Voxel2Mesh: 3D Mesh Model Generation from Volumetric Data

Udaranga Wickramasinghe[1(✉)], Edoardo Remelli[1], Graham Knott[2], and Pascal Fua[1]

[1] Computer Vision Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland
udaranga.wickramasinghe@epfl.ch
[2] BioEM Laboratory, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

Abstract. CNN-based volumetric methods that label individual voxels now dominate the field of biomedical segmentation. However, 3D surface representations are often required for proper analysis. They can be obtained by post-processing the labeled volumes which typically introduces artifacts and prevents end-to-end training. In this paper, we therefore introduce a novel architecture that goes directly from 3D image volumes to 3D surfaces without post-processing and with better accuracy than current methods. We evaluate it on Electron Microscopy and MRI brain images as well as CT liver scans. We will show that it outperforms state-of-the-art segmentation methods.

Keywords: Volumetric segmentation · 3D surfaces · Deep learning

1 Introduction

State-of-the-Art volumetric segmentation techniques rely on Convolutional Neural Networks (CNNs) operating on an image volume [3, 13, 17]. However in clinical and research practice, a mesh representation is often required to model the surface morphology and to compute area-based statistics. Unfortunately, converting volumes into surfaces relies on algorithms such as Marching Cubes [10] followed by mesh smoothing, which is not differentiable, prevents end-to-end training, and introduces artifacts.

We therefore introduce an end-to-end trainable architecture that goes directly from volumetric images to 3D surface meshes. Our **Voxel2Mesh** architecture is depicted by Fig. 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.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-59719-1_30) contains supplementary material, which is available to authorized users.

A. L. Martel et al. (Eds.): MICCAI 2020, LNCS 12264, pp. 299–308, 2020.
https://doi.org/10.1007/978-3-030-59719-1_30

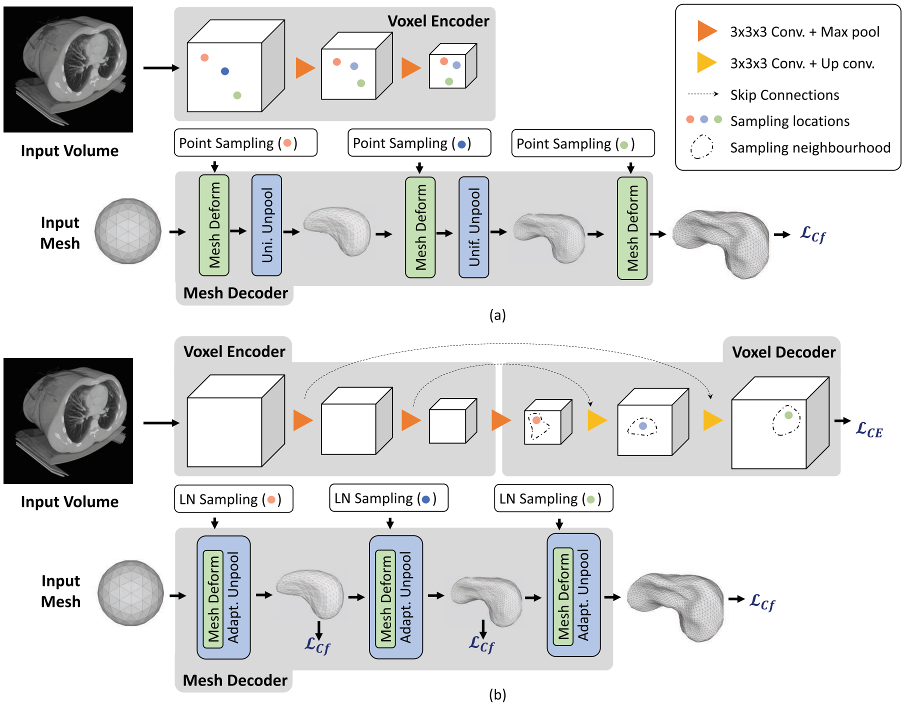

Fig. 1. Architectures (a) The **Pixel2Mesh-3D** architecture, a straightforward extension of [19], uses a surface decoder but no voxel decoder. (b) By contrast, our **Voxel2Mesh** architecture takes as input an image and spherical mesh. They are jointly encoded and then decoded into cubes and meshes 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.

- **Learned Neighborhood Sampling.** The mesh decoder learns to sample the output of the volume decoder only where needed, that is, in the neighborhood 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 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.

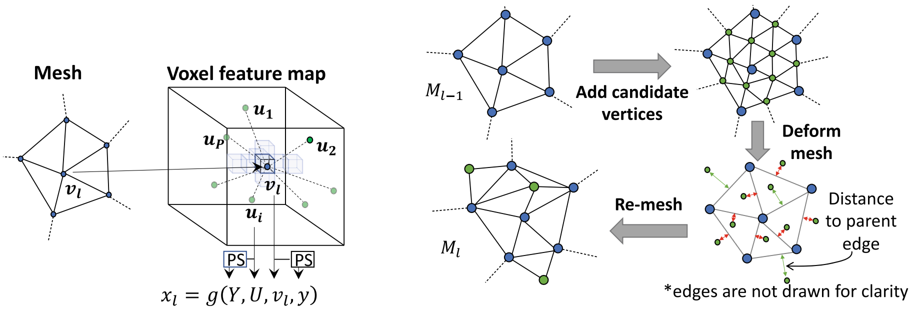

Fig. 2. Approach. (a) Learned Neighborhood Sampling. (b) Adaptive Mesh Unpooling.

2 Related Work

CNN-based volumetric methods such as U-Net and its variants [3,6,13,17] now dominate biomedical image segmentation. This is evident from the CHAOS challenge [2] and Medical Segmentation Decathlon [7] results. The winners of both competitions used ensembles of methods relying on volumetric CNNs.

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 algorithms such as Marching Cubes [10] 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 Deformable Models

Deformable surface models became popular in the 1990s to model biological structures in volumetric data [5,12] and are still being developped [8,9,16]. They are now used in conjunction with deep networks [11] 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.

2.3 Deep Surfaces

In this context, the Pixel2Mesh [19] approach and its more recent variants [15,20] 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. 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.

3 Method

Our **Voxel2Mesh** architecture is depicted by Fig. 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 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. 1(b) are based on a standard U-Net [3] 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.

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

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

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

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

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. 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 [19] or in pre-determined neighborhoods around the vertex [20]. This restricts the sampler's ability to pool information from its neighborhood.

Instead, we introduce our LNS strategy that learns optimum sampling locations. 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

$$\mathbf{U} = \{u_i\}_{i=1}^{P} = f(\mathbf{y}, \mathbf{v}_l). \tag{3}$$

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

$$\mathbf{x}_l = g(\mathbf{Y}, \mathbf{U}, \mathbf{y}, \mathbf{v}_l). \tag{4}$$

As shown in Fig. 2 (a), LNS only samples from the voxel feature map corresponding 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 properly 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 (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 (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 $\mathcal{E}'$ between the points on the sphere. $\mathcal{E}'$ 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 differentiability.

3.2 Loss Function

We use the cross entropy loss $\mathcal{L}_{ce}$ with ground-truth volumes and the Chamfer distance $\mathcal{L}_{cf}$ 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 $\mathcal{L}_{cf}$, we randomly sample points from the 3D mesh [14]. We also introduce three regularization terms normal loss $\mathcal{L}_n$, laplacian loss $\mathcal{L}_{lap}$, and edge length loss $\mathcal{L}_{el}$ to improve convergence and smooth the output mesh [19]. We write the complete loss as

$$\mathcal{L} = \sum_{l=1}^{L} \mathcal{L}_{cf}^{l} + \lambda_1 \mathcal{L}_{ce} + \lambda_2 \mathcal{L}_n + \lambda_3 \mathcal{L}_{lap} + \lambda_4 \mathcal{L}_{el}, \tag{5}$$

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

4 Experiments

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. It 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. They are shown in blue, green, and red respectively in the first two rows of Fig. 4.

Hippocampus Dataset. It consists of 260 labeled MRI image cubes from the Medical Segmentation Decathlon [18]. The task is to segment the hippocampus, as depicted by the fourth row of Fig. 4.

Liver Dataset. It consists of 20 labeled CT image cubes from the CHAOS challenge [2]. The task is to segment the liver as shown in the third row of Fig. 4.

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 approaches, we use **TernausNet**, **LinkNet34**, **ResNet50** and **ResNet50-SE**,

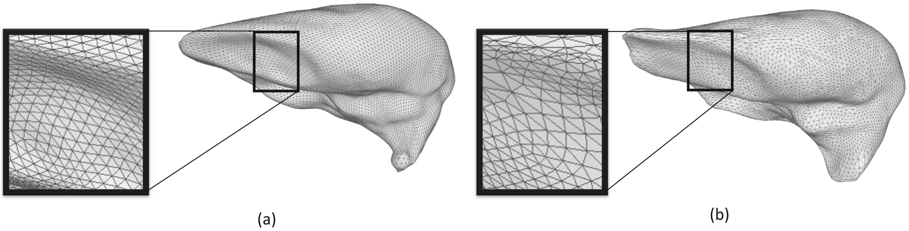

Fig. 3. Levels of resolution. (a) **Pixel2Mesh-3D** result (10422 vertices). (b) **Voxel2Mesh** result (7498 vertices). With our adaptative unpooling, we obtain better results with fewer vertices.

Table 1. Comparative results on three datasets using the IoU metric.

	Liver	Hippo.	Synaptic junction		
			Pre-synap	Synapse	Post-synap
TernausNet [6]	*84.4 ± 1.3*	78.4 ± 1.2	73.5 ± 1.3	64.4 ± 0.5	*78.4 ± 1.3*
LinkNet34 [17]	82.8 ± 1.4	79.4 ± 0.8	72.3 ± 0.5	63.2 ± 1.2	78.2 ± 1.1
ResNet50 [1]	82.1 ± 0.7	80.7 ± 0.2	70.3 ± 0.8	63.3 ± 0.6	76.2 ± 1.4
ResNet50-SE [1]	82.6 ± 1.2	80.5 ± 1.3	71.3 ± 0.6	63.6 ± 0.7	76.3 ± 0.9
V-NET [13]	81.5 ± 1.4	75.3 ± 1.4	64.3 ± 0.7	65.2 ± 1.3	74.1 ± 0.7
U-NET [3]	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**

Table 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**	$\mathbf{1.1 \times 10^{-3}}$

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.

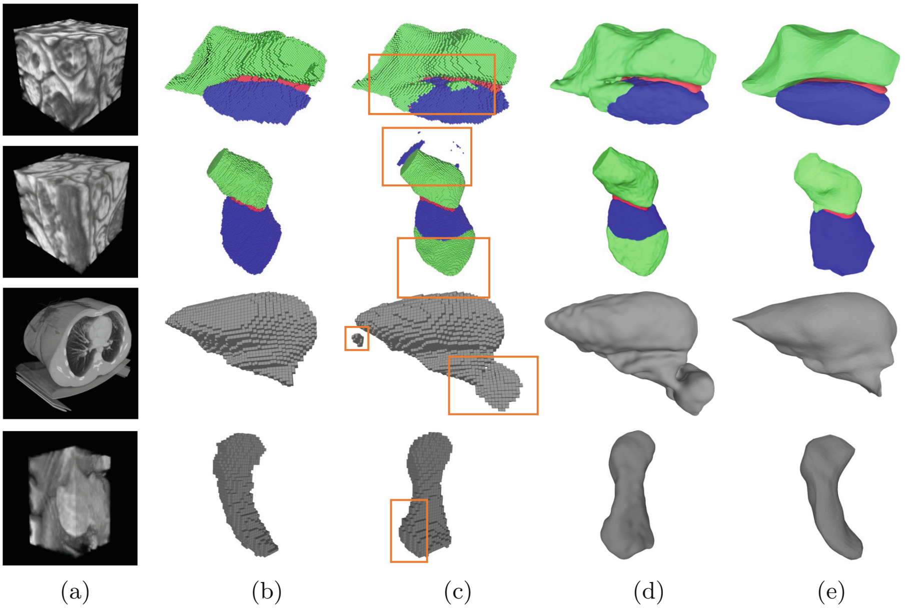

Fig. 4. Qualitative results. (a) Input volumes. EM (row 1,2), CT (row 3), MRI (row 4) (b) Ground truth (c) CNN baseline (d) CNN baseline + post processing (e) **Voxel2Mesh**. The orange boxes highlight false positive regions. (Color figure online)

4.3 Comparative Results

Figure 4 and Fig. 3 depict our results qualitatively and we report quantitative ones in Table 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 [4]. 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.

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 equivalent ones in **Pixel2Mesh-3D**. We also evaluate the performance of Hypothesis Sampling (**HS**), the sampling strategy of [20] that relies on a fixed neighborhood sampling. We report the results in Table 2 in Chamfer distance terms. They demonstrate that our adaptative strategies **LNS+AMU** deliver a clear benefit.

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.

Acknowledgments. This work was supported in part by a Swiss National Science Foundation grant.

References

1. Kavur, A., et al.: CHAOS Challenge - Combined (CT-MR) Healthy Abdominal Organ Segmentation. arXiv Preprint (2020)
2. CHAOS: Combined (CT-MR) Healthy Abdominal Organ Segmentation. In: International Symposium on Biomedical Imaging (2019)
3. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3D U-Net: learning dense volumetric segmentation from sparse annotation. In: Ourselin, S., Joskowicz, L., Sabuncu, M.R., Unal, G., Wells, W. (eds.) MICCAI 2016. LNCS, vol. 9901, pp. 424–432. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46723-8_49
4. Guennebaud, G., Gross, M.: Algebraic point set surfaces. In: ACM SIGGRAPH (2007)
5. He, L., et al.: A comparative study of deformable contour methods on medical image segmentation. Image Vis. Comput. **26**(2), 141–163 (2008)

6. Iglovikov, V., Shvets, A.: Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. arXiv Preprint (2018)
7. IsenseeEmail, F., Petersen, J., Zimmerer, A.K., Jaeger, P., Kohl, S., Wasserthal, J.: nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation. arXiv Preprint (2018)
8. Jorstad, A., Nigro, B., Cali, C., Wawrzyniak, M., Fua, P., Knott, G.: Neuromorph: a toolset for the morphometric analysis and visualization of 3D models derived from electron microscopy image stacks. Neuroinformatics **13**(1), 83–92 (2014)
9. Leventon, M.E., Grimson, W.E., Faugeras, O.: Statistical shape influence in geodesic active contours. In: Conference on Computer Vision and Pattern Recognition, pp. 316–323 (2000)
10. Lorensen, W., Cline, H.: Marching cubes: a high resolution 3D surface construction algorithm. In: ACM SIGGRAPH (1987)
11. Marcos, D., Tuia, D., Kellenbergerg, B., Urtasun, R.: Learning deep structured active contours end-to-end. In: Conference on Computer Vision and Pattern Recognition (2018)
12. Mcinerney, T., Terzopoulos, D.: Deformable models in medical image analysis: a survey. Med. Image Anal. **1**, 91–108 (1996)
13. Milletari, F., Navab, N., Ahmadi, S.A.: V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation. arXiv Preprint, June 2016
14. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Shape distributions. ACM Trans. Graph. **21**(4), 807–832 (2002)
15. Pan, J., Jia, K.: Deep mesh reconstruction from single rgb images via topology modification networks. In: International Conference on Computer Vision (2019)
16. Prevost, R., Cuingnet, R., Mory, B., Cohen, L.D., Ardon, R.: Incorporating shape variability in image segmentation via implicit template deformation. In: Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (eds.) MICCAI 2013. LNCS, vol. 8151, pp. 82–89. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40760-4_11
17. Shvets, A., Rakhlin, A., Kalinin, A., Iglovikov, V.: Automatic Instrument Segmentation in Robot-Assisted Surgery Using Deep Learning. arXiv Preprint (2018)
18. Simpson, A., Cardoso, M.: A large annotated medical image dataset for the development and evaluation of segmentation algorithms. arXiv Preprint (2019)
19. Wang, N., Zhang, Y., Li, Z., Fu, Y., Liu, W., Jiang, Y.: Pixel2mesh: generating 3D mesh models from single RGB images. In: European Conference on Computer Vision (2018)
20. Wen, C., Zhang, Y., Li, Z., Fu, Y.: Pixel2mesh++: multi-view 3D mesh generation via deformation. In: International Conference on Computer Vision (2019)