M2F3D: Mask2Former for 3D Instance Segmentation

Jonas Schult[1] Alexander Hermans[1] Francis Engelmann[2] Siyu Tang[3] Otmar Hilliges[3] Bastian Leibe[1]
[1]RWTH Aachen University [2]ETH AI Center [3]ETH Zurich

Abstract

In this work, we show that the top performing Mask2Former approach for image-based segmentation tasks works surprisingly well when adapted to the 3D scene understanding domain. Current 3D semantic instance segmentation methods rely largely on predicting centers followed by clustering approaches and little progress has been made in applying transformer-based approaches to this task. We show that with small modifications to the Mask2Former approach for 2D, we can create a 3D instance segmentation approach, without the need for highly 3D specific components or carefully hand-engineered hyperparameters. Initial experiments with our M2F3D model on the ScanNet benchmark are very promising and sets a new state-of-the-art on ScanNet test ($+0.4\,mAP_{50}$).

1. Introduction

For image-based tasks, CNN architectures [9, 10, 28, 29] have dominated for a long time. However, recently, we observe a strong shift towards transformer-based models [3, 5, 20]. We now even see universal architectures such as Mask2Former (M2F) [3] that can directly be applied to a whole range of different segmentation tasks, achieving state-of-the-art results. This move towards transformer-based models is less pronounced for tasks performed on 3D point clouds. Most current transformer-based methods focus on either 3D object detection [21, 23] or semantic segmentation [16, 26]. Often very specific modifications need to be made to deploy attention modules at low computational cost due to the quadratic memory complexity of the attention matrix. Here, we investigate whether the Mask2Former meta architecture can also generalize to 3D segmentation tasks. In particular, we focus on instance segmentation of 3D indoor scans which has not yet been explored by transformer-based methods. In contrast to concurrent 3D transformer-based methods, we mitigate the limiting factor of attention complexity by cross-attending a fixed number of instance queries with point cloud features. In contrast to most approaches for 3D instance segmentation [2, 6, 13, 27, 32], we do not rely on any center voting, heuristic grouping mechanism, or non-maximum suppression step, making the architecture agnostic to domain specific and carefully hand-engineered hyperparameters. We show that with a few modifications, a Mask2Former can indeed be applied to the task of instance segmentation on 3D point clouds. We use a sparse convolutional backbone to extract features from a point cloud, we introduce a loss defined over point cloud segments to improve the memory consumption during training, and carefully ablate a set of design decisions such as the use of nonparametric queries proposed by 3DETR [23]. Using the resulting setup, M2F3D achieves state-of-the-art results for 3D instance segmentation on the ScanNet v2 benchmark, even surpassing architectures that are highly tuned for 3D point cloud processing.

2. Method

Our proposed model for 3D segmentation tasks consists of three components and draws inspiration from the Mask2Former architecture [3]. ① A feature backbone network (consisting of an encoder and a decoder) that extracts multi-scale point features from the input point cloud. ② A query refinement step, which takes as input a set of object queries and multi-scale point features and then iteratively lets the queries cross-attend to the input features in order to refine their representation of objects in the scene. And finally, ③ the mask prediction, which creates a full resolution mask and predicts a class for each query. Figure 1 visualizes these three components. In the following sections, we present these components individually and highlight important design decisions which are key for 3D point clouds.

① Feature Backbone. The backbone provides higher level pointwise features. We use a sparse convolutional backbone based on the MinkowskiEngine [4] as it constitutes a good trade-off between accuracy and efficiency. We extract features using a sparse U-Net, consisting of a symmetric convolutional encoder and decoder with skip connections between them. We build a multi-scale representation using the coarsest 4 feature maps of the decoder and use the full resolution feature map for predicting instance masks.

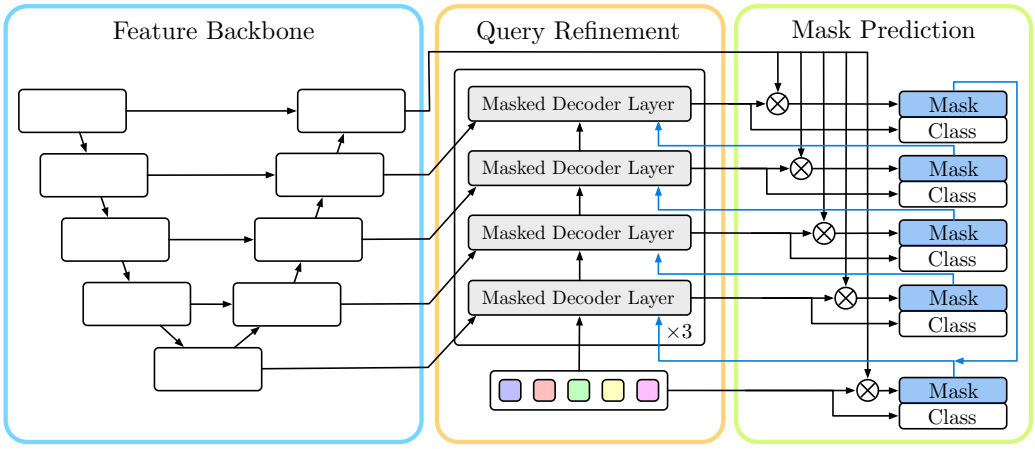


Figure 1. **Mask2Former3D Architecture.** We adopt the same meta architecture as Mask2Former [3] which can be roughly divided into three main components. ① (*Feature Backbone*) We extract multi-scale features from the input point cloud, ② (*Query Refinement*) we update query features by cross attending to multi-scale point cloud features and ③ (*Mask Prediction*) for each query, we obtain a full resolution instance mask and class prediction.

② Query Refinement. We aggregate information from our multi-scale representation to iteratively refine a set of queries to describe objects in the scene. We adopt the same transformer decoder variant as in Mask2Former [3], *i.e.* applying cross attention from queries to point cloud features, followed by self-attention between queries. This removes the quadratic complexity requirements introduced by the attention matrix between point cloud features. Within each decoder block, query features are updated in a coarse-to-fine fashion starting with the coarsest point cloud features.

In contrast to the 2D domain, 3D point clouds have a variable number of points, *i.e.* they are not organized in a dense grid of predefined size. As we cannot batch point clouds of variable sizes, leveraging a standard transformer is non-trivial. We pad point clouds to fit the size of the largest entry in the batch and mask out attention where needed. In case the point cloud size exceeds a certain threshold, we resort to sampling point features to keep the memory requirements in bounds.

Inspired by 3DETR [23], we leverage nonparametric queries to obtain point cloud specific initial object queries. The object queries are split into two parts. The query features, from which the query vectors are computed and the query positions, from which we compute positional embeddings. In the nonparametric query setting, we sample a set of points via furthest point sampling (FPS), and use their location to compute the positional embeddings for the queries, while we set the actual query features to zero. In essence, this provides a location bias to the object queries. Alternatively, we additionally use the backbone features of the sampled points as query features. The three different settings are depicted in Figure 2. Moreover, we use random Fourier [31] instead of the typical sinusoidal positional embeddings [1, 3] for points fed into the transformer decoder.

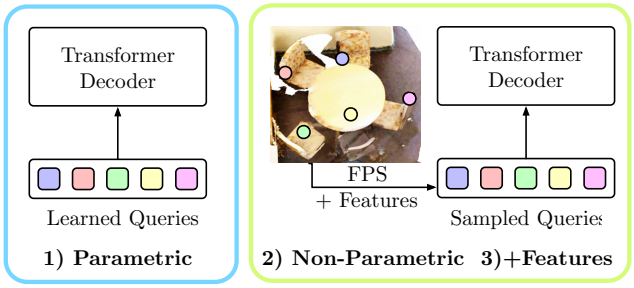


Figure 2. **Parametric vs Nonparametric Queries.** While M2F uses learned parametric object queries, we use furthest point sampling to sample query positions which are fed to the transformer decoder. As a third setting, we additionally feed the point features from the sampled points as object feature queries into the decoder.

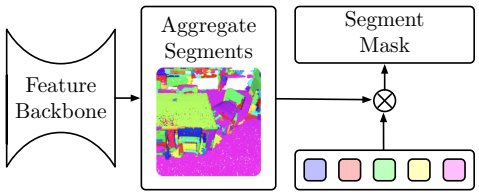


Figure 3. **Feature Aggregation Across Segments.** Instead of computing mask logits and mask losses for every voxel, we average features per segment which saves significant memory.

③ Mask Prediction. In contrast to Mask2Former, we do not predict masks over pixels or 3D points but over spatially contiguous sets of 3D points, *i.e.* segments, which we obtain by using the graph-based Felzenszwalb and Huttenlocher algorithm [7] (*c.f.* Fig. 3). We average features of points falling into the same segment and calculate the loss on segments instead of individual points. As we have 2 orders of magnitude fewer segments than points, this results

in an overall reduced memory consumption. Mask2Former leverages PointRend [14] to reduce the memory complexity. Since this relies on interpolation between points and we extract features using a sparse backbone, segment level aggregation was easier to realize, while at the same time incorporating a meaningful geometric prior.

3. Experiments

Dataset. We use the ScanNet v2 benchmark dataset containing a total of 1613 richly-annotated 3D indoor scenes, which we split according to the default train/validation/test split. The scenes are annotated with 20 semantic classes out of which 18 are used for the 3D instance segmentation evaluation. We use the default average precision evaluation metrics mAP, mAP_{25}, and mAP_{50}.

3.1. Training Details

We use the same mask and classification losses and loss weights as Mask2Former. In particular, we use the class and full resolution mask predictions after each transformer decoder layer to calculate auxiliary losses. As our backbone network, we deploy a Minkowski Res16UNet34C [4]. We train for a total of 600 epochs using AdamW [22] and a one-cycle learning rate schedule [30] with a maximal learning rate of 10^{-4}. In early experiments, we validated that longer training (1000 epochs) does not improve results. Training takes roughly 30 hours on an NVIDIA A40 GPU. We perform standard data augmentation: horizontal flipping, random rotations around the z-axis, elastic distortion [29] and random scaling. Color augmentations include jittering, brightness and contrast augmentations. As post-processing, we smoothen final predictions by finding dominant labels within segments (obtained from a graph-based segmentation [7], similar to OccuSeg [8] or Mix3D [25]).

3.2. Comparison with state-of-the-art methods

Table 1 shows a comparison to the state-of-the-art 3D instance segmentation approaches on the ScanNet validation and test set, showing that we improve over the previous best method by $0.4\,\text{mAP}_{50}$. At the same time, we can match the speed of the previously fastest method, indicating that M2F3D could be interesting for real-world applications. Some qualitative results from our model can be seen in the supplementary material.

3.3. Additional Experiments

Apart from the main method comparison, we highlight some of the aspects of our approach in more detail. Several of these, we did not yet include in our final benchmark model, but only experimented with later on.

Parametric vs. Non-parametric queries. Table 2 shows the effects of using parametric or non-parametric queries.

Method	Runtime (in ms)	Validation mAP_{50}	Validation mAP_{25}	Test mAP	Test mAP_{50}	Test mAP_{25}
SGPN [33]	158439	-	-	4.9	14.3	39.0
GSPN [35]	12702	37.8	53.4	-	30.6	-
3D-SIS [12]	-	18.7	35.7	16.1	38.2	55.8
MASC [18]	-	-	-	25.4	44.7	61.5
PanFusion [24]	-	-	-	21.4	47.8	69.3
3D-Bonet [34]	9202	-	-	25.3	48.8	68.7
MTML [15]	-	40.2	55.4	28.2	54.9	73.1
3D-MPA [6]	-	59.1	72.4	35.5	61.1	73.7
Dyco3D [11]	-	57.6	72.9	39.5	64.1	76.1
PE [36]	-	57.1	73.8	39.6	64.5	77.6
PointGroup [13]	452	56.7	71.3	40.7	63.6	77.8
GICN [19]	8615	-	-	34.1	63.8	78.8
OccuSeg [8]	1904	60.7	71.9	48.6	67.2	74.2
SSTNet [17]	428	64.3	74.0	50.6	69.8	78.9
HAIS [2]	**339**	64.4	75.6	45.7	69.9	80.3
SoftGroup [32]	345	67.6	78.9	50.4	76.1	**86.5**
M2F3D (Ours)	**339**	**72.2**	**81.4**	**56.1**	**76.5**	85.3

Table 1. **Instance Segmentation results on ScanNet (15/04/22).** We report state-of-the-art performance on the ScanNet validation and test set while being computationally efficient. Inference speed is averaged over the validation set and computed on a TITAN X GPU (*c.f.* [32]), excluding the calculation of segments.

	Query Positions	Query Features	mAP	mAP_{50}	mAP_{25}
①	parametric	parametric	45.1	63.3	76.6
②	sampled point	zeros	**46.8**	65.1	76.8
③	sampled point	point features	45.5	**65.6**	**78.3**

Table 2. **Query Types.** We explore two variants for query positions and features. Parametric queries ① are learned during training. Nonparametric queries consist of sampled point positions ② and potentially their features ③, resemble scene-specific queries.

For ① parametric queries, we follow Mask2Former [3]. For ②/③ non-parametric queries, we follow 3DETR [23]. ② only uses the positions of sampled points but uses query features that are set to zero-initialized. We also experiment with additionally using the point features from the sampled points as query features ③. Both non-parametric variants work slightly better than the parametric variant and have the added benefit of sampling different amounts of points during inference. The difference is significantly smaller than for 3DETR [23], potentially based on the direct assignment of points to queries based on the mask supervision.

Masks on segments. Table 3 evaluates two aspects of how segments are used in our approach. First, we aggregate features within segments during the mask computation in our model and only compute mask logits (and losses) on a segment level. This reduces the required memory and enables us to train models with a more fine-grained voxel

Voxel Size	Segment Aggreg.	Post Proc.	Train Memory	mAP	mAP_{50}	mAP_{25}
5cm			29.1GB	41.0	64.5	79.1
5cm		✓	29.1GB	47.7	67.2	79.2
5cm	avg		26.5GB	41.3	62.9	77.1
5cm	avg	✓	26.5GB	46.8	65.1	76.8
2cm			-	-	-	-
2cm	avg		44.5GB	51.5	71.5	82.0
2cm	avg	✓	44.5GB	**53.2**	**72.4**	**82.0**

Table 3. **Masks on Segments.** Segment level post-processing improves results by up to 6.7 mAP. Segment aggregation during mask computation reduces performance slightly, but it enables training on smaller voxels.

Positional Embedding	σ	mAP	mAP_{50}	mAP_{25}
Sine/Cosine	-	47.3	67.0	79.1
Random Fourier	1.0	46.8	65.1	76.8
Random Fourier	10.0	**48.0**	**67.6**	**79.3**

Table 4. **Positional Embeddings.** Random Fourier positional embeddings can be superior to sinusoidal positional embeddings depending on the scale σ used to sample projection parameters.

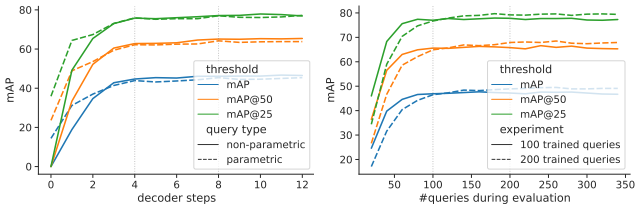
# Object Queries	mAP	mAP_{50}	mAP_{25}
100	46.8	65.1	76.8
200	**48.2**	**66.5**	**79.1**

Table 5. **Number of Queries.** Training the network with more object queries results in a better performance.

size. However, at a slightly decreased performance of the model. Additionally, we can also apply a segment level post-processing at the point cloud level on our final predictions. This gives a quite significant boost, albeit it is less pronounced for the model trained on 2cm voxels.

Positional embeddings. Inspired by 3DETR, we switched to random Fourier embeddings [31], which work slightly better. Table 4 shows that for non-parametric queries they result in slightly worse performance when the scale of the Gaussian, from which the projection parameters are sampled, is not selected optimally. With a proper scale, small improvements can be made, however, at the cost of additional randomness.

Number of object queries. While for most image-based tasks, M2F reports that 100 object queries yields best performance, we here find that increasing the number to 200 gives a significant performance boost as can be seen in Table 5. However, this does come at an increased computational cost, which would require additional modifications to make the training of the 2 cm voxel model feasible. As such we use 100 object queries in our benchmark model.



(a) Decoder steps (b) Number of sampled queries

Figure 4. **Varying of Decoder Steps/Queries.** During inference we can evaluate a subset of the decoder steps or sample a different number of non-parametric queries.

Varying decoder parameters during inference. The transformer decoder goes through the coarse-to-fine hierarchy three times and a segmentation output can be created after every decoder layer. Figure 4a shows the performance of the different outputs for both parametric and non-parametric queries. In both cases the performance saturates rather quickly, but the parametric queries start off at a significantly higher performance level. When using non-parametric queries, we are not bound to sampling the same fixed amount of queries as used during training. Figure 4b shows that sampling more queries can have a small positive effect, fewer queries however decrease performance, albeit the drop is fairly small at first. It is interesting to see that the decrease is larger for the model trained with 200 queries. Overall it performs slightly better when sampling more queries, even though both models have the exact same amount of trainable parameters. Varying either of these two settings can be seen as a speed/accuracy trade-off.

4. Conclusion and Outlook

We have shown that the Mask2Former architecture is a generic approach that can, with a few modifications, be applied to the task of 3D instance segmentation, setting a new state-of-the-art performance on the competitive ScanNet v2 benchmark. It is one of the first transformer-based models that effectively works on 3D data and it has the potential to bridge the gap between 2D and 3D specific architectures. Our experiments explore parts of the design space of the M2F3D architecture and show that there is ample room for further improvement.

We plan to evaluate this model on further datasets (*e.g.* S3DIS), on different domains (outdoor autonomous driving scenarios), and perform more thorough experiments with other tasks such as semantic segmentation or object detection on 3D data. While some challenges will likely be encountered, we expect M2F3D to generalize well and hope that interesting synergies between the 2D and 3D domains can be created based on this shared architecture.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-End Object Detection with Transformers. In *ECCV*, 2020. 2

[2] Shaoyu Chen, Jiemin Fang, Qian Zhang, Wenyu Liu, and Xinggang Wang. Hierarchical Aggregation for 3D Instance Segmentation. In *ICCV*, 2021. 1, 3

[3] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. *arXiv:2112.01527*, 2021. 1, 2, 3, 6

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. 1, 3, 6

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1

[6] Francis Engelmann, Martin Bokeloh, Alireza Fathi, Bastian Leibe, and Matthias Nießner. 3D-MPA: Multi-Proposal Aggregation for 3D Semantic Instance Segmentation. In *CVPR*, 2020. 1, 3

[7] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient Graph-Based Image Segmentation. *IJCV*, 59(2):167–181, 2004. 2, 3

[8] Lei Han, Tian Zheng, Lan Xu, and Lu Fang. OccuSeg: Occupancy-aware 3D Instance Segmentation. In *CVPR*, 2020. 3

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 1

[11] Tong He, Chunhua Shen, and Anton van den Hengel. DyCo3D: Robust Instance Segmentation of 3D Point Clouds through Dynamic Convolution. In *CVPR*, 2021. 3

[12] Ji Hou, Angela Dai, and Matthias Nießner. 3D-SIS: 3D Semantic Instance Segmentation of RGB-D Scans. In *CVPR*, 2019. 3

[13] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. PointGroup: Dual-Set Point Grouping for 3D Instance Segmentation. In *CVPR*, 2020. 1, 3

[14] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image Segmentation as Rendering. In *CVPR*, 2020. 3

[15] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3D Instance Segmentation via Multi-Task Metric Learning. In *ICCV*, 2019. 3

[16] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified Transformer for 3D Point Cloud Segmentation. In *CVPR*, 2022. 1

[17] Zhihao Liang, Zhihao Li, Songcen Xu, Mingkui Tan, and Kui Jia. Instance Segmentation in 3D Scenes using Semantic Superpoint Tree Networks. In *ICCV*, 2021. 3

[18] Chen Liu and Yasutaka Furukawa. MASC: Multi-scale Affinity with Sparse Convolution for 3D Instance Segmentation. *arXiv:1902.04478*, 2019. 3

[19] Shih-Hung Liu, Shang-Yi Yu, Shao-Chi Wu, Hwann-Tzong Chen, and Tyng-Luh Liu. Learning Gaussian Instance Segmentation in Point Clouds. *arXiv:2007.09860*, 2020. 3

[20] 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 *ICCV*, 2021. 1

[21] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-Free 3D Object Detection via Transformers. *arXiv:2104.00678*, 2021. 1

[22] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019. 3

[23] Ishan Misra, Rohit Girdhar, and Armand Joulin. An End-to-End Transformer Model for 3D Object Detection. In *ICCV*, 2021. 1, 2, 3, 6

[24] Gaku Narita, Takashi Seno, Tomoya Ishikawa, and Yohsuke Kaji. Panopticfusion: Online volumetric semantic mapping at the level of stuff and things. In *IROS*, 2019. 3

[25] Alexey Nekrasov, Jonas Schult, Or Litany, Bastian Leibe, and Francis Engelmann. Mix3D: Out-of-Context Data Augmentation for 3D Scenes. In *3DV*, 2021. 3

[26] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3D Object Detection With Pointformer. In *CVPR*, 2021. 1

[27] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. In *ICCV*, 2019. 1

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *NeurIPS*, 2015. 1

[29] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 1, 3

[30] Leslie N Smith and Nicholay Topin. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. In *Artificial intelligence and machine learning for multi-domain operations applications*, 2019. 3

[31] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*, 2020. 2, 4, 6

[32] Thang Vu, Kookhoi Kim, Tung M Luu, Xuan Thanh Nguyen, and Chang D Yoo. SoftGroup for 3D Instance Segmentation on Point Clouds. In *CVPR*, 2022. 1, 3, 6

[33] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation. In *CVPR*, 2018. 3

[34] Bo Yang, Jianan Wang, Ronald Clark, Qingyong Hu, Sen Wang, Andrew Markham, and Niki Trigoni. Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds. In *NeurIPS*, 2019. 3

[35] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. GSPN: Generative Shape Proposal Network for 3D Instance Segmentation in Point Cloud. In *CVPR*, 2019. 3

[36] Biao Zhang and Peter Wonka. Point Cloud Instance Segmentation using Probabilistic Embeddings. In *CVPR*, 2021. 3

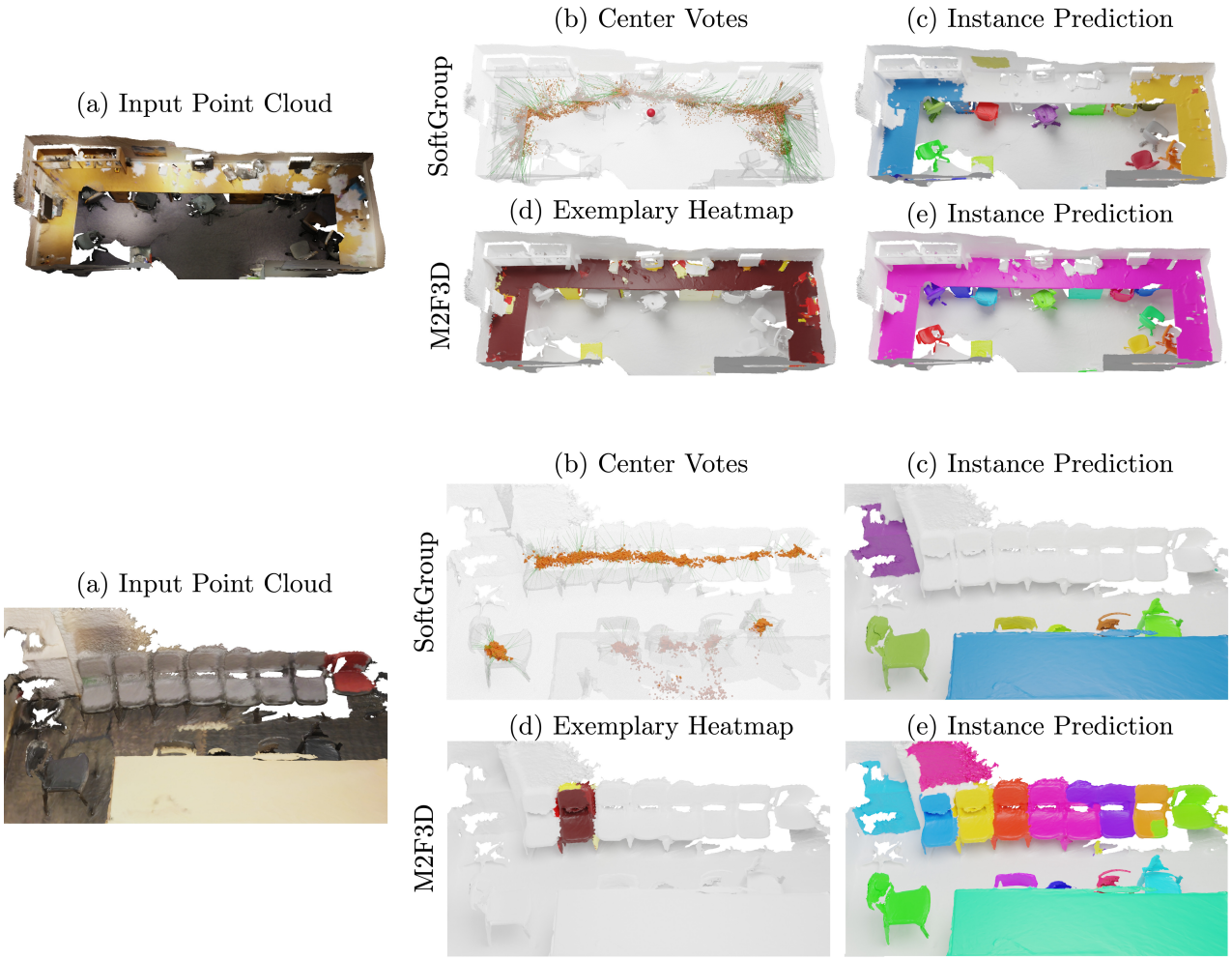


Figure 5. **Qualitative Comparison to SoftGroup [32].** We compare M2F3D with the current top-performing voting-based approach SoftGroup. The top example shows a scene containing a single large U-shaped table, see (e) in pink. SoftGroup is based on center-voting and tries to predict the instance center, shown in (b) in red. However, predicting centers of such very large non-convex shapes can be difficult for voting-based approaches. Indeed, SoftGroup fails to correctly segment the table and returns two partial instances (c). Our M2F3D, on the other side, does not rely on hand-selected geometric properties such as centers and can handle arbitrarily shaped and sized objects. It correctly predicts the tables instance mask (e). In the bottom example, we see that SoftGroup has difficulties to predict precise centers for multiple chairs located next to each other (b). As a result, the manually tuned grouping mechanism aggregates them all into one big instance which is later discarded by the refinement step. It therefore misses to segment all eight chairs (c). M2F3D does not rely on hand-crafted grouping mechanisms and can successfully segment most of the chairs.

Comparison to SoftGroup In Fig. 5, we qualitatively compare M2F3D with SoftGroup [32], the best performing voting-based 3D instance segmentation approach.

Model Details. We deploy a Minkowski Res16UNet34C [4] and obtain feature maps from all of its 5 scales. The feature maps have $(96, 96, 128, 256, 256)$ channels (sorted from fine to coarse). As the Transformer decoder expects a feature dimension of 128, we apply a non-shared linear projection after each $\mathbf{F}_i$ to map the features to the expected dimension. Furthermore, we employ a modified Transformer decoder by Mask2Former [3] (swapped cross- and self-attention) leveraging an 8-headed attention and a feedforward network with 1024-dimensional features. For each intermediate feature map, we instantiate a dedicated decoder layer. We attend to the backbone features 3 times with Transformer decoders with shared weights. We use 100 object queries. Following Misra *et al.* [23], we calculate the query positions from random Fourier embeddings [31] based on relative voxel positions scaled to $[-1, 1]$.