RangeViT: Towards Vision Transformers for 3D Semantic Segmentation in Autonomous Driving — Supplementary Material —

Angelika Ando[1,2,*], Spyros Gidaris[1], Andrei Bursuc[1], Gilles Puy[1], Alexandre Boulch[1], Renaud Marlet[1,3]

[1]Valeo.ai, Paris, France [2]Centre for Robotics, Mines Paris, Université PSL, Paris, France
[3]LIGM, Ecole des Ponts, Univ. Gustave Eiffel, CNRS, Marne-la-Vallée, France

Contents

A. Additional visualizations

Figs. 6 and 7 show visualizations of the segmentation accuracy of RangeViT and Fig. 8 shows instances of correct and incorrect predictions. The visualizations are made on nuScenes validation point clouds. More information is provided in the captions of these figures.

B. Model parameter count analysis

In Tab. 1 of the main paper, we studied the impact of the non-linear convolutional stem and the UpConv decoder. In Tab. 9, we complete the results of Tab. 1 with a model (e) which, like model (a), has a linear stem and a linear decoder but for which the ViT backbone contains $L = 14$ transformer layers[1] instead of $L = 12$. Hence, this additional model (e) and our full RangeViT solution (model (d)) have a similar number of parameters. This experiment shows that the significant performance improvement of our full solution (d) is not simply due to a higher number of parameters, since model (e) performs much worse than our RangeViT (d). Besides, our full RangeViT solution with $D_\mathrm{h} = 64$ (model (f)) also reaches a significantly better mIoU than models (a), (b) and (e) while having nearly the same number of parameters as model (a). This confirms that the proposed convolutional stem and UpConv decoder play an important role in the performance improvement.

[*]This project was done during an internship at Valeo.ai.

[1]Note that, as pre-trained weights for the additional 2 transformer layers of model (e), which were placed on top of the existing 12 transformer layers of ViT-S, we used the pre-trained weights from the last available transformer layer.

	Stem	Decoder	Refiner	L	D_h	mIoU	#Params
(a)	Linear	Linear		12	N/A	65.52	22.0M
(b)	Conv	Linear		12	192	69.82	22.8M
(c)	Conv	UpConv		12	192	73.83	24.6M
(d)	Conv	UpConv	✓	12	192	**74.60**	25.2M
(e)	Linear	Linear		14	192	65.52	25.6M
(f)	Conv	UpConv	✓	12	64	74.00	22.7M

Table 9. **Model ablations.** Results on the nuScenes validation set. The linear stem refers to the linear patch embedding layer. When the 3D refiner layer (Refiner column) is not used, we use the K-NN post-processing technique [3].

Method	#Params	Inference time
SalsaNext [1]	6.7M	28 ms
KPRNet [2]	213.2M	-
Cylinder3D [8]	55.9M	49 ms
RangeViT (ours)	27.1M	25 ms

Table 10. **Parameter count and inference time.** Results on the nuScenes validation set.

C. Computation cost comparison

In Tab. 10, we compare the number of parameters and the inference time of RangeViT with other LiDAR segmentation methods. RangeViT has 27.1M parameters, which is four times more than SalsaNext [1] (6.73M), half of Cylinder3D [8] (55.9M) and eight times less than KPRNet [2] (213.2M). The inference time on nuScenes using the same GeForce RTX 2080 GPU is: 25 ms for RangeViT, 28 ms for SalsaNext with K-NN post-processing (15 ms without it), and 49 ms for Cylinder3D (using the simplified and faster re-implementation of [4]).

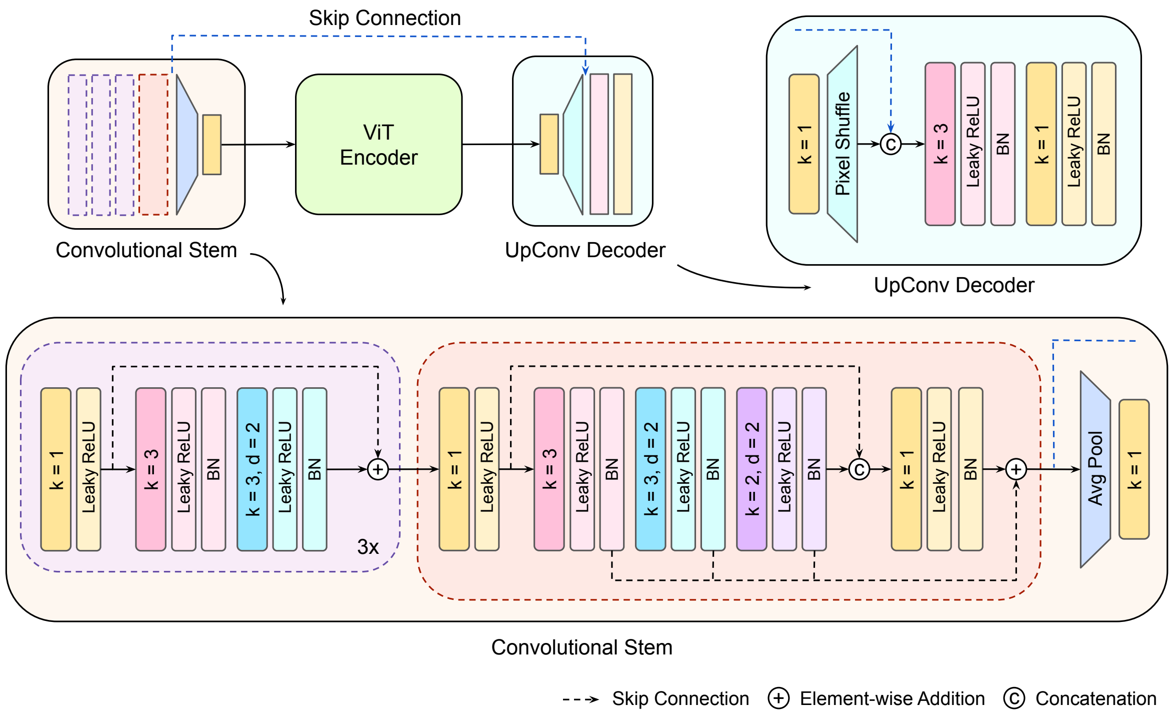

Figure 5. **Convolutional stem and UpConv decoder architecture.** In the upper left corner, there is an overview of the RangeViT architecture from the convolutional stem until the UpConv decoder. The stem and the decoder are also shown in more details. A convolution with kernel size (s, s) is denoted by $k = s$ and the dilation is denoted by d, where it is applied. Note that all the rectangles in the figure with k inside them are convolutional layers. The batch normalisation layers are denoted by BN.

Crop size	32×256	32×384	32×512
mIoU	74.40	**75.21**	74.51

Table 11. **Impact of crop size.** Effect of the training crop size on the nuScenes validation set with Cityscapes pre-training.

D. Additional ablation analysis

Impact of crop size. During training, we take a fixed-sized random crop from the range image, which is the input of the model. This design choice avoids computing self-attention on the whole range image in the ViT encoder, but it lacks the global information carried by the whole image. Nevertheless, our method is still successful for the semantic segmentation task since the $H \times 384$ window crop covers the whole vertical field-of-view (FOV) and one fifth of the horizontal (azimuthal) FOV (67.5 degrees). This is what a single nuScenes camera captures and where objects are already well identifiable. Moreover, we recall that sliding windows are also successfully used for 2D semantic segmentation with ViTs, e.g., in Segmenter [5].

In Tab. 11, we experiment with different crop sizes. As we see, the crop size has a small impact on the performance. It is also likely that tuning the learning rate and the number of epochs for these new crop sizes will reduce these small gaps even further.

Role of the classification token. The classification token interacts with the patch embeddings in the ViT encoder, but it is removed from the encoder output. As we do not use it directly for the semantic segmentation task, we explored its role by omitting it completely from the pipeline. Thus omitting the class token makes the mIoU drop from 75.21% to 74.64% and from 72.37% to 72.24% for the Cityscapes pre-training and no pre-training (random initialization), respectively. We hypothesize that the larger mIoU drop for Cityscapes pre-training is because, during 2D segmentation pre-training, the class token learned to carry global information that the patch tokens exploit via self-attention in order to extract better features for the segmentation task. In any case, the differences are small and thus feature extraction can still be achieved without adding the class token.

E. Additional implementation details

Convolutional stem and UpConv decoder. Fig. 5 shows the detailed architecture of the convolutional stem and the UpConv decoder. All convolutions that are applied before the average pooling layer in the convolutional stem or after the Pixel Shuffle layer in the decoder do not change the spatial dimensions of the feature map $H \times W$, so appropriate paddings were applied where necessary.

As described in Sec. 3 of the main paper, the average pooling layer reduces the spatial dimensions from $H \times W$ to $(H/P_H) \times (W/P_W)$. To achieve this, we use kernel size $(P_H+1) \times (P_W+1)$, kernel stride $P_H \times P_W$ and padding $(P_H/2) \times (P_W/2)$.

In the stem, the first three residual blocks use 32 feature channels and the change of dimensions from $C = 5$ input channels to 32 channels happens in the first convolutional layer of the first residual block. The fourth residual block in the stem uses D_h feature channels and similarly the change of dimensions from 32 input channels to D_h channels happens again in the first convolutional layer. Finally, the last convolutional layer in the stem (that is after the average pooling layer) uses D output feature channels.

KPConv layer of the 3D Refiner. The KPConv [6] layer of the 3D Refine has D_h input and output feature channels. Its kernel size is 15 points and the influence radius of each kernel point is 1.2.

Replacing ViT-S with ResNet-50. In Tab. 6 of the main paper, we report the results that we achieve with our model when we replace the ViT-S encoder backbone with a ResNet-50 (RN50) backbone. To implement this RN50-based model, instead of the ViT-S we use the four residual blocks of RN50, to which we introduced dilations to maintain a constant spatial resolution as in ViTs. The stem and decoder remain the same, except for the number of output channels of the stem (64) and the input channels of the decoder (2048) to make them compatible with RN50. The resulting model has comparable FLOPs (the inference time for both is 25 ms) but much more parameters (25.2M vs 35.3M).

Range-projection for the SemanticKITTI experiments. To generate the 2D range images for the SemanticKITTI experiments, instead of using the spherical projection described by Eq. (1) of the main paper, we follow [2, 7] and unfold the LiDAR scans according to the order in which they are captured by the sensor.

References

[1] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds. In *ISVC*, 2020. 1

[2] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. KPRNet: Improving projection-based LiDAR semantic segmentation. In *ECCV*, 2020. 1, 3

[3] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. RangeNet++: Fast and accurate LiDAR semantic segmentation. In *IROS*, 2019. 1

[4] Corentin Sautier, Gilles Puy, Spyros Gidaris, Alexandre Boulch, Andrei Bursuc, and Renaud Marlet. Image-to-LiDAR self-supervised distillation for autonomous driving data. In *CVPR*, 2022. 1

[5] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *ICCV*, 2021. 2

[6] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *CVPR*, 2019. 3

[7] Larissa T Triess, David Peter, Christoph B Rist, and J Marius Zöllner. Scan-based semantic segmentation of lidar point clouds: An experimental study. In *IV*, 2020. 3

[8] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Wei Li, Yuexin Ma, Hongsheng Li, Ruigang Yang, and Dahua Lin. Cylindrical and asymmetrical 3D convolution networks for LiDAR-based perception. In *CVPR*, 2021. 1

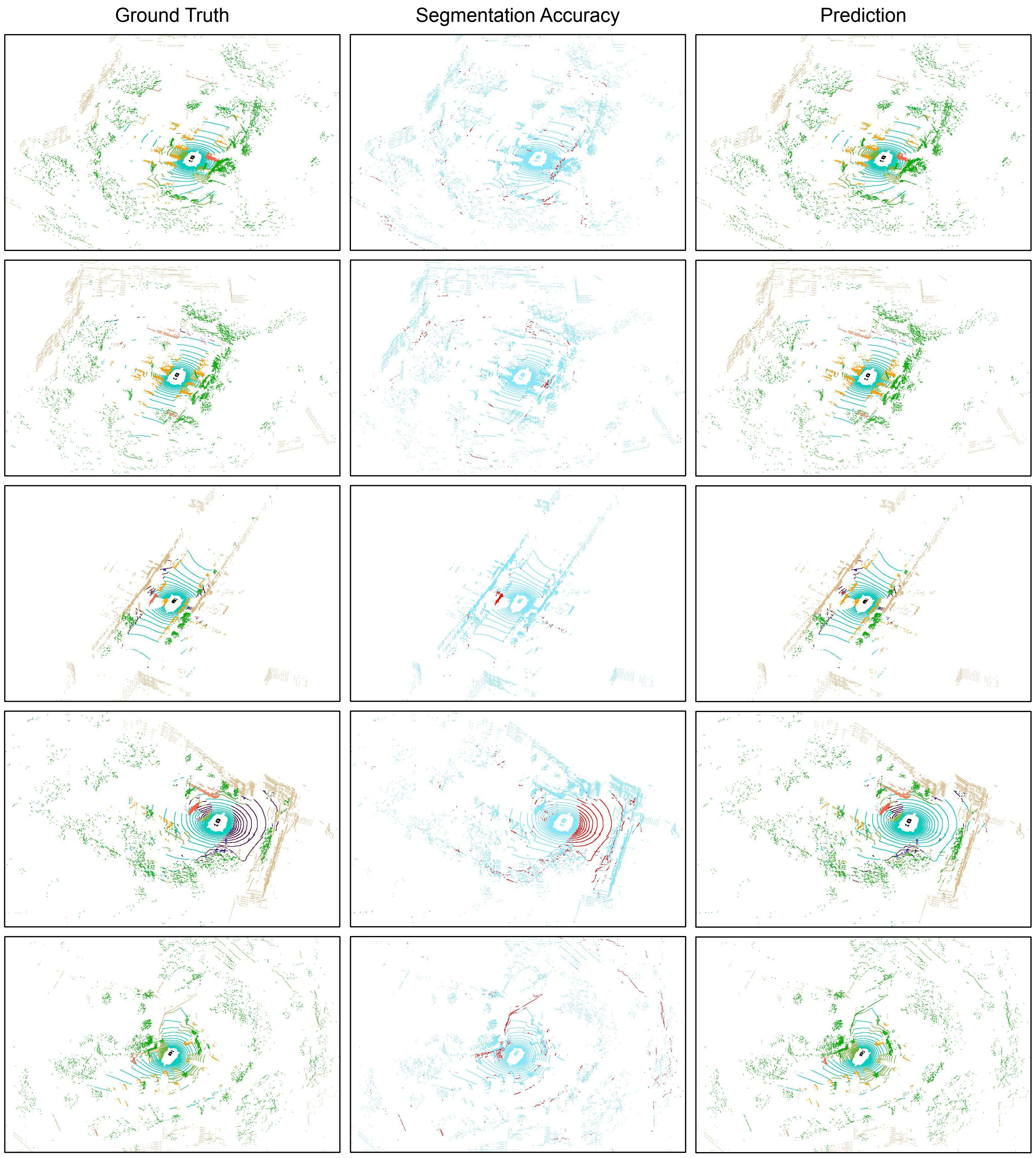

Figure 6. **Segmentation accuracy visualizations** of RangeViT on validation point clouds of nuScenes. In the left column, the points are coloured with their ground truth label colours and in the right column they are coloured with the colour of the predicted label colours. In the middle column, the good predictions are colored in blue and the bad predictions are colored in red.

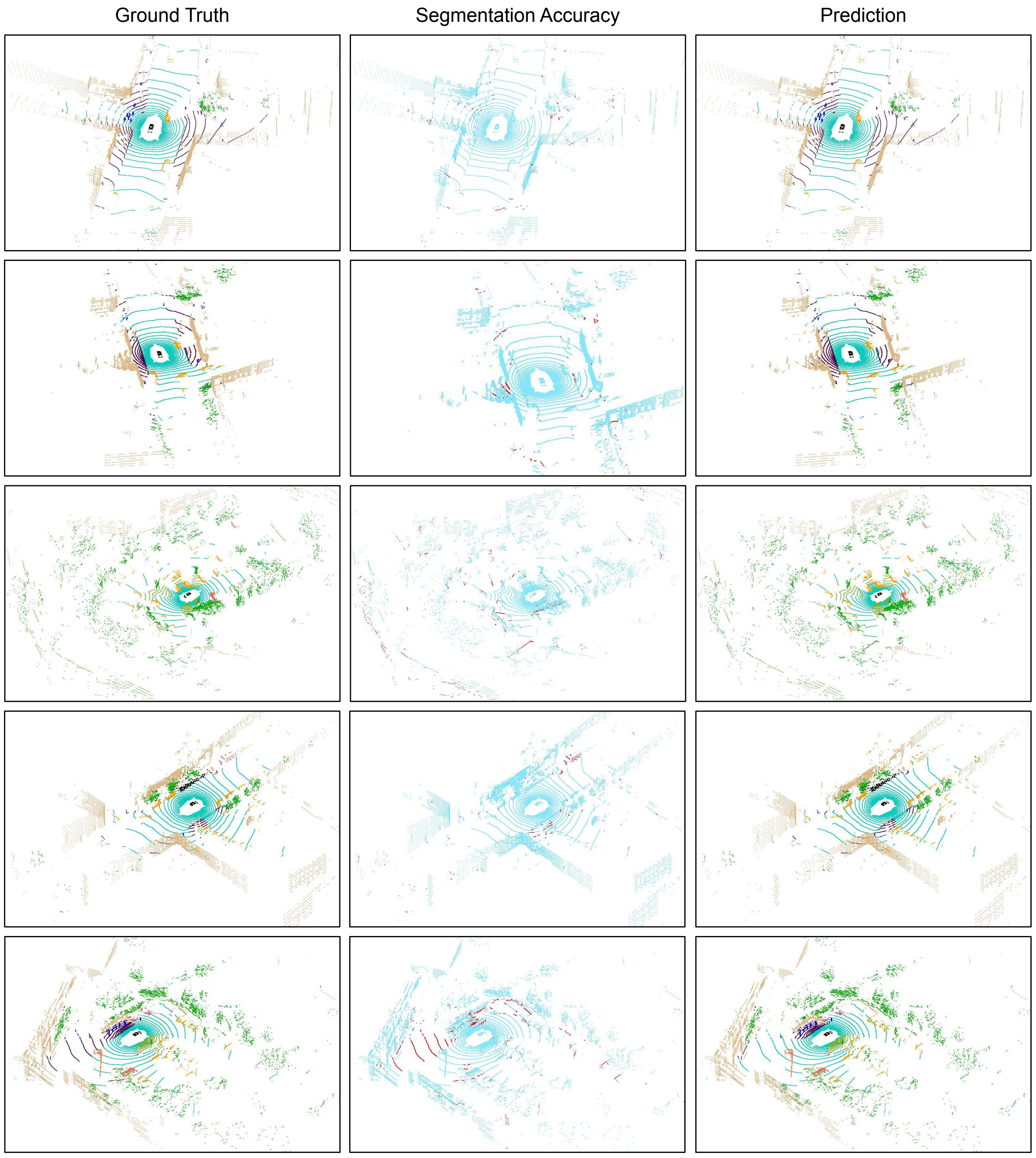

Figure 7. **Segmentation accuracy visualizations** of RangeViT on validation point clouds of nuScenes. In the left column, the points are coloured with their ground truth label colours and in the right column they are coloured with the colour of the predicted label colours. In the middle column, the good predictions are colored in blue and the bad predictions are colored in red.

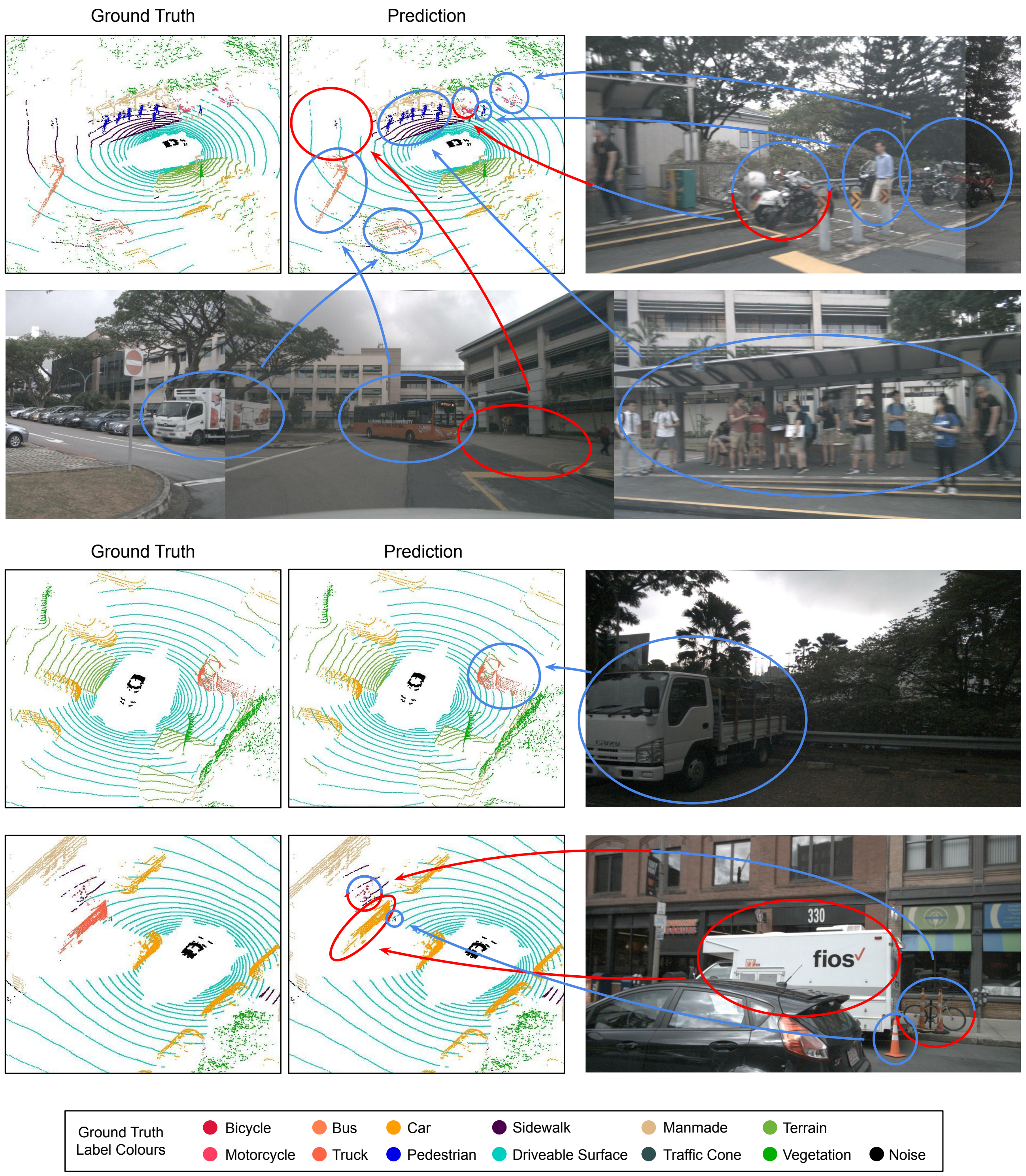

Figure 8. **Segmentation accuracy visualizations** of RangeViT on validation point clouds of nuScenes. The correct predictions are circled in blue, the incorrect ones in red and predictions that are mostly correct except for a few points are circled in half blue-half red. In the first example, the truck, the bus and the pedestrians are correctly predicted. The motorcycles are mostly correctly predicted except for a few points that were predicted as vegetation or manmade. However, part of the sidewalk is predicted as driveable surface and as we can see in the image, there is no height difference between the two, so it is difficult to recognize the sidewalk. In the second example, the truck was correctly predicted. In the third example, the truck was incorrectly predicted as a car and the traffic cones next to the bicycle were predicted as bicycle. The traffic cone next to the truck was however correctly predicted as well as the bicycle.