Appendix: Tree Instance Segmentation with Temporal Contour Graph

A1. Additional Results

In this section, we expand upon the methods and results we presented in our main paper. We further explain the reasoning and rationales for making certain choices in our methodology.

Additional qualitative results. Fig. A1 illustrates several additional input-to-output data flows on various types of forest appearances. In particular, we selected varying shadows, lighting, shapes, colors, and distributions. This example shows the robustness of our method.

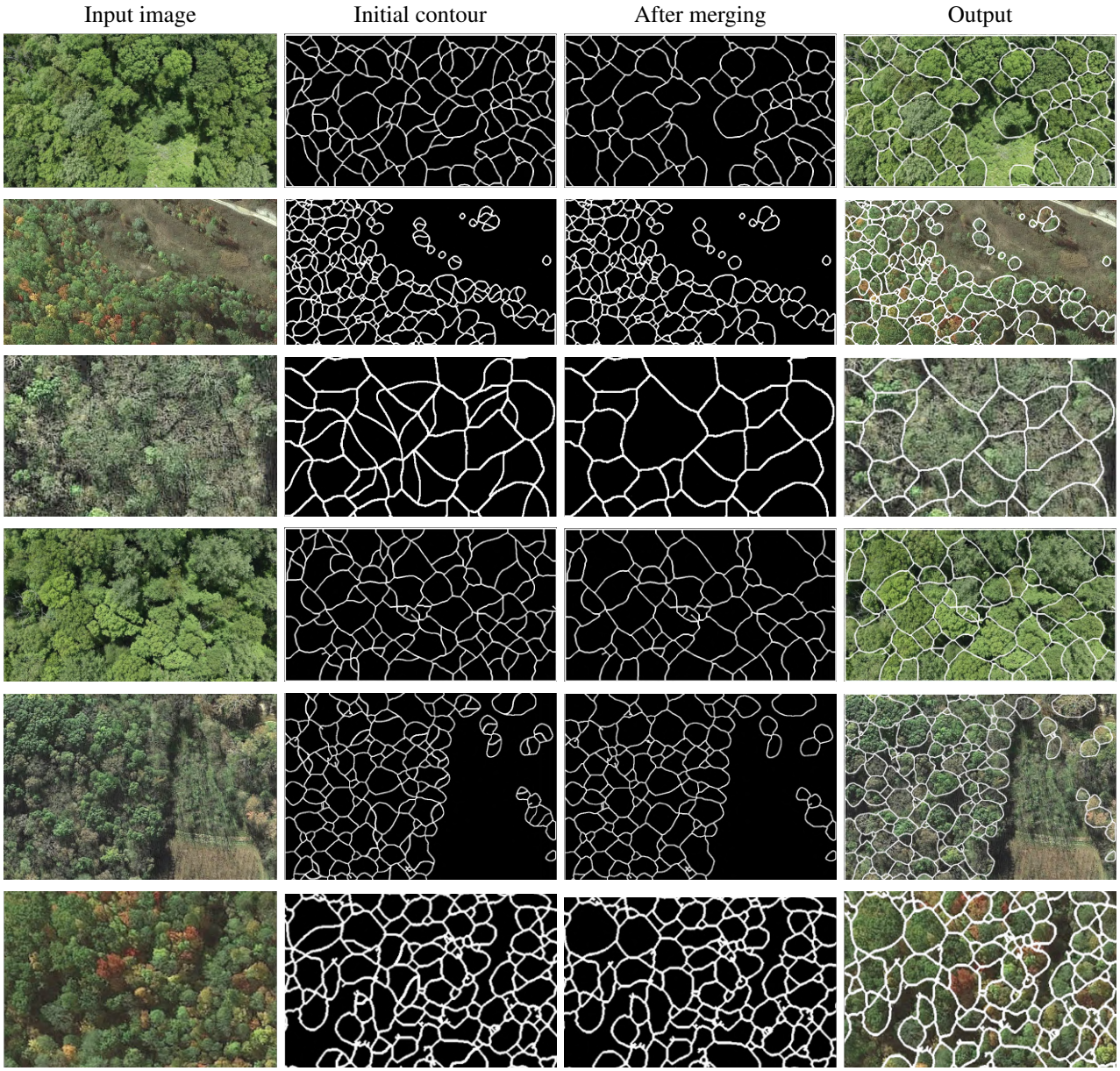

Figure A1. Additional qualitative results. We visualize the contour map before and after contour merging.

Additional visual comparisons to baselines: Here, we present more qualitative comparisons to other baselines in Figs. Fig. A2 and Fig. A3.

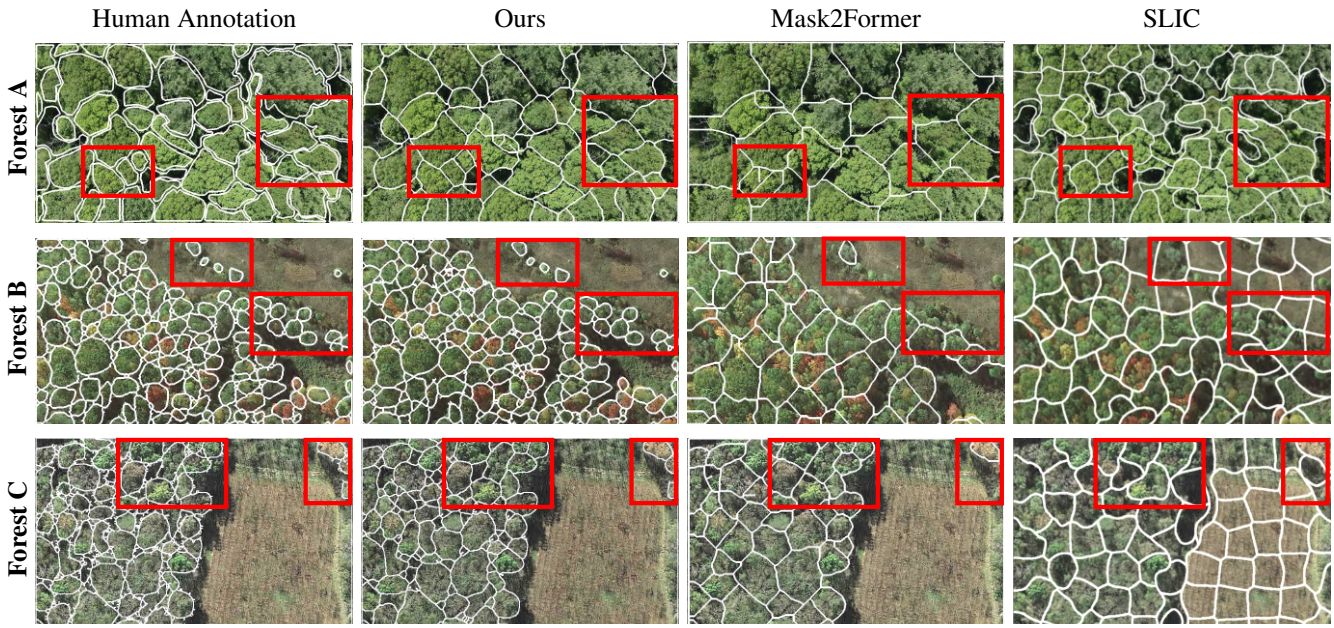

Figure A2. **Extended Qualitative Comparisons.** Comparisons of our approach vs. Mask2Former and SLIC. We observe that our approach produces instance segmentation masks that more closely match the human annotation.

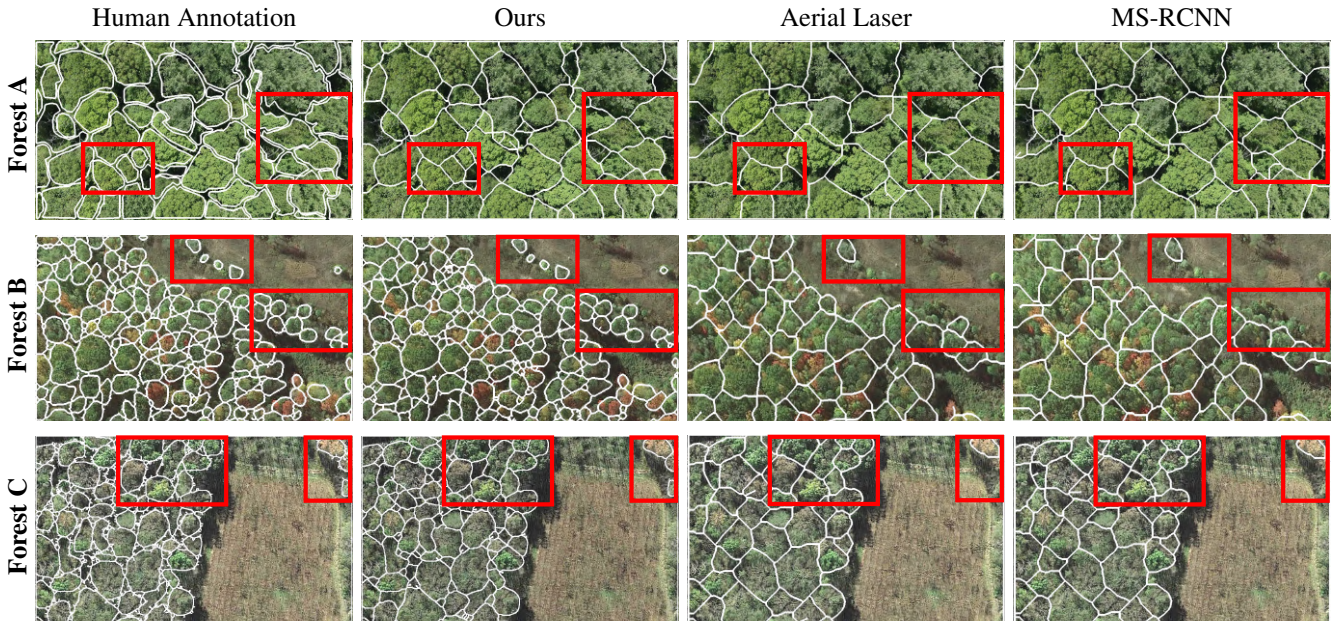

Figure A3. **More Qualitative comparisons.** Comparisons of our approach vs. Aerial Laser Model and Mask Scoring RCNN. We observe that our approach produces instance segmentation masks that more closely match the human annotation.

Comprehensive ablation. To offer a more detailed ablation on the effect of out-of-distribution data, we conduct ablations and report quantitative metrics for Forest B and Forest C (in addition to Forest A, which was presented in the main paper). In addition, we show the (very low) performance of just using the initial contours as determined by the edge detection network. Tab. A1 presents the aforementioned metrics.

Justification for Guo-Hall skeletonization [23]. Our method uses Guo Hall (GH) skeletonization [23] to convert and simplify grayscale contours to binary maps. We illustrate the effects of using GH in Fig. A4, and the resulting segmentation

Ablated feature	Forest A				Forest B				Forest C			
	AP ↑	AP_{50} ↑	AP_{75} ↑	Acc. ↑	AP ↑	AP_{50} ↑	AP_{75} ↑	Acc. ↑	AP ↑	AP_{50} ↑	AP_{75} ↑	Acc. ↑
All Features	74.5	81.6	72.8	91.5	69.8	76.2	71.5	86.8	70.1	75.4	72.5	90.2
All — aspect ratio	67.6	72.9	68.4	82.4	55.3	60.7	57.5	62.2	57.5	62.2	57.7	56.1
All — solidity	69.2	74.1	66.2	80.1	63.5	75.1	70.2	78.8	65.9	73.6	65.9	71.3
All — self-occlusion	62.9	77.3	71.1	78.2	56.2	58.4	52.5	62.7	56.5	61.8	60.2	66.5
All — patch	51.3	56.8	63.2	71.8	60.2	57.9	61.1	65.9	57.2	54.5	51.9	59.2
All — deviation	59.8	62.2	64.1	70.9	61.3	64.6	62.1	60.4	50.5	62.9	61.4	65.3
All — area	55.1	58.8	61.6	68.3	51.1	52.5	44.6	54.5	54.7	57.2	56.5	59.4
All — neighbor sim.	57.4	61.7	59.6	63.5	54.9	59.7	58.9	59.9	68.6	72.2	70.7	79.5
All — Guo-Hall	32.6	39.2	30.8	41.7	28.8	36.3	31.6	44.2	30.7	38.1	34.7	42.2
Initial Contour	25.7	29.1	27.3	38.5	21.6	22.5	21.8	31.3	28.4	34.8	29.7	36.9

Table A1. Ablation analysis of all forest datasets: Performance of our approach on real-world forest datasets as we ablate each feature. Also, we show performance when not using Guo-Hall skeletonization and also when directly using the initial contours. The table is sorted on count accuracy of Forest A to stay consistent with base paper.

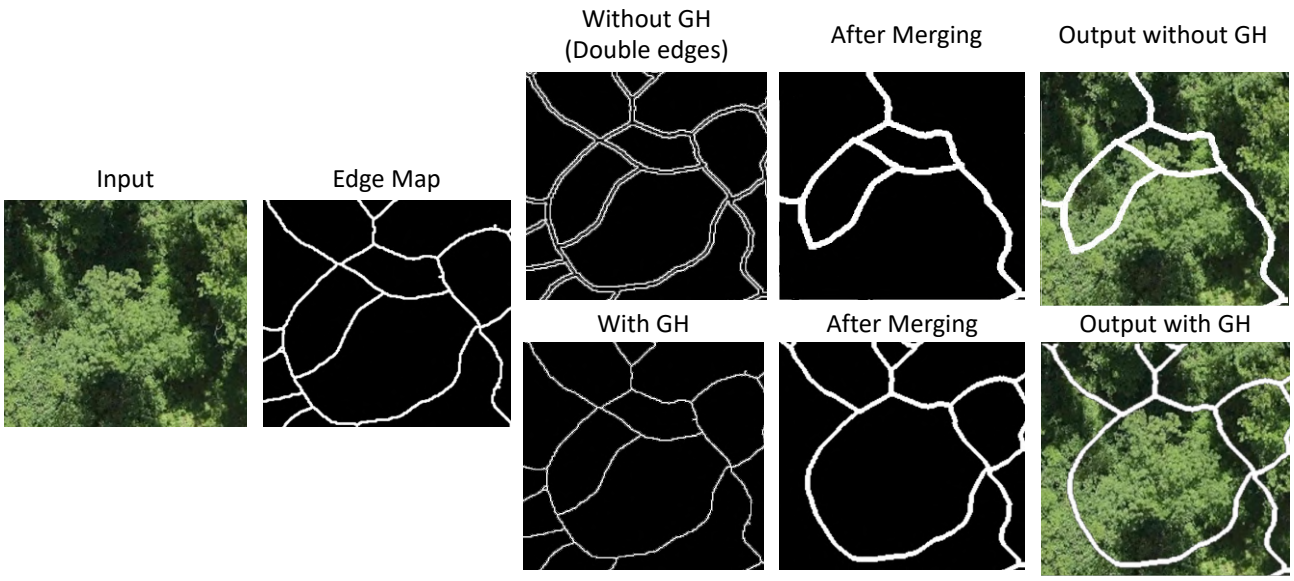

Figure A4. Impact of Guo-Hall Skeletonization (GH) [23]. We visualize, with and without Guo-Hall, the resulting intermediates throughout the proposed method.

is different as can be seen in the top and bottom rows. The initial edge detector generates edges with doubled contours. When constructing the contour graph, without GH, we would potentially receive two instead of one contour for each side (see Fig. A4). This results in double edges in the edge maps and leads to incorrect connectivity between the nodes. This leads to incorrect contour merging. Such effect is also shown quantitatively in Tab. A1.

Hyperparameter tuning. We used 30×30 pixels for each node's representative patch. We selected this dimension through a hyperparameter tuning experiment using several choices of patch sizes as shown in Tab. A2. The resolution of 30×30 was chosen because it outperformed both larger and smaller patches. Using larger patches would pick up pixels from neighboring nodes that are undesirable at this point of the pipeline. Graph Fig. A5 shows the values of AP and Acc. with varying patch sizes.

Patch Size (px)	AP ↑	AP_{50} ↑	AP_{75} ↑	Acc. ↑
10 x 10	52.1	61.8	57.9	74.4
20 x 20	61.5	74.8	68.1	87.2
30 x 30	**68.9**	**77.3**	**71.1**	**90.1**
40 x 40	64.2	75.9	70.3	88.3

Table A2. Hyperparameter tuning for patch size on Forest A.

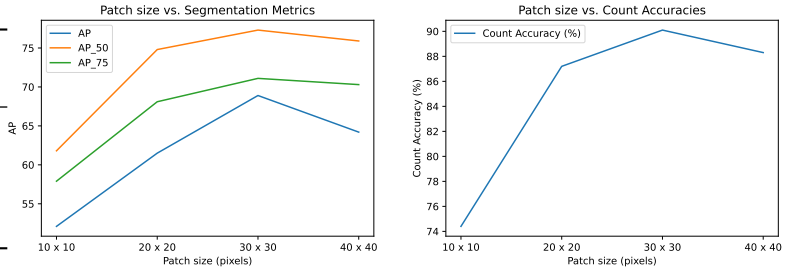

Figure A5. Plots showing AP performance and count accuracies vs. patch sizes.

A2. Implementation Details

Compute resource. All our experiments and training were done on a machine with AMD EPYC 7302 16-Core Processor with 3 GHz clock speed, and 128 GB RAM. Our model is trained with a single NVIDIA RTX 3090 GPU.

Implementation details. Our work is implemented in Python using PyTorch. We utilize Pytorch Geometric [18] for graph processing and OpenCV for image processing. The deep contour/edge map generation is generated using a deep edge detection network named Pixel Difference Networks for Efficient Edge Detection (PIDINet) [56] from repository. We retrained the edge detection network with our annotated dataset. It took approximately 22 hours to train the network with a base learning rate of 0.01 and weight decay of 0.0005 for 200 epochs. The optimizer used was ADAM [30]. Then, the network could generate the initial edge maps necessary for our pipeline.

We use a GCN [31] to perform message passing of node embedding, where input features capture each node's accompanying neighborhood information. In our GCN implementation, we set our walk length to two, which corresponds to K in the original GCN paper[31]. In this way, we aggregate information from at most two steps (i.e., neighbor of neighbors) away for each node embedding. We used a base learning rate of 0.1 and weight decay of 0.005 with ADAM [30] and trained for 200 epochs. It took approximately 15 hours to process and train the model on our dataset.

Finally, the MLP used for edge classification is trained with a base learning rate of 0.001 and momentum of 0.9. using ADAM [30]. It took under two hours to train the MLP classifier for 500 iterations with early stopping at 10 steps.

Baseline implementation details. We document how we train/use the nine baseline models on our dataset.

Mask-RCNN [27]. We use the publicly available implementation Matterport official GitHub repository (which was acknowledged by the original authors). We used the ResNet-101 backbone and retrained on our labeled dataset. We train the modeling using SGD with a learning rate of 0.001, a momentum of 0.9, and a weight decay of 0.0001. It took approximately four days to train 200 epochs on our dataset.

Swin-T [41] backbone with Mask-RCNN [27]. We use the official branch in Github from the author's affiliation – Microsoft Research, which was specific for instance segmentation. We train the model using a base learning rate of 0.001 and weight decay of 0.0001 using ADAM [30]. This baseline was trained for 200 epochs, which takes approximately three days.

TraDes [62]. We use the official repository on GitHub. We used the MoT17 [44] pre-trained weights and evaluated our validation dataset to confirm consistency. We note that MoT17 also had trees and other vegetation classes subjectively similar to ours.

BoundaryFormer [34]. We use the official implementation on GitHub. We retrained the model using our own dataset. It took approximately 2 days to finish the training for 200 epochs. We set the base learning rate is set to 0.02 and use the ADAM [30] optimizer.

Mask Scoring RCNN [28]. We use the official implementation on GitHub. We retrained the model using our own dataset in COCO format. It took approximately one day to finish the training for 100 epochs. We set the base learning rate to 0.02 with weight decay of 0.0001 and use the SGD optimizer.

Mask2Former [14]. We use the official implementation on GitHub. We retrained the model using our own dataset in COCO format. It took approximately eight hours to finish the training for 100 epochs. We set the base learning rate to 0.0001, weight decay of 0.05 and use the ADAM [30] optimizer.

Aerial Laser Model [57]. It is an improved parameterized version of YOLOv4. We make the instructed changes to the PyTorch implementation on GitHub. We retrained the model using our own dataset. It took approximately 12 hours to finish the training for 200 epochs. We set the base learning rate to 0.0013, weight decay of 0.0005, the momentum of 0.949 and use the ADAM [30] optimizer.

SLIC (Superpixel) [3]. As this is a traditional algorithm as opposed to a deep learning model, we used the SLIC implementation from the Python Scikit-Learn package.

OCISIS [15]. We use the official implementation on GitHub. We retrained the model using our own dataset in COCO format. It took approximately two days to finish the training for 200 epochs. We set the base learning rate to 0.01, weight decay of 0.0001, and momentum of 0.9 and used the SGD optimizer.

References

[1] Real time design and animation of fractal plants and trees. *ACM SIGGRAPH Comput. Graph.*, 1986. 3

[2] Review on convolutional neural networks (CNN) in vegetation remote sensing. *ISPRS P&RS)*, 2021. 3

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *TPAMI*, 2012. 6, 15

[4] M. Aono and T.L. Kunii. Botanical tree image generation. *IEEE Computer Graphics and Applications*, 1984. 3

[5] Hasan Asy'ari Arief, Geir-Harald Strand, Håvard Tveite, and Ulf Geir Indahl. Land cover segmentation of airborne lidar data using stochastic atrous network. *Remote Sensing*, 2018. 1

[6] James Arvo and David Kirk. Modeling plants with environment-sensitive automata. In *Proc. of Ausgraph*, 1988. 3

[7] Bedrich Benes, Nathan Andrysco, and Ondřej Šťava. Interactive modeling of virtual ecosystems. In *Proc. Eurographics Conference on Natural Phenomena*, 2009. 5

[8] Bedrich Benes and Erik Uriel Millán. Virtual climbing plants competing for space. In *Computer Animation*. IEEE Computer Society, 2002. 3

[9] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. YOLOv4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2

[10] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 3

[11] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. MaskLab: Instance segmentation by refining object detection with semantic and direction features. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 3

[12] Steven W. Chen, Guilherme V. Nardari, Elijah S. Lee, Chao Qu, Xu Liu, Roseli Ap. Francelin Romero, and Vijay R. Kumar. Sloam: Semantic lidar odometry and mapping for forest inventory. *IEEE RA-L*, 2020. 3

[13] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 3

[14] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 6, 15

[15] Hisham Cholakkal, Guolei Sun, Fahad Shahbaz Khan, and Ling Shao. Object counting and instance segmentation with image-level supervision. In *CVPR*, 2019. 6, 15

[16] Phillippe De Reffye, Claude Edelin, Jean Françon, Marc Jaeger, and Claude Puech. Plant models faithful to botanical structure and development. *ACM Siggraph Comp. Graph.*, 1988. 3

[17] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, 2003. 4, 8

[18] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019. 14

[19] Adnan Firoze, Bedrich Benes, and Daniel Aliaga. Urban tree generator: spatio-temporal and generative deep learning for urban tree localization and modeling. *The Visual Computer*, 2022. 2

[20] Ross Girshick. Fast R-CNN. In *Int. Conf. Comput. Vis.*, 2015. 2

[21] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2014. 3

[22] Ned Greene. Voxel space automata: Modeling with stochastic growth processes in voxel space. *ACM SIGGRAPH Comp. Graph.*, 1989. 3

[23] Zicheng Guo and Richard W. Hall. Parallel thinning with two-subiteration algorithms. *ACM Commun.*, 1989. 4, 12, 13

[24] Torsten Hädrich, Daniel T. Banuti, Wojtek Pałubicki, Sören Pirk, and Dominik L. Michels. Fire in paradise: Mesoscale simulation of wildfires. *ACM Trans. Graph.*, 2021. 3

[25] Torsten Hädrich, Bedrich Benes, Oliver Deussen, and Sören Pirk. Interactive modeling and authoring of climbing plants. 2017. 3

[26] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Eur. Conf. Comput. Vis.*, 2014. 3

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Int. Conf. Comput. Vis.*, 2017. 1, 3, 6, 14

[28] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring R-CNN. In *CVPR*, 2019. 6, 14

[29] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 1, 3

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 14, 15

[31] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Int. Conf. Learn. Represent.*, 2017. 4, 14

[32] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 3

[33] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Eur. Conf. Comput. Vis.*, 2018. 2

[34] Justin Lazarow, Weijian Xu, and Zhuowen Tu. Instance segmentation with mask-supervised polygonal boundary transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 3, 6, 14

[35] Bosheng Li, Jacek Kałużny, Jonathan Klein, Dominik L. Michels, Wojtek Pałubicki, Bedrich Benes, and Sören Pirk. Learning to reconstruct botanical trees from single images. *ACM Trans. Graph.*, 2021. 2, 5

[36] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, Yiduo Li, Bo Zhang, Yufei Liang, Linyuan Zhou, Xiaoming Xu, Xiangxiang Chu, Xiaoming Wei, and Xiaolin Wei. Yolov6: A single-stage object detection framework for industrial applications, 2022. 2

[37] Dongze Lian, Jing Li, Jia Zheng, Weixin Luo, and Shenghua Gao. Density map regression guided detection network for rgb-d crowd counting and localization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 1, 2

[38] Lingbo Liu, Jiaqi Chen, Hefeng Wu, Guanbin Li, Chenglong Li, and Liang Lin. Cross-modal collaborative representation learning and a large-scale rgbt benchmark for crowd counting. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 1, 2

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *Eur. Conf. Comput. Vis.*, 2016. 2

[40] Xinni Liu, Fengrong Han, Kamarul Hawari Ghazali, Izzeldin Ibrahim Mohamed, and Yue Zhao. A review of convolutional neural networks in remote sensing image. In *ICSCA*, 2019. 3

[41] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2022. 3, 14

[42] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Int. Conf. Comput. Vis.*, 2021. 3, 6

[43] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Int. Conf. Learn. Represent.*, 2013. 4

[44] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, Mar. 2016. arXiv: 1603.00831. 14

[45] Radomír Měch and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *ACM SIGGRAPH Comp. Graph.*, 1996. 3, 5

[46] Alejandro Newell and Jia Deng. Pixels to graphs by associative embedding. In *Adv. Neural Inform. Process. Syst.*, 2017. 1

[47] Till Niese, Sören Pirk, Matthias Albrecht, Bedrich Benes, and Oliver Deussen. Procedural urban forestry. *ACM Trans. Graph.*, 2022. 3, 5

[48] Masanori Onishi and Takeshi Ise. Explainable identification and mapping of trees using uav rgb image and deep learning. *Nature: Scientific Reports*, 2021. 3

[49] Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Měch, and Przemyslaw Prusinkiewicz. Self-organizing tree models for image synthesis. *ACM Trans. Graph.*, 2009. 3, 5

[50] Sören Pirk, Till Niese, Torsten Hädrich, Bedrich Benes, and Oliver Deussen. Windy trees: Computing stress response for developmental tree models. *ACM Trans. Graph.*, 2014. 3

[51] Tomas Polasek, David Hrusa, Bedrich Benes, and Martin Cadik. Ictree: Automatic perceptual metrics for tree models. *ACM Trans. Graph.*, 2021. 3, 5

[52] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. 1990. 3

[53] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016. 2

[54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Adv. Neural Inform. Process. Syst.*, 2015. 2

[55] Alvy Ray Smith. Plants, fractals, and formal languages. In *ACM SIGGRAPH Comp. Graph.*, 1984. 3

[56] Z. Su, Wenzhe Liu, Zitong Yu, Dewen Hu, Qing Liao, Qi Tian, Matti Pietikäinen, and Li Liu. Pixel difference networks for efficient edge detection. In *Int. Conf. Comput. Vis.*, 2021. 4, 14

[57] Chenxin Sun, Chengwei Huang, Huaiqing Zhang, Bangqian Chen, Feng An, Liwen Wang, and Ting Yun. Individual tree crown segmentation and crown width extraction from a heightmap derived from aerial laser scanning data using a deep learning framework. *Frontiers in plant science*, 2022. 6, 15

[58] Mingxing Tan, Ruoming Pang, and Quoc V Le. EfficientDet: Scalable and efficient object detection. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020. 2

[59] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Int. Conf. Comput. Vis.*, 2019. 2

[60] Ondrej Šťava, Sören Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. Inverse procedural modelling of trees. *Comput. Graph. Forum*, 2014. 2, 5

[61] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv preprint arXiv:2207.02696*, 2022. 2, 6

[62] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Int. Conf. Comput. Vis.*, 2021. 3, 6, 14

[63] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal attention for long-range interactions in vision transformers. In *Adv. Neural Inform. Process. Syst.*, 2021. 3

[64] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018. 4

[65] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable ConvNetsv2: More deformable, better results. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2019. 3