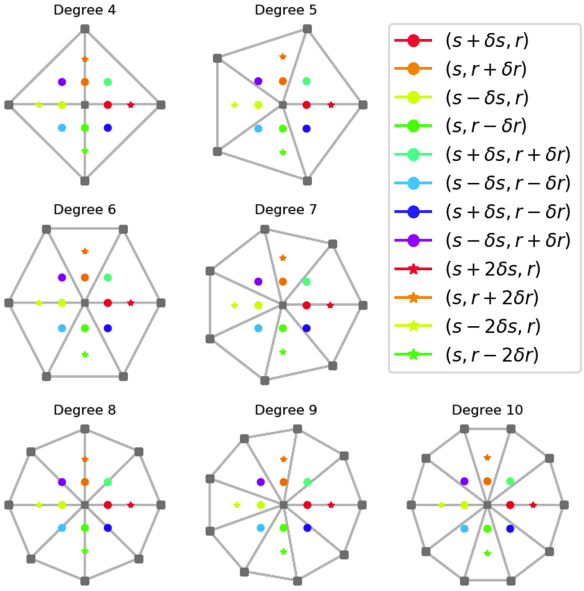


Figure A.1: **Finite Differences.** Relative positions of $(s+k_1\delta, r+k_2\delta)$ terms w.r.t (s,r) which is at the center and its 1-ring neighbors from degree 4 to 10.

Given a small value of δs, finite-difference approximations for the derivatives w.r.t s can be written as,

$$\frac{\partial v}{\partial s} \approx \frac{1}{\delta s}[v(s+\delta s, r) - v(s,r)]\,,$$
$$\frac{\partial^2 v}{\partial s^2} \approx \frac{1}{\delta s^2}[v(s+\delta s, r) - 2v(s,r) + v(s-\delta s, r)]\,,$$
$$\frac{\partial^3 v}{\partial s^3} \approx \frac{1}{\delta s^3}[v(s+2\delta s, r) - 3v(s+\delta s, r) + 3v(s,r) - v(s-\delta s, r)]\,,$$
$$\frac{\partial^4 v}{\partial s^4} \approx \frac{1}{\delta s^4}[v(s+2\delta s, r) - 4v(s+\delta s, r) + 6v(s,r) - 4v(s-\delta s, r) + v(s-2\delta s, r)]\,,$$

Similarly, we can write finite difference equations w.r.t r as well.

Now to compute these approximations, we need to compute $v(s+\delta s)$ and other similar terms. Let us therefore take (s,r) be the 2D coordinates that v maps to the coordinates of a specific vertex. In an irregular grid, $(s, r+\delta r)$, $(s+\delta s, r)$, or any of s, r coordinates that appear in the derivative computations will in general *not* be be mapped to another vertex for any choice of $\delta s, \delta r$. Fig. A.1 illustrates their actual positions depending on the number of neighbors the vertex has.

We can nevertheless compute the 3D coordinates they map to as follows. Let us first consider the 3D point $v(s+\delta s, r)$ that $(s+\delta s, r)$ gets mapped to and it is depicted by orange circle in Fig. 3.2. For δs small enough, it belongs to a facet of which $v(s,r)$ is a vertex and let $v(s_1, r_1)$ and $v(s_2, r_2)$ be the other two. We can compute the barycentric coordinates λ, λ_1, and λ_2 of $v(s+\delta s, r)$ in that

facet by solving

$$\begin{bmatrix} s+\delta s \\ r \\ 1 \end{bmatrix} = \begin{bmatrix} s & s_1 & s_2 \\ r & r_1 & r_2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda \\ \lambda_1 \\ \lambda_2 \end{bmatrix} . \tag{A.2}$$

Given these barycentric coordinates, we can now estimate $v(s+\delta s, r)$ as

$$\lambda * v(s,r) + \lambda_1 * v(s_1,r_1) + \lambda_2 * v(s_2,r_2) , \tag{A.3}$$

which allows us to estimate $\frac{\partial v}{\partial s}$ according to the above finite-difference equations. For this approximation to be valid, we pick δs such that all terms in finite-difference expressions lie within the 1-ring neighborhood of $v(s,r)$. We can repeat the process for all the other expressions involving δs in these equations and, hence, compute all required derivatives. Regular square and hexagonal grids are special cases in which these computations can be simplified.

A.1.3 Matrix Inversion using Neumann Series

We are approximating the inverse of $(\mathbf{A}+\alpha\mathbf{I})^{-1}$ using the Neumann series given in Eq. 3.7. In Fig A.2, we plot both RMSE in estimating the inverse and the time it takes to perform the estimation as a function of K. Given the trade off between running time and accuracy, we pick $K = 4$ for the estimation.

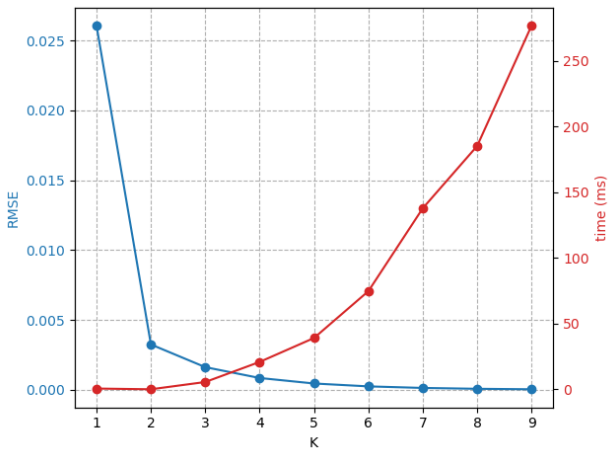


Figure A.2: **Neumann Series Approximation.** RMSE between the approximated inverse and the true one in blue and computation time in red as function of K. $K = 4$ gives an acceptable trade-off between the two.

A.1.4 Quantitatively Measuring Mesh Regularization with Consistency Metrics

All the metrics used in Sec. 3.4 evaluate the accuracy of the meshes. We use them because they are the standard metrics used in the literature. But to get a better understanding of the quality of the meshes, we provide two more metrics; mean edge length and mean surface Laplacian. We observe that around abnormalities such as those highlighted by orange arrows in Fig. 3.5, 3.7, the edge lengths and surface Laplacians tend to increase significantly. This increases mean edge length and mean surface Laplacian and its effect can be seen in Table A.1.

Table A.1: **Results on ShapeNet** as a function of λ_{edge}. Note that this is an extension of Table 3.2.

$\lambda_{e.}$		Chf. (↓)	Edg. Length	Surf. Lap.
1.0	Mesh R-CNN	0.232	0.023 ± 0.011	0.033 ± 0.031
	Ad.-DASM	**0.231**	0.022 ± 0.009	0.023 ± 0.019
0.6	Mesh R-CNN	0.212	0.024 ± 0.015	0.045 ± 0.045
	Ad.-DASM	**0.206**	0.023 ± 0.011	0.029 ± 0.020
0.2	Mesh R-CNN	0.189	0.028 ± 0.021	0.066 ± 0.075
	Ad.-DASM	**0.183**	0.025 ± 0.015	0.037 ± 0.049
0.0	Mesh R-CNN	**0.144**	0.141 ± 0.142	0.708 ± 0.671
	Ad.-DASM	0.167	0.070 ± 0.072	0.254 ± 0.276

Bibliography

[1] Acuna, D., Kar, A., and Fidler, S. (2019). Devil is in the Edges: Learning Semantic Boundaries from Noisy Annotations. In *Conference on Computer Vision and Pattern Recognition*.

[2] Akuna, D., Ling, H., Kar, A., and Fidler, S. (2018). Efficient Interactive Annotation of Segmentation Datasets with Polygon-RNN++. In *Conference on Computer Vision and Pattern Recognition*.

[3] andl O. Russakovsky, A. B., Ferrari, V., and Fei-Fei, L. (2016). WhatâĂŹs the Point: Semantic Segmentation with Point Supervision. In *European Conference on Computer Vision*.

[4] Arnab, A., Jayasumana, S., Zheng, S., and Torr, P. H. S. (2015). Higher Order Potentials in End-To-End Trainable Conditional Random Fields. *CoRR*, abs/1511.08119.

[5] Arnab, A., Zheng, S., Jayasumana, S., Romera-Paredes, B., Larsson, M., Kirillov, A., Savchynskyy, B., Rother, C., Kahl, F., and Torr, P. H. S. (2018). Conditional Random Fields Meet Deep Neural Networks for Semantic Segmentation: Combining Probabilistic Graphical Models with Deep Learning for Structured Prediction. *IEEE Signal Processing Magazine*, 35(1):37–52.

[6] Atzeni, A., Jansen, M., Ourselin, S., and Iglesias, J. (2018). A Probabilistic Model Combining Deep Learning and Multi-Atlas Segmentation for Semi-Automated Labelling of Histology. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[7] Balakrishnan, G., Zhao, A., Dalca, A., Durand, F., and Guttag, J. (2018). Synthesizing Images of Humans in Unseen Poses. In *Conference on Computer Vision and Pattern Recognition*.

[8] Bertasius, G., Park, H. S., Yu, S. X., and Shi, J. (2017). Unsupervised Learning of Important Objects from First-Person Videos. In *International Conference on Computer Vision*.

[9] Bielski, A. and Favaro, P. (2019). Emergence of Object Segmentation in Perturbed Generative Models. In *Advances in Neural Information Processing Systems*.

[10] Chang, A., Funkhouser, T., G., L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., and Yu, F. (2015). Shapenet: An Information-Rich 3D Model Repository. In *arXiv Preprint*.

[11] Chen, M., Artieres, T., and Denoyer, L. (2019). Unsupervised Object Segmentation by Redrawing. In *Advances in Neural Information Processing Systems*.

[12] Chen, X. and Williams, R. (2019). Learning Active Contour Models for Medical Image Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[13] Chen, Z. and Zhang, H. (2019). Learning Implicit Fields for Generative Shape Modeling. In *Conference on Computer Vision and Pattern Recognition*.

[14] ChenEmail, S., Bortsova, G., JuÃąrez, A., Tulder, G., and Bruijne, M. (2019). Multi-Task Attention-Based Semi-Supervised Learning for Medical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[15] Cheng, D., Liao, R., Fidler, S., and Urtasun, R. (2019). DARNet: Deep Active Ray Network for Building Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[16] Cheng, G., Zhou, P., and Han, J. (2016). RIFD-CNN: Rotation-Invariant and Fisher Discriminative Convolutional Neural Networks for Object Detection. In *Conference on Computer Vision and Pattern Recognition*.

[17] Chibane, J., Alldieck, T., and Pons-Moll, G. (2020). Implicit Functions in Feature Space for 3D Shape Reconstruction and Completion. In *Conference on Computer Vision and Pattern Recognition*.

[18] Choy, C., Xu, D., Gwak, J., Chen, K., and Savarese, S. (2016). 3DR2N2: A unified approach for single and multi-view 3d object reconstruction. In *European Conference on Computer Vision*.

[19] Çiçek, Ö., Abdulkadir, A., Lienkamp, S., Brox, T., and Ronneberger, O. (2016). 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 424–432.

[20] Cohen, L. and Cohen, I. (1993). Finite-Element Methods for Active Contour Models and Balloons for 2D and 3D Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147.

[21] Cremers, D., Osher, S., and Soatto, S. (2006). Kernel Density Estimation and Intrinsic Alignment for Shape Priors in Level-Set Segmentation. *International Journal of Computer Vision*.

[22] Dalca, A., Yu, E., Golland, P., Fischl, B., Sabuncu, M., and Iglesias, J. (2019). Unsupervised Deep Learning for Bayesian Brain MRI Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[23] Dong, S., Luo, G., Wang, K., Cao, S., Mercado, A., Shmuilovich, O., Zhang, H., and Li, S. (2018). VoxelAtlasGan: 3D Left Ventricle Segmentation on Echocardiography with Atlas Guided Generation and Voxel-To-Voxel Discrimination. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 622–629.

[24] Dong, S. and Zhang, H. (2018). A Combined Fully Convolutional Networks and Deformable Model for Automatic Left Ventricle Segmentation Based on 3D Echocardiography. In *BioMed Research International*.

[25] Dorent, R., Joutard, S., Shapey, J., Bisdas, S., Kitchen, N., Bradford, R., Saeed, S., Modat, M., Ourselin, S., and Vercauteren, T. (2020). Scribble-based Domain Adaptation via Co-segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[26] Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. In *Journal of Machine Learning Research*.

[27] Duque, V., Chanti, D., Crouzier, M., Nordez, A., Lacourpaille, L., and Mateus, D. (2020). Spatio-temporal Consistency and Negative Label Transfer for 3D freehand US Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[28] Durand, T., Mordan, T., Thome, N., and Cord, M. (2017). WILDCAT: Weakly Supervised Learning of Deep Convnets for Image Classification, Pointwise Localization and Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[29] Fei, W., Mengqing, J., Chen, Q., Shuo, Y., Cheng, L., Honggang, Z., and Xiaogang, W. (2017). Residual attention network for image classification. In *Conference on Computer Vision and Pattern Recognition*.

[30] Feng, X., Yang, J., Laine, A., and Angelini, E. (2017). Discriminative Localization in CNNs for Weakly-Supervised Segmentation of Pulmonary Nodules. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 568–576.

[31] Feyjie, A., Azad, R., Pedersoli, M., Kauffman, C., Ayed, I., and Dolz, J. (2020). Semi-supervised few-shot learning for medical image segmentation. In *arXiv Preprint*.

[32] Freedman, D. and Zhang, T. (2005). Interactive Graph-Cut Based Segmentation with Shape Priors. In *Conference on Computer Vision and Pattern Recognition*, pages 755–62.

[33] Fua, P. (1996). Model-Based Optimization: Accurate and Consistent Site Modeling. In *International Society for Photogrammetry and Remote Sensing*.

[34] Fua, P. and Leclerc, Y. G. (1995). Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading. *International Journal of Computer Vision*, 16:35–56.

[35] Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059.

[36] Ge, W., Lin, X., and Yu, Y. (2019). Weakly Supervised Complementary Parts Models for Fine-Grained Image Classification from the Bottom Up. In *Conference on Computer Vision and Pattern Recognition*.

[37] Ge, W., Yanga, S., and Yu, Y. (2018). Multi-Evidence Filtering and Fusion for Multi-Label Classification, Object Detection and Semantic Segmentation Based on Weakly Supervised Learning. In *Conference on Computer Vision and Pattern Recognition*.

[38] Gkioxari, G., Malik, J., and Johnson, J. (2019). Mesh R-CNN. In *International Conference on Computer Vision*.

[39] Guennebaud, G. and Gross, M. (2007). Algebraic point set surfaces. In *ACM SIGGRAPH*.

[40] Gur, S., Wolf, L., Golgher, L., and Blinder, P. (2019). Unsupervised Microvascular Image Segmentation Using an Active Contours Mimicking Neural Network. In *International Conference on Computer Vision*.

[41] Harley, A., Derpanis, K., and Kokkinos, I. (2017). Segmentation-Aware Convolutional Networks Using Local Attention Masks. In *International Conference on Computer Vision*.

[42] Hatamizadeh, A., Sengupta, D., and Terzopoulos, D. (2020). End-To-End Trainable Deep Active Contour Models for Automated Image Segmentation: Delineating Buildings in Aerial Imagery. In *arXiv Preprint*.

[43] He, L., Peng, Z., E., B., Wang, X., Han, C. Y., Weiss, K. L., and Wee, W. G. (2008). A Comparative Study of Deformable Contour Methods on Medical Image Segmentation. *Image and Vision Computing*, 26(2):141–163.

[44] Hering, A., Kuckertz, S., Heldmann, S., and Heinrich, M. P. (2019). Enhancing Label-Driven Deep Deformable Image Registration with Local Distance Metrics for State-Of-The-Art Cardiac Motion Tracking. In *Bildverarbeitung für die Medizin 2019*.

[45] Hsu, C., Hsu, K., Tsai, C., Lin, Y., and Chuang, Y. (2019). Weakly Supervised Instance Segmentation Using the Bounding Box Tightness Prior. In *Advances in Neural Information Processing Systems*.

[46] Huang, Z., Wang, X., Wang, J., Liu, W., and Wang, J. (2018). Weakly-Supervised Semantic Segmentation Network with Deep Seeded Region Growing. In *Conference on Computer Vision and Pattern Recognition*.

[47] Huo, Y., Xu, Z., Aboud, K., Parvathaneni, P., Bao, S., Bermudez, C., Resnick, S., Cutting, L., and Landman, B. (2018). Spatially Localized Atlas Network Tiles Enables 3D Whole Brain Segmentation from Limited Data. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[48] Iglesias, J. E. and Sabuncu, M. R. (2015). Multi-Atlas Segmentation of Biomedical Images: A Survey. *Medical Image Analysis*, 24(1):205–219.

[49] Iglovikov, V. and Shvets, A. (2018). Ternausnet: U-Net with VGG11 Encoder Pre-Trained on Imagenet for Image Segmentation. In *arXiv Preprint*.

[50] IsenseeEmail, F., Petersen, J., Zimmerer, A. K., Jaeger, P., Kohl, S., and Wasserthal, J. (2018). Nnu-Net: Self-Adapting Framework for U-Net-Based Medical Image Segmentation. In *arXiv Preprint*.

[51] Januszewski, M., Kornfeld, J., Li, P., Pope, A., Blakely, T., Lindsey, L., Maitin-Shepard, J., Tyka, M., Denk, W., and Jain, V. (2018). High-Precision Automated Reconstruction of Neurons with Flood-Filling Networks. *Nature Methods*.

[52] Jorstad, A., Nigro, B., Cali, C., Wawrzyniak, M., Fua, P., and Knott, G. (2014). Neuromorph: A Toolset for the Morphometric Analysis and Visualization of 3D Models Derived from Electron Microscopy Image Stacks. *Neuroinformatics*, 13(1):83–92.

[53] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active Contour Models. *International Journal of Computer Vision*, 1(4):321–331.

[54] Katouzian, A., Wang, H., Conjeti, S., Tang, H., Dehghan, E., Karargyris, A., Pillai, A., Clarkson, K., and Navab, N. (2018). Hashing-Based Atlas Ranking and Selection for Multiple-Atlas Segmentation. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2018*.

[55] Kavur, A. and Selver, M. (2020). CHAOS Challenge - Combined (CT-MR) Healthy Abdominal Organ Segmentation. In *arXiv Preprint*.

[56] Khoreva, A., Benenson, R., Hosang, J., Hein, M., and Schiele, B. (2017). Simple Does It: Weakly Supervised Instance and Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 1665–1674.

[57] Kingma, D. P. and Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

[58] Krähenbühl, P. and Koltun, V. (2011). Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. In *Advances in Neural Information Processing Systems*.

[59] Kulharia, V., Chandra, S., Agrawal, A., Torr, P., and Tyagi, A. (2020). Box2seg: Attention Weighted Loss and Discriminative Feature Learning for Weakly Supervised Segmentation. In *European Conference on Computer Vision*.

[60] Lee, M., Petersen, K., Pawlowski, N., Glocker, B., and Schaap, M. (2019). TETRIS: Template Transformer Networks for Image Segmentation With Shape Priors. In *IEEE Transactions on Medical Imaging*.

[61] Lengagne, R., Fua, P., and Monga, O. (1997). Using Differential Constraints to Reconstruct Complex Surfaces from Stereo. In *Conference on Computer Vision and Pattern Recognition*.

[62] Lengagne, R., Fua, P., and Monga, O. (2000). 3D Stereo Reconstruction of Human Faces Driven by Differential Constraints. *Image and Vision Computing*, 18(4):337–343.

[63] Leventon, M. E., Grimson, W. E., and Faugeras, O. (2000). Statistical Shape Influence in Geodesic Active Contours. In *Conference on Computer Vision and Pattern Recognition*, pages 316–323.

[64] Liang, J., Homayounfar, N., Ma, W., Xiong, Y., Hu, R., and Urtasun, R. (2020). Polytransform: Deep Polygon Transformer for Instance Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[65] Ling, H., Gao, J., Kar, A., Chen, W., and Fidler, S. (2019). Fast Interactive Object Annotation with Curve-Gcn. In *Conference on Computer Vision and Pattern Recognition*, pages 5257–5266.

[66] Liu, X., Thermos, S., OâĂŹNeil, A., and Tsaftaris, S. (2021). Semi-supervised Meta-learning with Disentanglement for Domain-generalised Medical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[67] Lorensen, W. and Cline, H. (1987). Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *ACM SIGGRAPH*, pages 163–169.

[68] Maninis, K., Caelles, S., Pont-Tuset, J., and Gool, L. (2018). Deep Extreme Cut: from Extreme Points to Object Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[69] Marcos, D., Tuia, D., Kellenbergerg, B., and Urtasun, R. (2018). Learning Deep Structured Active Contours End-To-End. In *Conference on Computer Vision and Pattern Recognition*.

[70] Mcinerney, T. and Terzopoulos, D. (1995). A Dynamic Finite Element Surface Model for Segmentation and Tracking in Multidimensional Medical Images with Application to Cardiac 4D Image Analysis. *Computerized Medical Imaging and Graphics*, 19(1):69–83.

[71] Mcinerney, T. and Terzopoulos, D. (1996). Deformable Models in Medical Image Analysis: A Survey. *Medical Image Analysis*, 1:91–108.

[72] Mescheder, L., Oechsle, M., Niemeyer, M., Nowozin, S., and Geiger, A. (2019). Occupancy Networks: Learning 3D Reconstruction in Function Space. In *Conference on Computer Vision and Pattern Recognition*, pages 4460–4470.

[73] Milletari, F., Navab, N., and Ahmadi, S.-A. (2016). V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. In *arXiv Preprint*.

[74] Mirikharaji, Z. and Hamarneh, G. (2018). Star Shape Prior in Fully Convolutional Networks for Skin Lesion Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 737–745.

[75] Mortensen, E. and Barrett, W. (1995). Intelligent Scissors for Image Composition. In *ACM SIGGRAPH*, pages 191–198.

[76] Newman, T. and Yi, H. (2006). A Survey of the Marching Cubes Algorithm. *Computers & Graphics*, 30(5):854–879.

[77] Orlando, J. and BogunoviÄĞ, H. (2019). REFUGE Challenge: A Unified Framework for Evaluating Automated Methods for Glaucoma Assessment from Fundus Photograph. In *arXiv Preprint*.

[78] Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2002). Shape Distributions. In *ACM Transactions on Graphics*.

[79] Pan, J. and Jia, K. (2019). Deep Mesh Reconstruction from Single RGB Images via Topology Modification Networks. In *International Conference on Computer Vision*.

[80] Park, J. J., Florence, P., Straub, J., Newcombe, R. A., and Lovegrove, S. (2019). DeepSdf: Learning Continuous Signed Distance Functions for Shape Representation. In *Conference on Computer Vision and Pattern Recognition*.

[81] Peng, S., Jiang, W., Pi, H., Li, X., Bao, H., and Zhou, X. (2020). Deep Snake for Real-Time Instance Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[82] Prevost, R., Cuingnet, R., Mory, B., L.D. C., D., and Ardon, R. (2013). Incorporating Shape Variability in Image Segmentation via Implicit Template Deformation. *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 82–89.

[83] Qu, H., Wu, P., Uuang, Q., Yi, J., Yan, Z., Riedlinger, G., De, S., Zhang, S., and Metaxas, D. (2020). Weakly Supervised Deep Nuclei Segmentation Using Partial Points Annotation in Histopathology Images. In *IEEE Transactions on Medical Imaging*.

[84] Remelli, E., Lukoianov, A., Richter, S., Guillard, B., Bagautdinov, T., Baque, P., and Fua, P. (2020). MeshSdf: Differentiable Iso-Surface Extraction. In *Advances in Neural Information Processing Systems*.

[85] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Conference on Medical Image Computing and Computer Assisted Intervention*, pages 234–241.

[86] Roth, H., Zhang, L., Yang, D., Milletari, F., Xu, Z., Wang, X., and Xu, D. (2019). Weakly supervised segmentation from extreme points. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[87] Scholkopf, B. and Smola, A. (2002). *Learning with Kernels*. MIT Press.

[88] Shvets, A., Rakhlin, A., Kalinin, A., and Iglovikov, V. (2018). Automatic Instrument Segmentation in Robot-Assisted Surgery Using Deep Learning. In *arXiv Preprint*.

[89] Simpson, A. L., Antonelli, M., Bakas, S., Bilello, M., Farahani, K., Van Ginneken, B., Kopp-Schneider, A., Landman, B. A., Litjens, G., Menze, B., et al. (2019). A Large Annotated Medical Image Dataset for the Development and Evaluation of Segmentation Algorithms. *arXiv Preprint*.

[90] Sitzmann, V., Martel, J., Bergman, A., Lindell, D., and Wetzstein, G. (2020). Implicit Neural Representations with Periodic Activation Functions. In *Advances in Neural Information Processing Systems*.

[91] Song, C., Huang, Y., Ouyang, W., and Wang, L. (2019). Box-Driven Class-Wise Region Masking and Filling Rate Guided Loss for Weakly Supervised Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[92] Spitzer, H., Amunts, K., Harmeling, S., and Dickscheid, T. (2017). Parcellation of Visual Cortex on High-Resolution Histological Brain Sections Using Convolutional Neural Networks. In *International Symposium on Biomedical Imaging*, pages 920–923.

[93] Spitzer, H., Kiwitz, K., Amunts, K., Harmeling, S., and Dickscheid, T. (2018). Improving Cytoarchitectonic Segmentation of Human Brain Areas with Self-Supervised Siamese Networks. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[94] Tang, H., Liu, X., Sun, S., Yan, X., and Xie, X. (2021). Recurrent Mask Refinement for Few-Shot Medical Image Segmentation. In *International Conference on Computer Vision*.

[95] Terzopoulos, D., Witkin, A., and Kass, M. (1987). Symmetry-Seeking Models and 3D Object Reconstruction. *International Journal of Computer Vision*, 1:211–221.

[96] Terzopoulos, D., Witkin, A., and Kass, M. (1988). Constraints on Deformable Models: Recovering 3D Shape and Nonrigid Motion. *Artificial Intelligence*, 36(1):91–123.

[97] Vasu, S., Talabot, N., Lukoianov, A., Baque, P., Donier, J., and Fua, P. (2021). HybridSDF: Combining Free Form Shapes and Geometric Primitives for effective Shape Manipulation. In *arXiv Preprint*.

[98] Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., and Jiang, Y. (2018). Pixel2mesh: Generating 3D Mesh Models from Single RGB Images. In *European Conference on Computer Vision*.

[99] Wang, Z., Acuna, D., Ling, H., Kar, A., and Fidler, S. (2020). Object Instance Annotation with Deep Extreme Level Set Evolution. In *European Conference on Computer Vision*.

[100] Wen, C., Zhang, Y., Li, Z., and Fu, Y. (2019). Pixel2mesh++: Multi-View 3D Mesh Generation via Deformation. In *International Conference on Computer Vision*.

[101] Wickramasinghe, U., Knott, G., and Fua, P. (2019). Probabilistic atlases to enforce topological constraints. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[102] Wickramasinghe, U., Knott, G., and Fua, P. (2021). Deep active surface models. In *Conference on Computer Vision and Pattern Recognition*.

[103] Wickramasinghe, U., Remelli, E., Knott, G., and Fua, P. (2020). Voxel2mesh: 3d mesh model generation from volumetric data. In *Conference on Medical Image Computing and Computer Assisted Intervention*.

[104] Wolf, I., Vetter, M., Wegner, I., Nolden, M., Bottger, T., Hastenteufel, M., Schobinger, M., Kunert, T., and Meinzer, H. (2004). The medical imaging interaction toolkit (MITK): a toolkit facilitating the creation of interactive software by extending VTK and ITK. In *IEEE Transactions on Medical Imaging*.

[105] Xia, X. and Kulis, B. (2017). W-Net: A Deep Model for Fully Unsupervised Image Segmentation. In *arXiv Preprint*.

[106] Xu, Q., Wang, W., Ceylan, D., Mech, R., and Neumann, U. (2019). DISN: Deep Implicit Surface Network for High-Quality Single-View 3D Reconstruction. In *Advances in Neural Information Processing Systems*.

[107] Xue1, Y., Xu1, T., Zhang, H., Long, L., and Huang1, X. (2018). SegAN: Adversarial Network with Multi-scale L1 Loss for Medical Image Segmentation. In *Neuroinformatics*.

[108] Yang, L., Wang, Y., Xiong, X., Yang, J., and Katsaggelos, A. (2018). Efficient Video Object Segmentation via Network Modulation. In *Conference on Computer Vision and Pattern Recognition*.

[109] Zhao, A. and Durand, F. (2019). Data Augmentation Using Learned Transformations for One-Shot Medical Image Segmentation. In *Conference on Computer Vision and Pattern Recognition*.

[110] Zhao, T. and Yin, Z. (2020). Weakly Supervised Cell Segmentation by Point Annotation. In *IEEE Transactions on Medical Imaging*.

[111] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. (2015). Conditional Random Fields as Recurrent Neural Networks. In *International Conference on Computer Vision*.

[112] Zotti, C., Luo, Z., Lalande, A., and Jodoin, P. (2019). Convolutional Neural Network With Shape Prior Applied to Cardiac MRI Segmentation. In *IEEE Transactions on Medical Imaging*.

Udaranga Wickramasinghe

Doctoral Student,
School of Computer and Communication Sciences,
École Polytechnique Fédérale de Lausanne,
https://udaranga.github.io

Permanent Address
Chemin des Triaudes 4,
1024 Ecublens, Switzerland.
E-mail: udaranga@gmail.com

Fields of Interest

Computer Vision, Machine Learning.

Education

PhD in Computer Science — Sept. 2017 - Jan. 2022
École Polytechnique Fédérale de Lausanne — Lausanne, Switzerland
Supervisor: Prof. Pascal Fua

Master of Science — Sept. 2014 - April 2017
Major: Computer Science — Lausanne, Switzerland
École Polytechnique Fédérale de Lausanne

Bachelor of the Science of Engineering — July 2009 - March 2014
Major: Electronics and Telecommunication Engineering — Moratuwa, Sri Lanka
University of Moratuwa

Publications

U. Wickramasinghe, P. Jensen, J.Yang, P. Fua, "Weakly Supervised Volumetric Image Segmentation with Deformed Templates," in *ArXiv*, 2021 [link].

U. Wickramasinghe, G. Knott, P. Fua, "Deep Active Surface Models," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021 [link].

U. Wickramasinghe, E. Remelli, G. Knott, P. Fua, "Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2020 [link].

U. Wickramasinghe, G. Knott, P. Fua, "Probabilistic Atlases to Enforce Topological Constraints," in *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2019 [link].

Work Experience

Facebook — Sept. 2021 - Dec. 2021
Research Intern — Pittsburgh - USA (remote)

Innvoview — Sept. 2016 - June 2017
Intern — Lausanne, Switzerland

ViDi Systems (Cognex) - Switzerland — March 2016 - Aug. 2016
Intern — Fribourg - Switzerland

LSP - EPFL — Sept. 2014 - Feb. 2016
Research assistant — Lausanne, Switzerland

Teaching Experience

Computer Vision — IC - EPFL

Linear Algebra — EPFL

Technical Skills

Programming languages : Proficient - Python (PyTorch), Matlab
: Familiar with - C/C++ (CUDA), Java, Android, C#