Supplementary Material for: Learning Multi-View Aggregation In the Wild for Large-Scale 3D Semantic Segmentation

Damien Robert[1, 2] damien.robert@ign.fr
Bruno Vallet[2] bruno.vallet@ign.fr
Loic Landrieu[2] loic.landrieu@ign.fr

[1]CSAI, ENGIE Lab CRIGEN, Stains, France
[2]Univ Gustave Eiffel, ENSG, IGN, LASTIG, F-77454 Marne-la-Vallee, France

SM-1. Interactive Visualization and Code

We release our code at https://github.com/drprojects/DeepViewAggand our run metrics at https://wandb.ai/damien_robert/DeepViewAgg-benchmark. The provided code allows for reproduction of our experiments and inference using pretrained models.

Our repository also contains interactive visualizations as HTML files, showing different images and model predictions for spheres sampled in S3DIS, as shown in Figure SM-1. This tool makes it easier to see the additional insights brought by images than the visuals included in the paper.

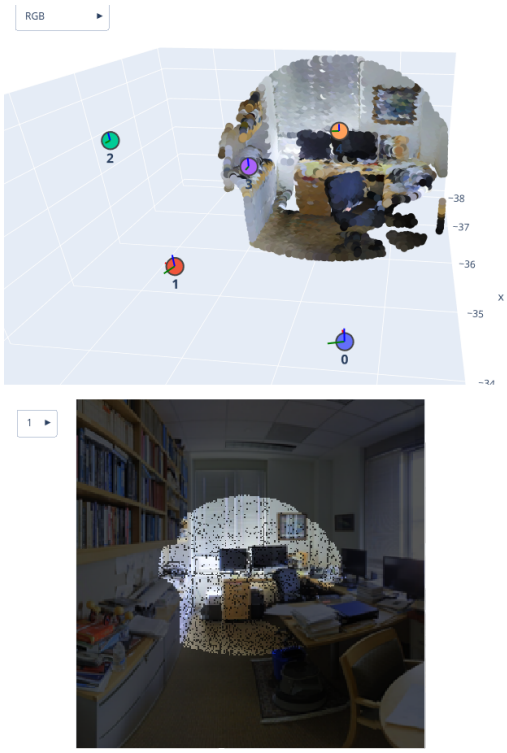


Figure SM-1. **Visualizations.** We propose interactive visualizations of hybrid 2D/3D data along with predictions of our model. We also provide the code necessary to create more such visualizations.

SM-2. Efficient Point-Pixel Mapping

In the Z-buffering step, we only consider points at a maximum distance $R = 8$m for indoor scenes and $R = 20$m for outdoor settings. We replace the points in image i by cubes oriented towards i and with a size given by the following formula involving $\text{dist}(p, i)$ the distance between point p and image i, $k = 1$ a swell factor ruling how much closer cubes are expanded and c the resolution of the voxel grid, or a typical inter-point distance (2-8cm in our experiments):

$$\text{size\_of\_cubes}(\text{dist}(p, i)) = c(1 + ke^{-\text{dist}(p,i)/R}) \,. \tag{SM-1}$$

This heuristic increases the size of cubes that are close to the image to ensure that they do *hide* the cubes behind them. Note that this heuristic operates on the size of the 3D cubes before camera projection and not on their projected pixel masks, which are computed based on camera intrinsic parameters. See Algorithm SM-1 for the pseudo-code of the mapping computation.

Storing point-pixel mappings for large-scale scenes with many images can be challenging. To minimize the memory impact of such a procedure, we use the Compressed Sparse Row (CSR) format. This allows us to represent the mappings compactly and treat large scenes at once.

SM-3. Projection Information

We use 8 handcrafted features to qualify the viewing conditions of a point p seen in image i.

Algorithm SM-1 Z-buffering-Based Point-Pixel Mapping

Input: I image set, P point cloud
for $i \in I$ **do**
  zBuffer $\leftarrow$ `maxFloat` array of size i
  indexMap $\leftarrow$ `NaN` array of size i
  $P' \leftarrow$ points of P in frustum of i and closer than R
  for $p \in P'$ **do**
    $s \leftarrow$ size_of_cubes$(\text{dist}(p, i))$
    $mask \leftarrow$ pixel mask covered by the projection of a cube of size s at p onto the image i
    for $(u, v) \in mask$ **do**
      if $\text{dist}(p, i) <$ zBuffer$[u, v]$ **then**
        zBuffer$[u, v] \leftarrow \text{dist}(p, i)$
        indexMap$[u, v] \leftarrow p$
      end if
    end for
  end for
  for $p \in P'$ **do**
    if p appears in indexMap **then**
      $\text{pix}(p, i) \leftarrow$ pixel at the projection of p on i
    else
      $\text{pix}(p, i)$ not defined, p not seen in i
    end if
  end for
end for

- **Normalized depth.** An image seeing a point at a distance may contain relevant contextual cues but poor textural information. We compute the distance $\text{dist}(p, i)$ between point p and image i and divide by the maximum viewing distance $R = 8$m for indoor scenes and $R = 20$m for outdoor scenes.

- **Local geometric descriptors.** The geometry of a point cloud can impact the quality of its views in images. Indeed, while planar surfaces may be better captured by a camera, a highly irregular surface may present many occlusions or grazing rays. We compute geometric descriptors (linearity, planarity, scattering) based on the eigenvalues of the covariance matrix between a point and its 50 neighbors [6].

- **Viewing angle.** An image seeing a surface from a right angle may better capture its surroundings than if the view angle is slanted with respect to the surface. We compute the absolute value of the cosine between the viewing angle and the normal estimated from the covariance matrix calculated at the previous step.

- **Pixel row.** To account for potential camera distortion near the top and bottom of the image (*e.g.* for equirectangular images), we report the row of pixels and divide by the image height (number of rows). Note that we could derive a similar feature for cameras with radial distortion, such as fisheye cameras.

- **Local density.** Density can impact occlusion and be an indicator of the local precision of the 3D sensor. We compute the area of the smallest disk containing the 50th neighbor and normalize it by the square of the voxel grid resolution.

- **Occlusion rate.** Occlusion may significantly impact the quality of the projected image features. We compute the ratio of the 50 nearest neighbors of p also seen in i.

SM-4. Fusion schemes

We denote by $\{f_i^{\text{2D}}\}_{i \in I}$ a set of 2D feature maps of width C associated with the images I, typically obtained with a convolutional neural network. f^{3D} designates the raw feature of the point cloud given by the sensor: position, but also intensity/reflectance if available (not used in this paper). We denote by $\mathcal{P}(f^{\text{2D}}, P)$ the projection of the learned image features f^{2D} onto the point cloud P by our multi-view aggregation technique. The early and late fusion schemes can be written as follows:

$$y_{\text{early}} = \mathcal{C}^{\text{3D}} \circ \mathcal{D}^{\text{3D}} \circ \mathcal{E}^{\text{3D}} \left(\left[f^{\text{3D}}, \mathcal{P}(f^{\text{2D}}, P) \right] \right) \quad \text{(SM-2)}$$

$$y_{\text{late}} = \mathcal{C}^{\text{3D}} \left(\left[\mathcal{D}^{\text{3D}} \circ \mathcal{E}^{\text{3D}} \left(f^{\text{3D}} \right), \mathcal{P}(f^{\text{2D}}, P) \right] \right) . \quad \text{(SM-3)}$$

For the intermediate fusion scheme, the 2D and 3D features are merged directly in the 3D encoder. Our 3D backbone follows a classic U-Net architecture, and its encoder is organized in L levels $\{\mathcal{E}_l^{\text{3D}}\}_{l=1}^{L}$ processing maps of increasingly coarse resolution. Each level $l > 1$ is composed of of a downsampling module down_l, typically strided convolutions ($\text{down}_1 = \text{Id}$), and a convolutional module conv_l, typically a sequence of ResNet blocks. The 2D encoder is also composed of L levels $\{\mathcal{E}_l^{\text{2D}}\}_{l=1}^{L}$ corresponding to the different image resolutions. We propose to match the 2D and 3D levels at full resolution (1024×512 and 2cm for S3DIS), and all subsequent levels after the same number of 2D/3D downsampling. At each level $l = 1 \dots L$, we concatenate the downsampled higher resolution 3D map f_{l-1}^{3D} with the map f_l^{2D} obtained from the images at the matched resolution:

$$f_l^{\text{3D}} = \text{conv} \left(\left[\text{down} \left(f_{l-1}^{\text{3D}} \right), \mathcal{P}(f_l^{\text{2D}}, P) \right] \right) , \quad \text{(SM-4)}$$

with f_0^{3D} the raw 3D features. The decoder and classifiers follow the same organization than the 3D backbone.

SM-5. Dynamic-Size Image-Batching

We consider P_{sample} a portion of a point cloud to add to the 3D part of a batch. In order to build the image

Algorithm SM-2 Dynamic-Size Image-Batching

Input: P_{sample} point cloud, I image set
Parameters: B budget of pixels, m border margin
$I_{\text{sample}} \leftarrow \{i \mid i \in I \text{ and } \exists p \in P_{\text{sample}} \text{ s.t } i \in v(p)\}$
scores $\leftarrow$ array of 0 of size I_{sample}
for $i \in I_{\text{sample}}$ **do**
  $i \leftarrow$ tightest crop(i) to contain P_{sample} with margin m
  scores$[i] \leftarrow \text{score}(i, P_{\text{sample}}, I_{\text{sample}}, \varnothing)$
end for
batch $\leftarrow$ []
while $B > 0$ and length $I_{\text{sample}} > 0$ **do**
  pick $i \in I_{\text{sample}}$ randomly w.r.t. scores
  batch $\leftarrow$ [batch, i]
  $I_{\text{sample}} \leftarrow I_{\text{sample}} \setminus \{i\}$
  $B \leftarrow B - \text{area}(i)$
  for $i \in I_{\text{sample}}$ **do**
    scores$[i] \leftarrow \text{score}(i, P_{\text{sample}}, I_{\text{sample}}, \text{batch})$
  end for
end while

batch we iteratively select images according to the following procedure. We first select the image set I_{sample} seeing at least one point of P_{sample}. For equirectangular images, we rotate the images to place the mappings at the center. We then crop each image i along the tightest bounding box containing all seen point of P_{sample} with a minimum margin of m along a fixed set of image size along: crops $= \{64 \times 64, 128 \times 64, 128 \times 128, 256 \times 128, 256 \times 256, 512 \times 256, 512 \times 512, 1024 \times 512\}$. We then associate with each cropped image a score defined as follows:

$$\text{score}(i, P_{\text{sample}}, I_{\text{sample}}, \text{batch})) = \tag{SM-5}$$

$$\frac{\text{area}(i)}{\max(\text{area}(i) \mid i \in I_{\text{sample}})} + \tag{SM-6}$$

$$\lambda \frac{\text{unseen}(i, P_{\text{sample}}, \text{batch}))}{\max(\text{unseen}(i, P_{\text{sample}}, \text{batch}) \mid i \in I_{\text{sample}})}, \tag{SM-7}$$

with $\text{area}(i)$ the area of the cropped image i in pixels, batch the current image batch, $\text{unseen}(i, P_{\text{sample}}, \text{batch})$ the number of points of P_{sample} seen in image i but not in any image of the current batch, and $\lambda = 2$ a parameter controlling the trade-off between maximum area and maximum coverage. The current image batch is initialized as an empty set, but the scores must be updated as it is filled. The images are chosen randomly with a probability proportional to their score. We chose in all experiments a margin $m = 8$ pixels and a budget corresponding to 4 full resolution mapping.

SM-6. Implementation Details

Our method is developed in Pytorch and is implemented within the open-source framework Torch-Points3d [2].

For our backbone 3D network, we use Torch-Point3D's [2] Res16UNet34 implementation of MinkowskiNet [3]. This UNet-like architecture comprises 5 encoding layers and 5 decoding layers. Encoding layers are composed of a strided convolution of `kernel_size=[3, 2, 2, 2, 2]` and `stride=[1, 2, 2, 2, 2]` followed by `N=[0, 2, 3, 4, 6]` ResNet blocks [9] of channel size `[128, 32, 64, 128, 256]`, respectively. The decoding layers are built in the same manner, with a strided transposed convolution of `kernel_size=[2, 2, 2, 2, 3]` and `stride=[2, 2, 2, 2, 1]` as their first operation, followed by a concatenation with the corresponding skipped-features from the encoder and `N=1` ResNet block of channel size `[128, 128, 96, 96, 96]`, respectively. A fully connected linear layer followed by a softmax converts the last features into class scores. Unless specified otherwise, ReLU activation and batch normalization [11] are used across the architecture.

For our 2D encoders, we use the encoder part of pre-trained ResNet18 networks [8]. For indoor scenes, we use the modified ResNet18 from [5] pre-trained on ADE20K [16], which has 5 layers of output channel sizes `[128, 64, 128, 256, 512]` and resolution `scale=[4, 4, 8, 8, 8]`. The relatively high resolution of the output feature map allows us to map the 2D features of size $C = 512$ from the last layer directly to the point cloud without upsampling. For outdoor scenes, we use the ResNet18 from [14] pre-trained on Cityscapes [4], which has 5 layers of output channel sizes `[128, 64, 128, 256, 512]` and resolution `scale=[4, 4, 8, 16, 32]`. We use pyramid feature pooling [15] on layers 1, 2, 3, and 4, which results in a feature vector of size $C = 960$ passed to the mapped points.

For our DeepViewAgg module, the extracted image features are converted into view features of size C with the MLP ϕ_0 of size $C \mapsto C \mapsto C$. For the computation of quality scores, we use the following MLPs: $\phi_1 : 8 \mapsto M \mapsto M$, $\phi_2 : M \mapsto M \mapsto M$, and $\phi_3 : 2M \mapsto M \mapsto K$ with $M = 32$ and $K = 4$.

For indoor scenes, we use the lowest resolution map of a network [5] pretrained on ADE20K [16]. For the outdoor scenes, we use pyramid feature pooling [15] with a network [14] pretrained on Cityscapes [4]. We use early fusion in all experiments unless specified otherwise.

All models are trained with SGD with an initial learning rate of 0.1 for 200 epochs with decay steps of 0.3 at epoch $80, 120, 160$ for indoor datasets, and 20 epochs with decay at epoch $10, 16$ for the outdoor dataset. The pre-trained 2D networks use a learning rate 100 times smaller than the rest of the model. We use random rotation and jittering for point

clouds, random horizontal flip and color jittering for images, and featurewise jittering for descriptors of the viewing conditions.

For more details on the implementation, we refer the reader to our provided code.

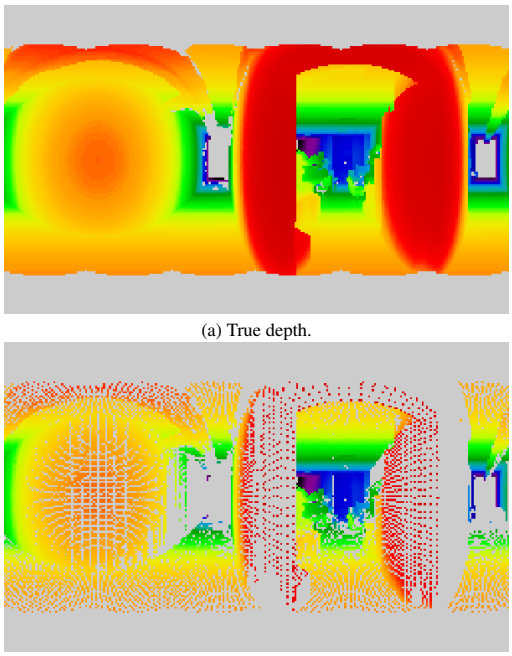
SM-7. Supplementary Ablation

We propose further ablations whose results in Table SM-1. To assess the quality of our visibility model, we propose to compute the point-pixel mappings using the depth maps provided with S3DIS instead of Z-buffering. When running our method with such mappings (Mapping from Depth), we observe lower performances. This can be explained by the fact that depth-based mapping computation is sensitive to minor discrepancies between the depth map and the real point positions. Such a phenomenon can be observed on S3DIS depth images where the surfaces are viewed from a slanted angle, resulting in fewer point-image mappings being recovered. See Figure SM-2 for an illustration of this phenomenon. In conclusion, not only can our approach bypass the need for specialized sensors or costly mesh reconstruction altogether, but our direct point-pixel mapping may yield better results than the provided mappings obtained with more involved methods.

Table SM-1. **Supplementary Ablation Study.** Mean IoU comparison of different design choices on Fold 2 and Fold 5 of S3DIS down-sampled at 5cm for processing.

Model	Fold 2	Fold 5
Best Configuration	63.1	67.5
Design Choices		
Mapping from Depth	-5.4	-1.9
Intermediate	-7.6	-3.2
XYZRGB + DeepViewAgg	-0.5	-0.9

To compare our fusion schemes, we evaluate a model with the intermediate fusion scheme described in SM-4 (Intermediate). We observe that, for our module, intermediate fusion does not perform as well as early and late fusion. This could indicate that fusing modalities at their highest respective resolutions yields better results and that matching the encoder levels of 2D and 3D networks may not be straightforward. To ensure that our proposed module captures all radiometric information contained in colorized point clouds, we trained our chosen architecture to run on colorized point clouds and images (XYZRGB + DeepViewAgg). The resulting performance confirms that colorizing 3D points does not bring additional information not already captured by images.



(a) True depth.

(b) Depth-based visibility.

Figure SM-2. **Depth-Based Mapping Computation.** Based on an input depth map (a), we compute the point-image mappings (b) by searching points within a small margin of the target depth. We note that slight depth discrepancies near slanted surfaces prevents mapping from being recovered. Better seen on a monitor.

SM-8. Influence of Maximum Depth

The maximum point-image depth is chosen as the distance beyond which adjacent 3D points appear in the same image pixel: 8m for S3DIS sampled at 5cm with images of width 1024 and 20m for KITTI-360. As illustrated in Table SM-2, reducing this parameter too much leads to a drop in performance both on S3DIS Fold 5 and KITTI-360, while slight modifications do not significantly affect the results. Since the number of point-image mappings grows quadratically with this parameter, one may consider smaller values to decrease the memory usage or computing time.

Table SM-2. **Effect of maximum Depth** We report the drop in mIoU when removing mappings beyond a threshold distance.

S3DIS FOLD 5

Max depth	8	7	6	5	4	3	2
mIoU drop	0.0	0.0	0.0	0.4	0.6	1.9	8.6

KITTI-360 Val

Max depth	20	15	10
mIoU drop	0.0	0.5	2.6

SM-9. Influence of Number of Images

In contrast to existing methods (*e.g.* MVPNet [12], VMVF [13], BPNet [10]), our set-based, sparse implementation of point-image mappings allows us to have a varying number of views per 3D point. In Table SM-3 we investigate the performance drop w.r.t. our best model when limiting the number of images per point cloud to a fixed number of images and not using dynamic batching. Under these conditions, our performance decreases by 6.8 pts on S3DIS Fold 5 and 3.5 pts on KITTI-360 when using only 3 images, *i.e.* the configuration of BPNet. For comparison, 3D points of S3DIS are seen in 5.0 images on average (STD 3.3), and 2.5 for KITTI-360 (STD 2.1).

Table SM-3. Impact of the Number of Images. We report the drop in mIoU when limiting the number of images per point cloud.

# images	8	7	6	5	4	3
S3DIS Fold 5	0.5	1.2	1.7	2.1	3.5	6.8
KITTI-360	0.5	0.1	0.5	1.3	1.9	3.5

SM-10. Influence of Viewing Conditions

We propose to highlight the role viewing conditions descriptor. In Table SM-4, we estimate the usage by our model of each feature as the drop in mIoU on S3DIS Fold 5 & KITTI-360 when they are replaced by their dataset average (*e.g.* all points appear at the same distance). We also measure the feature sensitivity by averaging the squared partial derivative [7, 3.3.1] of the view compatibility score x defined in (4) w.r.t. each view descriptor. We observe that our model makes use of all observation features, and that the compatibility scores are most sensitive to small differences in scattering for S3DIS Fold 5, and depth for KITTI-360 Val.

Table SM-4. **Usage and Sensitivity of Viewing Conditions.** Feature usage is reported as a drop in mIoU, and the sensitivity is given as the proportion of squared partial derivative of the compatibility across all features.

view feature	usage (mIoU drop)		sensitivity (in %)	
	S3DIS	KITTI-360	S3DIS	KITTI-360
depth	1.1	1.7	12.6	46.0
linearity	1.0	0.8	11.9	0.7
planarity	1.0	1.4	15.8	1.9
scattering	0.7	1.0	52.7	0.7
viewing angle	1.3	1.2	2.8	7.4
pixel row	1.1	0.8	1.6	33.2
local density	1.2	1.3	0.6	1.8
occlusion	0.7	0.9	2.0	8.2

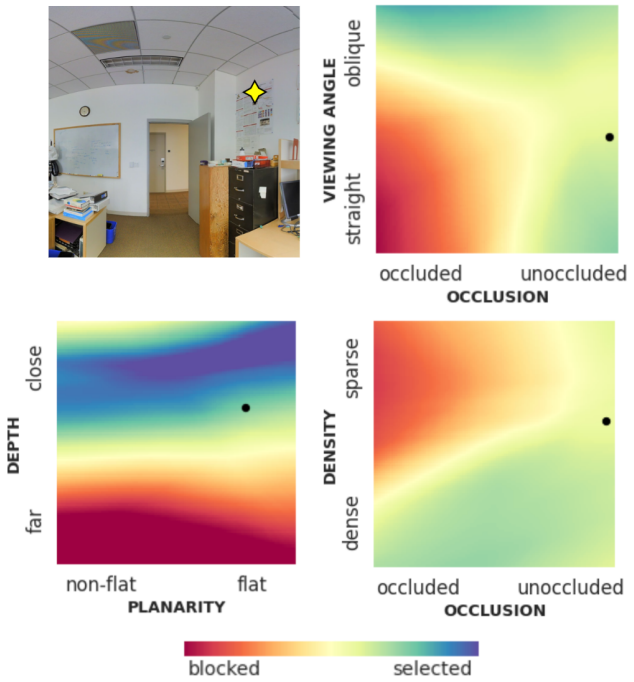


Figure SM-3. **Influence of Viewing Descriptors.** Given a point-image pair ✧ (top left), we compute the quality scores when varying two of its viewing conditions from their initial values •. For simplicity, we omit the influence of other images. We observe feature blocks specializing in retrieving information from views at a given depth range and containing planar objects (bottom left) or blocking straight yet occluded (top right) or sparse and occluded (bottom right) views.

To visualize the influence of viewing conditions, we represent in Figure SM-3 quality score heatmaps when varying pairs of features for a given view point from S3DIS.

SM-11. S3DIS Adjustment

We adjusted some room and image positions in S3DIS [1] and `https://github.com/alexsax/2D-3D-Semantics` to recover mappings between points and equirectangular images. More specifically, we rotate `Area_2/hallway_11` and `Area_5/hallway_6` by 180° around the Z-axis, and we shift and rotate all images in `Area_5b` by the same manually-found corrective offset and angle. These fixes are all available in our repository.

SM-12. Detailed Results

We report in Table SM-5 the classwise performance across all datasets. We see a clear improvement for indoor datasets for classes such as windows, boards, and pictures. These are expected results because these classes are hard to parse in 3D but easily identified in 2D. Besides, we can see that S3DIS's classes such as beams, columns,

Table SM-5. **Classwise Performance.** Classwise mIoU across all datasets for our 3D backbone network with and without learned multi-view aggregation.

S3DIS FOLD 5

Method	Avg	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
3D Backbone	64.7	**88.6**	**97.5**	82.1	**0.0**	16.6	53.8	66.2	**89.1**	78.2	73.4	**66.0**	69.6	**59.2**
+ DeepViewAgg	**67.2**	87.2	97.3	**84.3**	**0.0**	**23.4**	**67.6**	**72.6**	87.8	**81.0**	**76.4**	54.9	**82.4**	58.7

S3DIS 6-FOLD

3D Backbone	69.5	**91.2**	90.6	83.0	59.8	52.3	63.2	75.7	63.2	64.0	69.0	**72.1**	60.1	59.2
+ DeepViewAgg	**74.7**	90.0	**96.1**	**85.1**	**66.9**	**56.3**	**71.9**	**78.9**	**79.7**	**73.9**	**69.4**	61.1	**75.0**	**65.9**

ScanNet

Method	Avg	wall	floor	cabinet	bed	chair	sofa	table	door	window	bookshelf	picture	counter	desk	curtain	refrigerator	shower cur.	toilet	sink	bathtub	other
3D Backbone	69.0	82.7	**94.5**	**62.2**	77.9	88.9	**80.7**	70.3	61.4	60.2	**80.0**	28.4	60.3	**60.8**	70.4	46.2	67.4	**89.9**	62.6	**85.3**	51.0
+ DeepViewAgg	**71.0**	**84.3**	94.4	57.8	**78.9**	**90.1**	78.0	**71.9**	**63.4**	**66.9**	77.8	**39.9**	**62.2**	55.6	**71.9**	**57.4**	**71.4**	89.5	**66.2**	82.8	**56.3**

KITTI-360 Val

Method	Avg	road	sidewalk	building	wall	fence	pole	traffic lig.	traffic sig.	vegetation	terrain	person	car	truck	motorcycle	bicycle
3D Backbone	54.2	90.6	74.4	84.5	45.3	42.9	52.7	0.5	38.6	87.6	70.3	26.9	87.3	**66.0**	28.2	17.2
+ DeepViewAgg	**57.8**	**93.5**	**77.5**	**89.3**	**53.5**	**47.1**	**55.6**	**18.0**	**44.5**	**91.8**	**71.8**	**40.2**	**87.8**	30.8	**39.6**	**26.1**

chairs, and tables also benefit from the contextual information provided by images. For the KITTI-360 dataset, the multimodal model outperforms the 3D-only baseline for all classes. We can see the benefit of image features on small objects or underrepresented classes in 3D, such as traffic signs, persons, trucks, motorcycles, and bicycles.

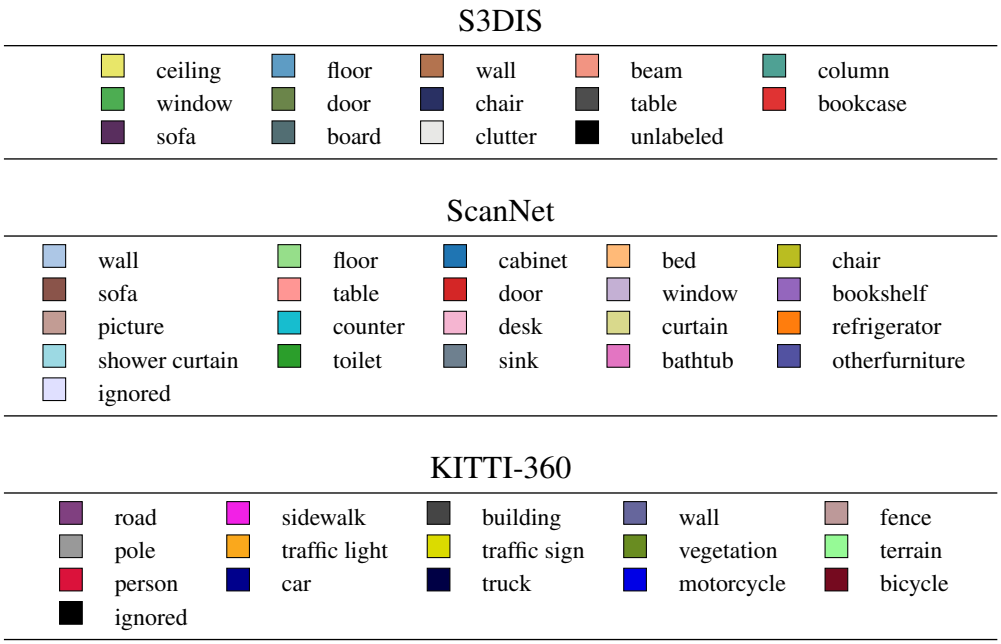


Figure SM-4. **Color Legend.**

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *CVPR*, 2016. 5

[2] Thomas Chaton, Nicolas Chaulet, Sofiane Horache, and Loic Landrieu. Torch-points3D: A modular multi-task frameworkfor reproducible deep learning on 3D point clouds. *3DV*, 2020. 3

[3] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019. 3

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 3

[5] CSAILVision. semantic-segmentation-pytorch. `github.com/CSAILVision/semantic-segmentation-pytorch`, 2020. 3

[6] Jérôme Demantké, Clément Mallet, Nicolas David, and Bruno Vallet. Dimensionality based scale selection in 3D LiDAR point clouds. In *Laserscanning*, 2011. 2

[7] Muriel Gevrey, Ioannis Dimopoulos, and Sovan Lek. Review and comparison of methods to study the contribution of variables in artificial neural network models. *Ecological Modelling*, 2003. 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 3

[10] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia, and Tien-Tsin Wong. Bidirectional projection network for cross dimension scene understanding. In *CVPR*, 2021. 5

[11] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 3

[12] Maximilian Jaritz, Jiayuan Gu, and Hao Su. Multi-view pointnet for 3D scene understanding. In *CVPR Workshops*, 2019. 5

[13] Abhijit Kundu, Xiaoqi Yin, Alireza Fathi, David Ross, Brian Brewington, Thomas Funkhouser, and Caroline Pantofaru. Virtual multi-view fusion for 3D semantic segmentation. In *ECCV*, 2020. 5

[14] Xiangtai Lee. Sfsegnets. `https://github.com/lxtGH/SFSegNets`, 2021. 3

[15] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 3

[16] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 3