3D Instances as 1D kernels Supplementary Materials

Yizheng Wu[1], Min Shi[1], Shuaiyuan Du[1], Hao Lu[1], Zhiguo Cao[1], and Weicai Zhong[2]

[1] Key Laboratory of Image Processing and Intelligent Control, Ministry of Education
School of AIA, Huazhong University of Science and Technology, China
[2] Huawei CBG Consumer Cloud Service Search & Maps BU
{yzwu21,min_shi,sydu,hlu,zgcao}@hust.edu.cn zhongweicai@huawei.com

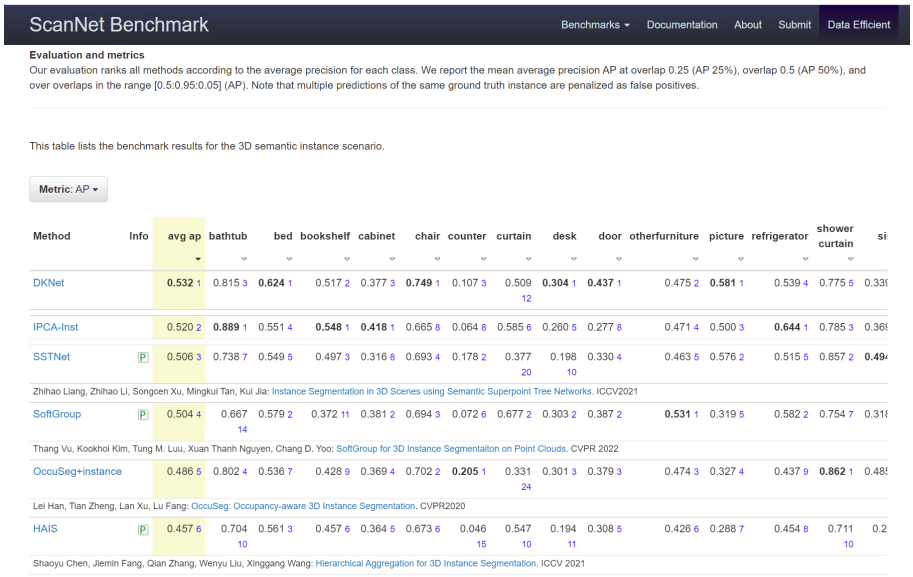

Fig. S1. Our DKNet ranks first on the mAP leaderboard of the ScanNetV2 benchmark. The snapshot is taken on 7 March 2022.

S1 List of Content

This supplementary material consists of the following contents:

- More details on the backbone and training loss functions.
- Detailed algorithm for candidate mining and describing.
- Detailed algorithm for duplicated candidates merging.
- More details on the components of instance kernels.
- More details on transforming instance kernels into the weights of dynamic convolutions layers.

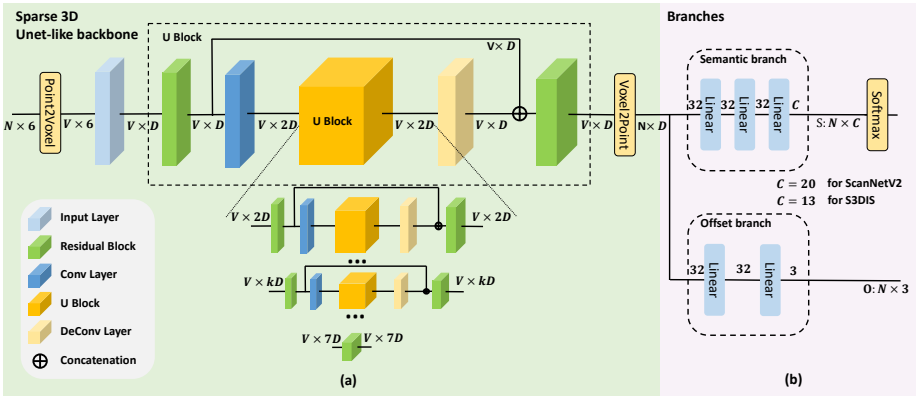

Fig. S2. Detailed network architecture. (a) Architecture of backbone. (b) Architecture of semantic branch and offset branch. V denotes the number of voxels. The number near linear layer denotes the number of output channels.

- More details on computing instance mask thresholds.
- More visualizations of the instance segmentation results.
- More details on the efficiency of DKNet.
- More detailed results on ScanNetV2 [2] dataset.

S2 Backbone and Loss Function

As mentioned in Sec. 3.2, a 3D UNet-like backbone [12] is adopted to extract point features $F_p \in \mathbb{R}^{N \times D}$. And two multi-layer perceptrons (MLPs) are used to predict semantic masks and centroid offsets. We specify the detailed architectures of these three components in Fig. S2. These three components are adopted from PointGroup [6], which is common for recent top-performing methods [1,8].

Loss for Semantic Branch. The semantic prediction branch outputs $S \in \mathbb{R}^{N \times C}$, where C is the number of categories. For point P_i, S_i denotes the probability of this point belonging to different semantic categories. Given the one-hot ground truth semantic label $\hat{S}_i$, the semantic loss $\mathcal{L}_{sem}$ can be computed as:

$$\mathcal{L}_{sem} = \frac{1}{N}\sum_{i=1}^{N} CE(S_i, \hat{S}_i) + 1 - \frac{2\sum_{i=1}^{N} S_i^T \hat{S}_i}{\sum_{i=1}^{N} S_i^T S_i + \sum_{i=1}^{N} \hat{S}_i^T \hat{S}_i}, \tag{S1}$$

where $CE(x, y)$ denotes the cross entropy loss. The second term in Eq. S1 is the multi-class dice loss [10], which can help address the imbalance between different semantic categories.

Loss for Offset Branch. The offset branch estimates the centroid offsets for all points, *i.e.*, $O \in \mathbb{R}^{N \times 3}$. Given a point P_i, we define the centroid of the instance that covers this point as $C_{p,i}$. Both the Euclidean norm and the direction are

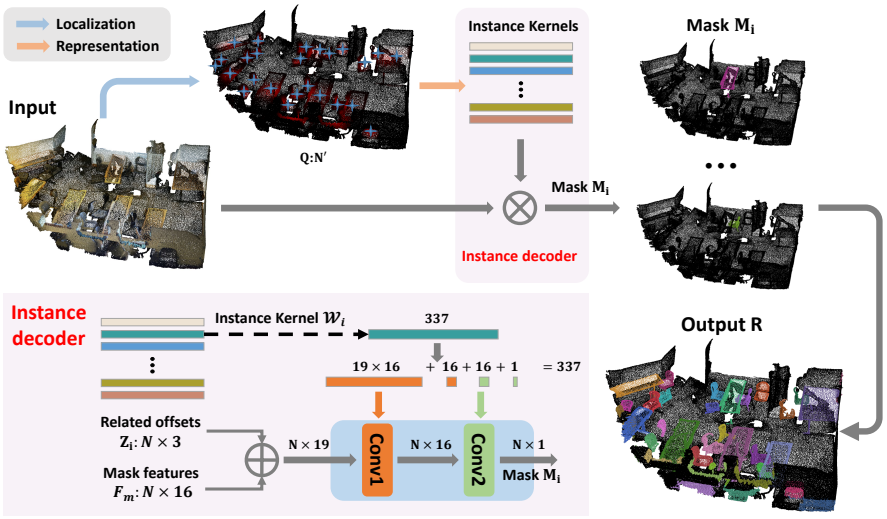

Fig. S3. Details of instance decoder. The instance decoder consists of two convolution layers. The elements in instance kernels are sequentially inserted into the weights and biases for convolution layers.

considered to measure the difference between the estimated centroid offset vector O_i and the ground truth offset $C_{p,i} - X_i$, where X_i denotes the 3D coordinate of the point P_i. Then, the offset loss $\mathcal{L}_{off}$ is computed as:

$$\mathcal{L}_{off} = \frac{1}{N'} \sum_{i=1}^{N} (\|O_i - (C_{p,i} - X_i)\| + \frac{O_i \cdot (C_{p,i} - X_i)}{\|O_i\| \cdot \|C_{p,i} - X_i\|}) \cdot \mathbb{I}(P_i), \tag{S2}$$

where $\mathbb{I}(P_i)$ is an indicator function that outputs 1 when point P_i belongs to one instance, otherwise outputs 0. N' denotes the number of points (excluding background points), which can be obtained via $N' = \sum_{i=1}^{N} \mathbb{I}(P_i)$.

S3 Algorithms

Candidate Mining Algorithm As mentioned in Sec. 3.3 (line 219), we design a customized non-maximum suppression algorithm with local normalization (LN-NMS) to localize instance centroids from the predicted heatmaps. The detailed candidate mining process is described in Algorithm 1. The semantic label $B \in \mathbb{R}^N$ denotes the hard semantic label derived from the soft semantic masks $S \in \mathbb{R}^{N \times C}$. B_i equals the category label with the maximum score in S_i. Note that, apart from localizing instance centroids, the candidate mining algorithm fetches the "foreground points" and "background points" of each candidate to describe the candidates for further processing, as mentioned in Sec.3.4 (line 244).

Algorithm 1 Candidate mining algorithm & thresholds T_θ, Q_θ, R

Input: centroids map H, coordinates X, semantic labels B,
dimension reduced features F_k.
Output: candidates $Q = \{Q_1, Q_2, ..., Q_{N'}\}$
neighbor features $F_n = \{F_{n,1}, F_{n,2}, ..., F_{n,N'}\}$
background features $F_b = \{F_{b,1}, F_{b,2}, ..., F_{b,N'}\}$

initialize an empty candidates set Q
initialize an empty neighbor features set F_n
initialize an empty background features set F_b
initialize an array f(available) of length N with all ones
initialize counter $k = 0$
while $k < T_\theta$ and $f.sum() > 0$ **do**
  initialize distance array d of length N with all zeros
  initialize neighbors feature f_n of length C with all zeros, counter $k_n = 0$
  initialize background feature f_b of length C with all zeros, counter $k_b = 0$
  /* Candidates mining */
  set $q = H.argmax()$
  set $d = \|X - X_q\|_2$, $f[d < R] = 0$
  if $H_q / H[d < R].max() < Q_\theta$ **then**
    continue
  /* Candidates describing */
  for $j \in [1, N]$ with $d_j < R$ and $B_j == B_q$ **do**
    $f_{n,k} += f_{d,j}$, k_n++
  for $j \in [1, N]$ with $d_j < 2R$ and $B_j != B_q$ **do**
    $f_{b,k} += f_{d,j}$, k_b++
  $Q.append(q)$
  $F_n.append(f_n / k_n)$
  $F_b.append(f_b / k_b)$
  $k++$
return Q, F_n, F_b

Candidate Merging Algorithm As mentioned in Sec. 3.4, to aggregate duplicated candidates with the predicted merging score map A , we design a candidates merging algorithm. Algorithm 2 illustrates this merging process in details.

S4 Instance kernels

To obtain a discriminative instance kernel, we encode semantic, positional and shape information to represent instance. The position and semantic information comes from candidate coordinates and point features. Shape information is thus encoded by splitting 'foreground points' and 'background points'. As illustrated in Fig. S4, 'foreground points' sketch a basic shape of instance (a chair). In Table S1 we show the performance when the position (coordinates) or shape (mixing foreground and background points for feature pooling) is ablated. One can observe that both information is vital for instance kernels.

Algorithm 2 Candidate merging algorithm

Input: merging score map A, centroids map H, candidates Q.
Output: instance centroid map $M_{ins} \in \mathbb{R}^{N'}$

initialize an array $M_{ins} = arange(N')$
while $A.max() > 0.5$ **do**
  $i, j =$ col, row of $A.argmax()$
  $G_i = (M_{ins} == M_{ins}[i])$, $G_j = (M_{ins} == M_{ins}[j])$, $G =$ where$(G_i \cup G_j)$
  $c = H[Q[G]].argmax()$
  update $M_{ins}[G] = M_{ins}[c]$
  for m in G **do**
    for n in G **do**
      update A[m,n] = 0
return M_{ins}

Table S1. Component analysis.

Info. components	*mAP*	*AP*@50	*AP*@25
W/o coord.	49.5	66.3	75.9
W/o shape	49.3	64.3	75.2
W/o both	47.9	63.6	73.8
Full	50.8	66.7	76.9

Table S2. Kernel shape analysis.

Kernel shape/size	mAP	AP@50	AP@25
$[8, 1]/169$	50.6	67.8	76.4
$[16, 1]/337$	50.8	66.7	76.9
$[16, 8, 1]/465$	51.2	67.1	77.0
$[16, 16, 1]/609$	50.9	66.0	76.5

S5 Dynamic Convolution

To generate instance masks, point features are fed into different instance decoders consisting of a few dynamic convolution layers. The parameters of dynamic convolutions are conditioned on the corresponding instance kernels. As shown in Fig. S3, we instantiate the instance decoder with two convolution layers, which have 16 and 1 output channels (its kernel shape is $[16, 1]$). The elements in one instance kernel are sequentially inserted into the weight vectors and biases of these two convolution layers. Hence, the length of instance kernels L depends on the specific configuration of the instance decoder. As for the instance decoder in Fig. S3, L can be computed by:

$$Conv1\ \#weight = (16 + 3) \times 1 \times 1 \times 16 = 304,\ \#bias = 16 \times 1 = 16\,, \tag{S3}$$

$$Conv2\ \#weight = 16 \times 1 \times 1 \times 1 = 16,\ \#bias = 1 \times 1 = 1\,, \tag{S4}$$

$$L = 304 + 16 + 16 + 1 = 337\,, \tag{S5}$$

To evaluate the effect of kernel size, we design a series of ablation experiments. As illustrated in Table S2, DKNet is stable under varying instance kernel sizes and dynamic convolution layer shapes.

S1 Thresholds for Soft Instance Mask

As mentioned in Section 3.5, in post-processing, we use the Otsu algorithm [11] to binarize the predicted soft instance masks. Otsu algorithm divides the pixels

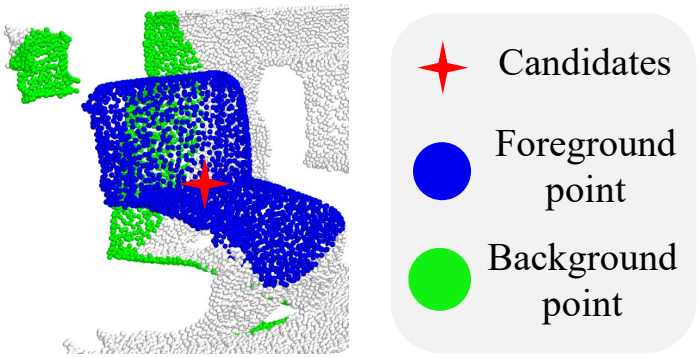

Fig. S4. Shape modeling.

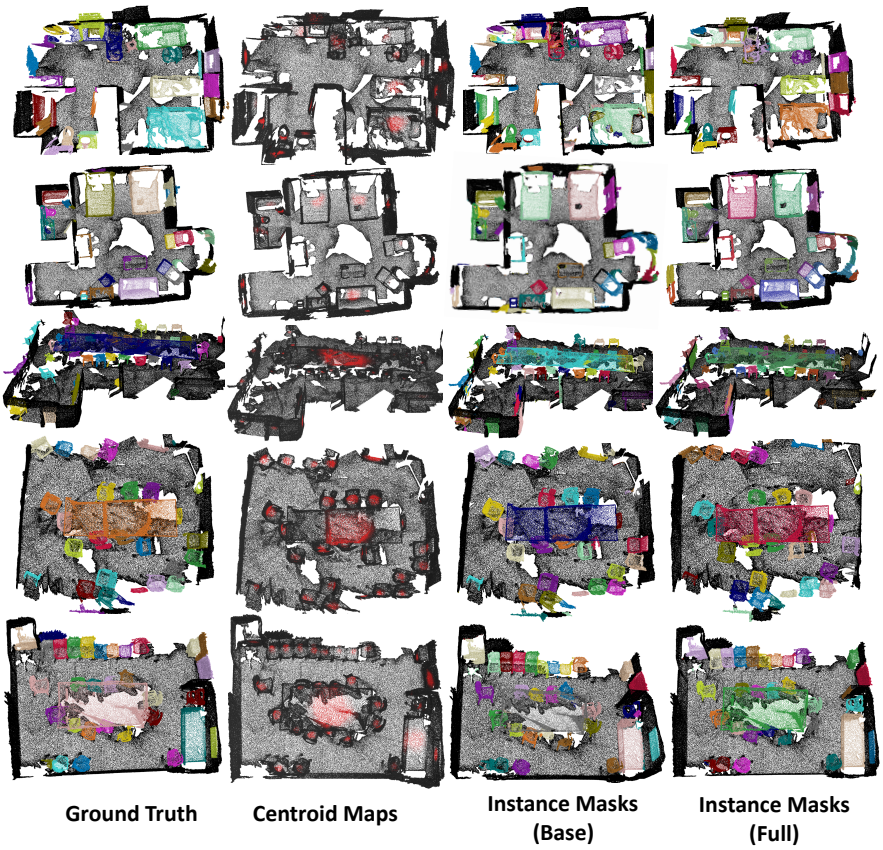

Fig. S5. Visualizations on ScanNetV2 validation set.

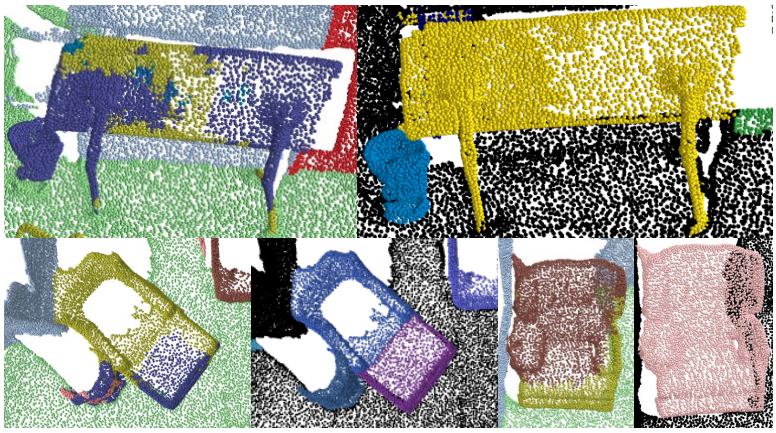

Fig. S6. The robustness against th errors in semantic predictions. Although errors occur in the semantic results (left parts), the instance decoder can still recover correct instance masks (right parts).

in a grey image into the foreground or the background category, whose essential idea is to maximize the inter-class variance. We re-purpose this idea to binarize the soft instance masks $M \in \mathbb{R}^{I \times N}$. As the original algorithm functions on pixels with discrete gray levels, similarly, we discreteize the value ($[0, 1]$) of soft instance mask into K confidence levels. Therefore, the quantified instance mask M' can be obtained by:

$$M'_k = \lfloor M_k * K \rfloor , \tag{S6}$$

where M_k denotes the k^{th} instance mask. Taking the quantified masks as inputs, Otsu algorithm processes each point in M'_k as a pixel in an image, and outputs $T_{m,k}$ for each instance. Instead of using fixed threshold, Otsu algorithm can adaptively generate thresholds, which can better preserve the shape of instance with weak responses.

S6 More Visualizations

More visualizations of instance segmentation results and intermediate centroid maps are shown in Fig. S5. **Base** denotes the baseline method without candidate aggregation while **Full** denotes our full method. We also observe that, by reconstructing instance masks from instance kernels, some errors in semantic predictions can be corrected. We show some examples in Fig S6.

S7 Efficiency

Here we specify the training and inference time of the proposed DKNet. Training DKNet on ScanNetV2 [2] with default settings consumes about 72 GPU hours

Table S3. Inference time of different stages in DKNet on RTX 3090.

Total	Backbone	Encoding	Decoding	Post-processing
357.5ms	160.6ms	129.3ms	16.3ms	45.9ms

Table S4. Full quantitative results of AP@50 on the ScanNetV2 test set. Best performance is in boldface.

approaches	AP@50	bathtub	bed	booksh.	cabinet	chair	counter	curtain	desk	door	otherfu.	picture	refrige.	s. curtain	sink	sofa	table	toilet	window
3D-BoNet[13]	48.8	**100.0**	67.2	59.0	30.1	48.4	9.8	62.0	30.6	34.1	25.9	12.5	43.4	79.6	40.2	49.9	51.3	90.9	43.9
MTML[7]	54.9	**100.0**	80.7	58.8	32.7	64.7	0.4	81.5	18.0	41.8	36.4	18.2	44.5	**100.0**	44.2	68.8	57.1	**100.0**	39.6
3D-MPA[3]	61.1	**100.0**	83.3	76.5	52.6	75.6	13.6	58.8	47.0	43.8	43.2	35.8	65.0	85.7	42.9	76.5	55.7	**100.0**	43.0
PointGroup[6]	63.6	**100.0**	76.5	62.4	50.5	79.7	11.6	69.6	38.4	44.1	55.9	47.6	59.6	**100.0**	66.6	75.6	55.6	99.7	51.3
GICN[9]	63.8	**100.0**	**89.5**	80.0	48.0	67.6	14.4	73.7	35.4	44.7	40.0	36.5	**70.0**	**100.0**	56.9	**83.6**	59.9	**100.0**	47.3
DyCo3D[5]	64.1	**100.0**	84.1	89.3	53.1	80.2	11.5	58.8	44.8	43.8	53.7	43.0	55.0	85.7	53.4	76.4	65.7	98.7	56.8
Occuseg[4]	67.2	**100.0**	75.8	68.2	57.6	84.2	**47.7**	50.4	52.4	56.7	58.5	45.1	55.7	**100.0**	75.1	79.7	56.3	**100.0**	46.7
SSTNet[8]	69.8	**100.0**	69.7	**88.8**	55.6	80.3	38.7	62.6	41.7	55.6	58.5	70.2	60.0	**100.0**	**82.4**	72.0	69.2	**100.0**	50.9
HAIS[1]	69.9	**100.0**	84.9	82.0	**67.5**	80.8	27.9	**75.7**	46.5	51.7	**59.6**	55.9	60.0	**100.0**	65.4	76.7	67.6	99.4	56.0
Ours	**71.8**	**100.0**	81.4	78.2	61.9	**87.2**	22.4	75.1	**56.9**	**67.7**	58.5	**72.4**	63.3	98.1	51.5	81.9	**73.6**	**100.0**	**61.7**

Table S5. Full quantitative results of mAP on the ScanNetV2 test set. Best performance is in boldface.

approaches	mAP	bathtub	bed	booksh.	cabinet	chair	counter	curtain	desk	door	otherfu.	picture	refrige.	s. curtain	sink	sofa	table	toilet	window
3D-BoNet[13]	25.3	51.9	32.4	25.1	13.7	34.5	3.1	41.9	6.9	16.2	13.1	5.2	20.2	33.8	14.7	30.1	30.3	65.1	17.8
MTML[7]	28.2	57.7	38.0	18.2	10.7	43.0	0.1	42.2	5.7	17.9	16.2	7.0	22.9	51.1	16.1	49.1	31.3	65.0	16.2
3D-MPA[3]	35.5	45.7	48.4	29.9	27.7	59.1	4.7	33.2	21.2	21.7	27.8	19.3	41.3	41.0	19.5	57.4	35.2	84.9	21.3
PointGroup[6]	40.7	63.9	49.6	41.5	24.3	64.5	2.1	**57.0**	11.4	21.1	35.9	21.7	42.8	66.0	25.6	56.2	34.1	86.0	29.1
GICN[9]	34.1	58.0	37.1	34.4	19.8	46.9	5.2	56.4	9.3	21.2	21.2	12.7	34.7	53.7	20.6	52.5	32.9	72.9	24.1
DyCo3D[5]	39.5	64.2	51.8	44.7	25.9	66.6	5.0	25.1	16.6	23.1	36.2	23.2	33.1	53.5	22.9	58.7	43.8	85.0	31.7
Occuseg[4]	48.6	80.2	53.6	42.8	36.9	70.2	**20.5**	33.1	30.1	37.9	47.4	32.7	43.7	**86.2**	48.5	60.1	39.4	84.6	27.3
SSTNet[8]	50.6	73.8	54.9	49.7	31.6	69.3	17.8	37.7	19.8	33.0	46.3	57.6	51.5	85.7	**49.4**	63.7	45.7	**94.3**	29.0
HAIS[1]	45.7	70.4	56.1	45.7	36.4	67.3	4.6	54.7	19.4	30.8	42.6	28.8	45.4	71.1	26.2	56.3	43.4	88.9	34.4
Ours	**53.2**	**81.5**	**62.4**	**51.7**	**37.7**	**74.9**	10.7	50.9	**30.4**	**43.7**	**47.5**	**58.1**	**53.9**	77.5	33.9	**64.0**	**50.6**	90.1	**38.5**

on an single RTX 3090. In terms of inference, DKNet is relatively efficient; the average inference time (per scene) of DKNet on a Titan XP is 521 ms, which is on par with recent bottom-up approaches PointGroup [6] (452 ms) and HAIS [1] (339 ms) on the same device, only introducing limited latency (100-200 ms). Note that, DKNet is much more efficient than recent top-down approach such as GICN [9] (8615 ms).

S8 Full Evaluation Results

The results under AP@50 and mAP metrics on ScanNetV2 benchmark are reported in Table S4 and Table S5.

References

1. Chen, S., Fang, J., Zhang, Q., Liu, W., Wang, X.: Hierarchical aggregation for 3d instance segmentation. In: Proceedings of IEEE International Conference on Computer Vision (ICCV) (2021)
2. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of IEEE Conference on Computer Vision Pattern Recognition (CVPR) (2017)
3. Engelmann, F., Bokeloh, M., Fathi, A., Leibe, B., Nießner, M.: 3d-mpa: Multi-proposal aggregation for 3d semantic instance segmentation. In: Proceedings of IEEE Conference on Computer Vision Pattern Recognition (CVPR) (2020)
4. Han, L., Zheng, T., Xu, L., Fang, L.: Occuseg: Occupancy-aware 3d instance segmentation. In: Proceedings of IEEE Conference on Computer Vision Pattern Recognition (CVPR) (2020)
5. He, T., Shen, C., van den Hengel, A.: Dyco3d: Robust instance segmentation of 3d point clouds through dynamic convolution. In: Proceedings of IEEE Conference on Computer Vision Pattern Recognition (CVPR) (2021)
6. Jiang, L., Zhao, H., Shi, S., Liu, S., Fu, C.W., Jia, J.: Pointgroup: Dual-set point grouping for 3d instance segmentation. In: Proceedings of IEEE Conference on Computer Vision Pattern Recognition (CVPR) (2020)
7. Lahoud, J., Ghanem, B., Oswald, M.R., Pollefeys, M.: 3d instance segmentation via multi-task metric learning. In: Proceedings of IEEE International Conference on Computer Vision (ICCV) (2019)
8. Liang, Z., Li, Z., Xu, S., Tan, M., Jia, K.: Instance segmentation in 3d scenes using semantic superpoint tree networks. In: Proceedings of IEEE International Conference on Computer Vision (ICCV) (2021)
9. Liu, S., Yu, S., Wu, S., Chen, H., Liu, T.: Learning gaussian instance segmentation in point clouds. arXiv Computer Research Repository (2020)
10. Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: International Conference on 3D Vision (3DV) (2016)
11. Otsu, N.: A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics (1979)
12. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proceedings of International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI). Springer (2015)
13. Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., Trigoni, N.: Learning object bounding boxes for 3d instance segmentation on point clouds. In: Proceedings of Advances in Neural Information Processing Systems (NeurIPS) (2019)