IntrA: 3D Intracranial Aneurysm Dataset for Deep Learning Supplementary Materials

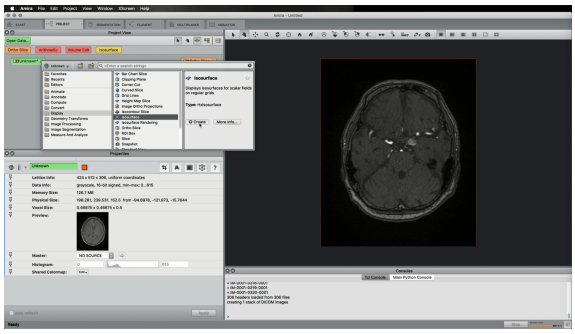
Xi Yang[1] Ding Xia[1,2] Taichi Kin[1] Takeo Igarashi[1]
[1]The University of Tokyo [2]South China University of Technology

1. Data Processing Details

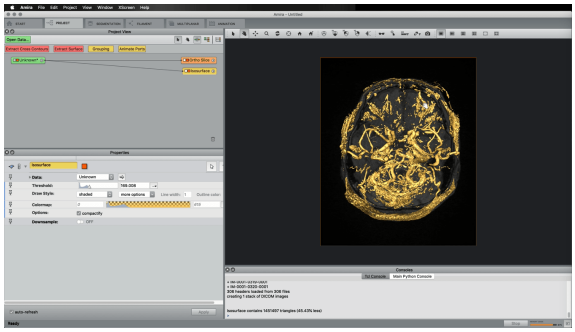
1.1. 3D surface reconstruction

We show a step-by-step instruction with screenshots using Amira as follows.

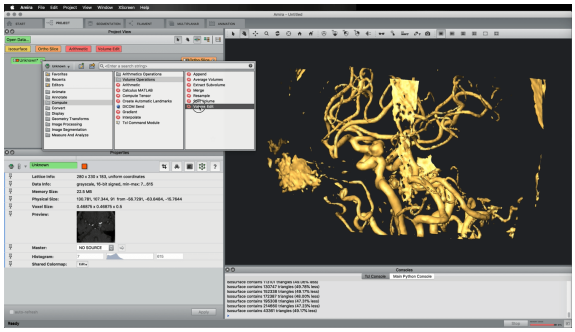
1. Import a set of MRA images, and use the function "Isosurface" to create a surface model from MRA images.



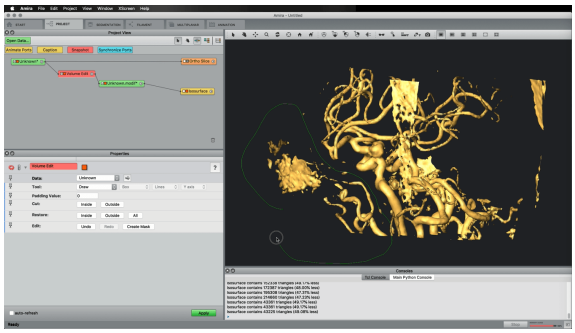
2. Adjust the threshold of isosurface to generate the details you want of the surface model.



3. Use the function "Volume Edit" to refine the generated model.



4. Choose "Cut: Inside" to remove the parts.



1.2. Aneurysm restoration

Since the pixel values are different in MRA at each location of thick blood vessels, thin blood vessels, and turbulent flow, it is impossible to extract an accurate blood vessel model with the single threshold method [5]. Thus, the multi-threshold method is proposed as follows [6]:

1. Rendering the surface of the aneurysm and coarse area of surrounding blood vessels from MRA with the single threshold method.

2. Rendering the surface with a threshold value different from step 1 for unrendered areas, *e.g.* a part of the aneurysm.

3. The threshold value $X\%$ of SI_{max} was calculated from the following equation 1. We use $X = 50$ in

this paper. Here, SI_{max} is the signal intensity maximum of a blood vessel, and SI_{bg} is the signal intensity of the background level.

$$X\%\ of\ SI_{max} = SI_{bg} + \frac{(SI_{max} - SI_{bg})}{100} \times X \quad (1)$$

4. Repeating step 2 as many times as necessary to complete the aneurysm model.

1.3. Data clean

Uncleaned data is feasible for voxel-based and points-based methods. However, the mesh-based method (MeshCNN) requires manifold surfaces. We use MeshLab v0.3.2 [2] and Blender 2.8.0 [1] to clean the data, the details as follows. MeshLab can check them as the description of MeshLab 2.Render.

- MeshLab
  1. Filters → Clean and Repairing →
     - Remove Duplicate Faces
     - Remove Duplicated Vertex
     - Remove Faces from Non Manifold Edges
     - Remove Isolated pieces (wrt. Diameter) with value 100
  2. Render →
     - Show Non Manif Edges
     - Show Non Manif Vertices
- Blender
  1. Import obj files
  2. Select Edit mode
  3. Select vertex → Rip Vertices
  4. Select edge → Edge Split
  5. Press G → move vertices to optimal positions

2. Dataset Organization

According to the submission requirements (maximum 100MB), we do not include the data into supplementary materials. Our data and codes can be accessed at https://github.com/intra3d2019/IntrA.

2.1. Data

Our dataset is organized as follows. The complete models are recorded as OBJ files. The annotated and generated models have two formats: OBJ and AD. The AD file consists of point position (3), normal (3), and segmentation label (1) of each line. The label 0, 1, 2 denote parent vessel, aneurysm, and boundary line, respectively. In our segmentation experiments, we assigned boundary line points (label 2) into the aneurysm (label 1) to conduct binary segmentation.

- annotated
  - ad
  - obj
  - geo
- generated
  - aneurysm
    - ad
    - obj
  - vessel
    - ad
    - obj
- complete

2.2. Tools

- Annotation
  - annotation/main.py
- Vessel segment generation
  - random_pick.py
  - selection.py
- Visualization
  - show_ann_data.py
  - show_result.py

3. Experimental Details for Benchmarks

We implemented the data loader to fulfill the default input requirements of every method, including normal information and data augmentation. The number of categories is reduced to 2 for classification. We recorded results when the networks reach the best IOU of parent vessels, the best IOU of aneurysms, the best DSC of parent vessels, and the best DSC of aneurysms, respectively. The network models that have the best IOU of aneurysms were selected for comparison in our paper.

SO-Net [8] (Pytorch) The hyper-parameters and settings of SO-Net for segmentation and classification are the same as that for training ShapeNet. The optimizer of the encoder, segmenter, and classifier is Adam. The learning rate is 10^{-3}. We trained each network for up to 401 epochs, with batch size 8.

PointConv [13] (TensorFlow) We used the hyper-parameters originally for ScanNet. For training, we used 501 epochs and batch size 4, with Adam optimizer and learning rate starting from 10^{-3}.

PointNet++ & PointNet [11, 10] (Pytorch) for segmentation. The hyper-parameters were for training ShapeNet. We used batch size 8 and learning rate 10^{-3}, and trained models for 201 epochs with Adam as the optimizer.

Figure 1: Two examples of mis-classified aneurysm segments.

(TensorFlow) for classification. We used 181 epochs for training with batch size 8 and learning rate 10^{-3}. The optimizer of the model is Adam.

PointNet++g (Pytorch) We adapted the PointNet++ model for segmentation by adding geometry information. The geodesic distance was used as the metric to replace the original Euclidean distance, in the ball query function and the normalization of pre-processing for the point cloud.

PointCNN [9] (TensorFlow) We took the segmentation architecture of ShapeNet. The batch size was 4, and the total epoch was 1024. Adam was used as the optimizer, and the learning rate started from 5×10^{-3} to 10^{-5}.The architecture of classification was for ModelNet. In the training of our dataset, the total epoch was 1024, and the learning rate decreased from 5×10^{-3} to 10^{-6}. The optimizer we used was also Adam. However, the batch size was 128.

MeshCNN [4] (Pytorch) We used the same pooling resolutions (1800, 1350, 600) with the setting of segmentation architecture for 2250 edges, then modified that as (1200, 800, 300) for 1500 edges and (600, 450, 180) for 750 edges. The batch size was 12, with 200 epoch, and the learning rate was 10^{-3} for Adam.

SpiderCNN [14] (TensorFlow) Both segmentation and classification codes were for ShapeNet, and the optimizers were Adam. The main difference was that the segmentation model trained for 501 epochs with batch size 8, while the classification part was 151 epochs with batch size 16.

SSCN [3] (Pytorch) These models were designed for ShapeNet. We trained the model for 400 epochs with batch size 16. The optimizer of the model was SGD (momentum

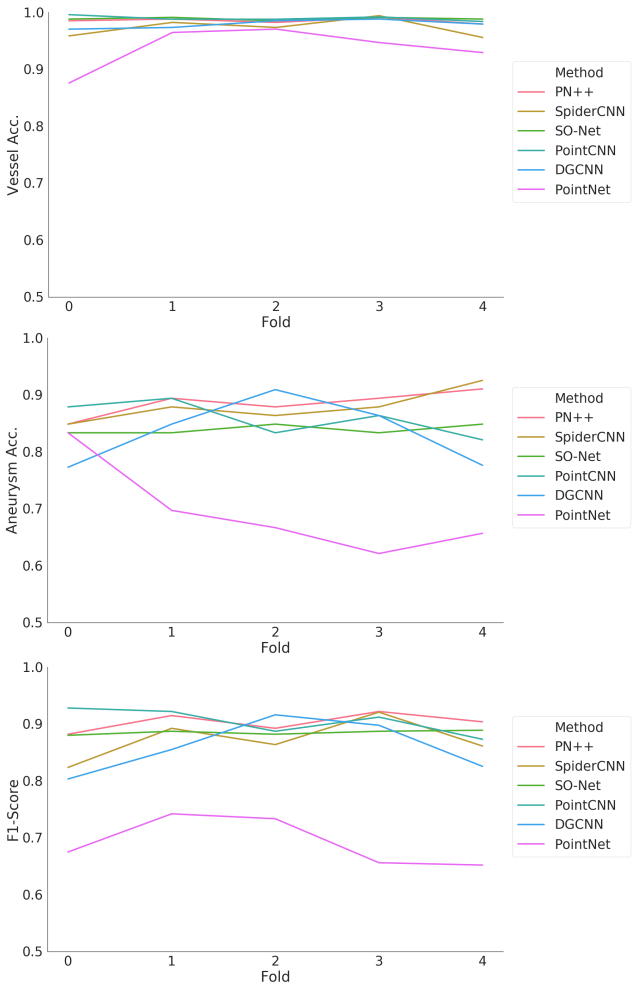


Figure 2: Classification results of networks. These methods are compared with their best performance.

0.9), and the learning rate was 10^{-1}.

PointGrid [7] (TensorFlow) The architecture of PointGrid was for ShapeNet. We trained the model with batch size 32 and 200 epochs. The optimizer was Adam, and the learning rate was 10^{-4}.

DGCNN [12] (Pytorch) The model was trained by batch size 16 and epoch 200. The optimizer used SGD (momentum 0.9) with learning rate 10^{-1}. We removed the 'translation' in the data augmentation because when it was used, we obtained poor performance on our dataset.

4. More Experimental Results

Two misclassified aneurysm segments are shown in Figure 1 and the reason is explained in Section 5.1 of the paper. Figure 2 demonstrates detailed information of the classification results for each fold, including blood vessel accuracy, aneurysm accuracy, and F1-score.

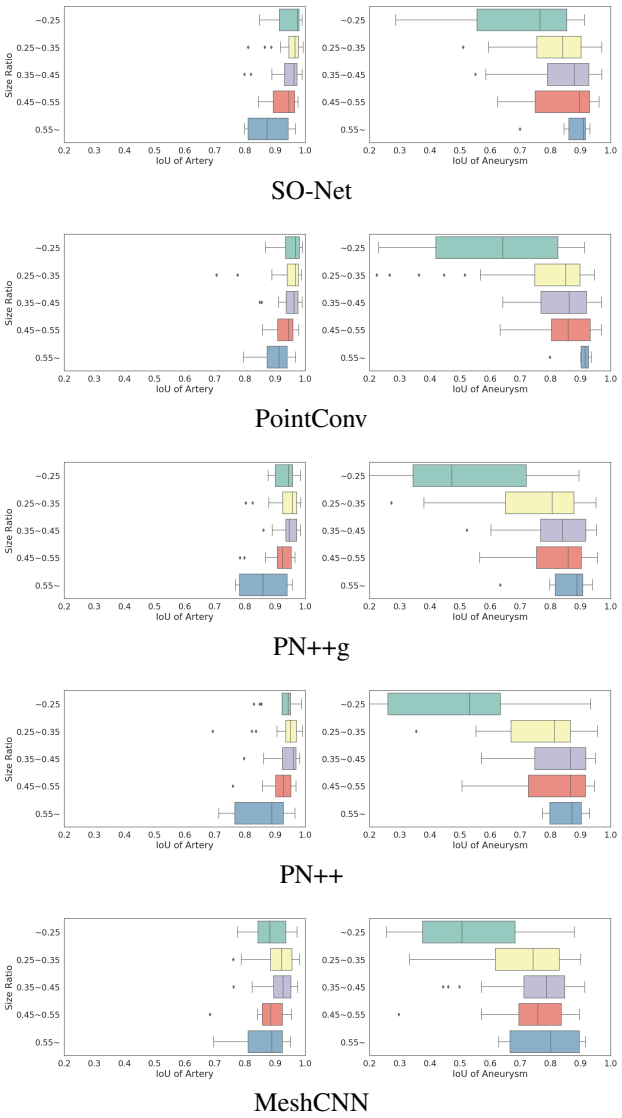


Figure 3: Segmentation results of five networks counted by the size ratio of aneurysms.

10 representative results of five networks are presented in the Figure 4.We transformed the result of MeshCNN into point clouds. The first 2 rows are good results from these methods. The second part contains segments with aneurysms close to arteries, which can cause an unclean segmentation. We noticed that SO-Net (third row) has aneurysm points on the parent vessel part, and PN++ (fourth row) has parent vessel points on the aneurysm part, which is the result of omitting geodesic information. In the third part, we add four poor results caused by small aneurysms. Finally, we provide two uncommon cases in the last part.

Since the diameter of the aneurysm has an important value for diagnosis, we also count the segmentation results of five networks according to the size ratio of aneurysms as shown in Figure 3. Unsatisfactory segmentation on small size ratio aneurysms can be confirmed.

5. Statement

This study was approved by the Research Ethics Committee of The University of Tokyo.

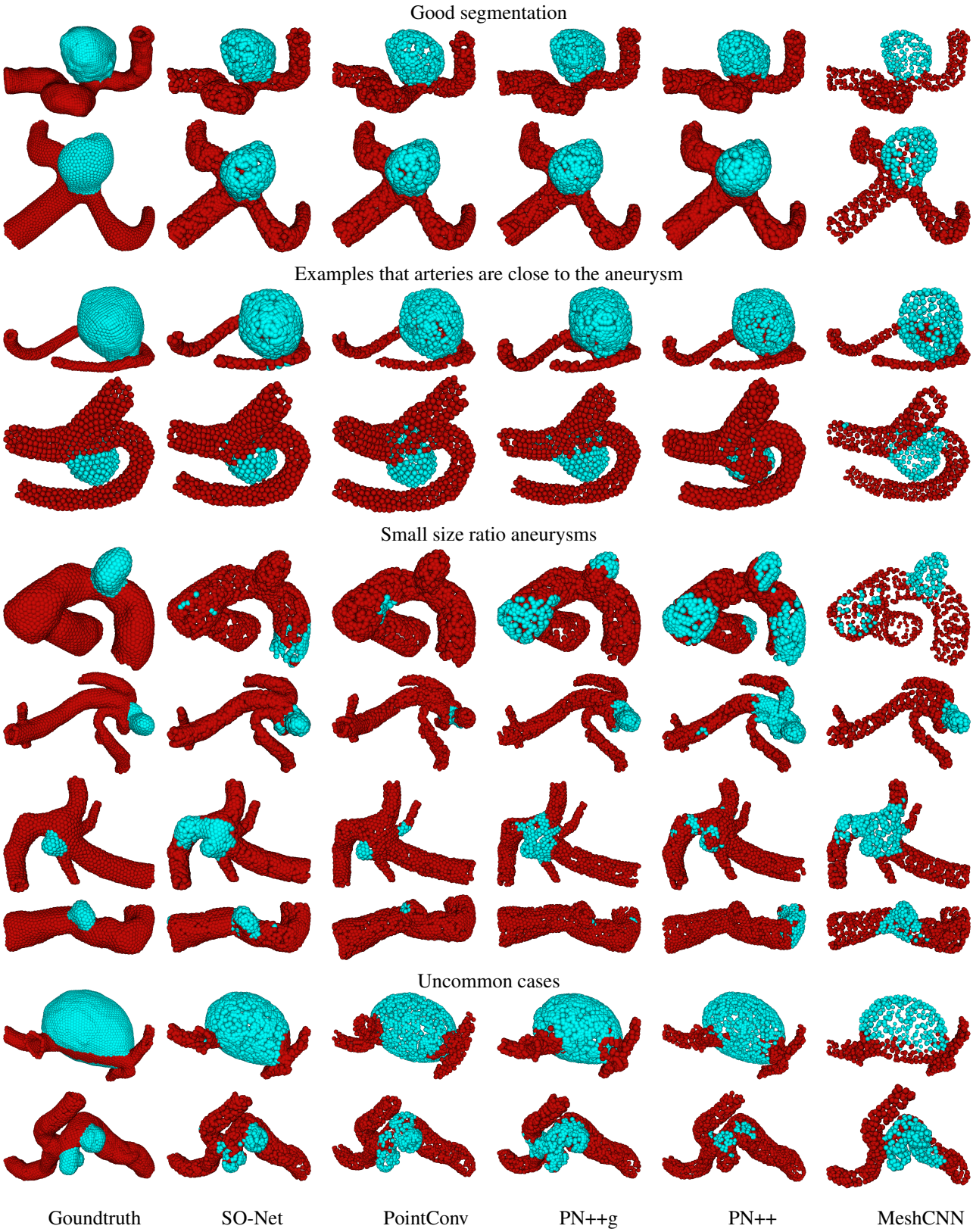


Figure 4: Results comparison of five networks.

References

[1] Blender. https://www.blender.org/. 2

[2] MeshLab. http://www.meshlab.net/. 2

[3] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018. 3

[4] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):90, 2019. 3

[5] Romhild M Hoogeveen, Chris JG Bakker, and Max A Viergever. Limits to the accuracy of vessel diameter measurement in mr angiography. *Journal of Magnetic Resonance Imaging*, 8(6):1228–1235, 1998. 1

[6] Taichi Kin, Hirofumi Nakatomi, Masaaki Shojima, Minoru Tanaka, Kenji Ino, Harushi Mori, Akira Kunimatsu, Hiroshi Oyama, and Nobuhito Saito. A new strategic neurosurgical planning tool for brainstem cavernous malformations using interactive computer graphics with multimodal fusion images. *Journal of neurosurgery*, 117(1):78–88, 2012. 1

[7] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 3

[8] Jiaxin Li, Ben M Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9397–9406, 2018. 2

[9] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. 3

[10] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2

[11] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 2

[12] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019. 3

[13] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 2

[14] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 87–102, 2018. 3