Supplementary Materials: PointMotionNet: Point-Wise Motion Learning for Large-Scale LiDAR Point Clouds Sequences

Jun Wang[1*] Xiaolong Li[2*] Alan Sullivan[3] Lynn Abbott[2] Siheng Chen [4]
[1]University of Maryland [2]Virginia Tech [3]Mitsubishi Electric Research Labs (MERL)
[4]Shanghai Jiao Tong University
junwang@umiacs.umd.edu, {lxiaol9,abbott}@vt.edu,
sullivan@merl.com, sihengc@sjtu.edu.cn

1. More Experimental Results Details

As shown in Table 1 and Table 2, we give more details on PointMotionNet performance in the multisweep semantic segmentation on SemanticKITTI [1] test set. Without bells and whistles, our approach outperforms previous SOTA [6] by over 3.3 % mIoU with 25 classes including 6 objects.

2. More Ablation Studies

To better explore the performance bound of our method, and understand how different parameters would affect the model performance, we designed several more experiments over the choice of input frame number, number of proxy points. The results are shown in Table 3.

Number of frames. Using more frames as input would bring in richer information of the 3D scene regarding both geometric features and motion dynamics, but also more complexity to the learning process. We tested our model's performance on motion estimation by inputting 2-4 sequential frames, while other parameters are fixed. As demonstrated by Table 3, our model could still work well when taking 4 frames as input, and it actually achieves the best performance under the fact that more computation is involved. To achieve a better balance between efficiency and accuracy, we consider 3 frames as our final setting for our motion estimation task.

Number of proxy points. Proxy points interact with all spatiotemporal neighbors at multiple scales in our pyramid network, and their number will affect the model performance. By decreasing the proxy points number from 15 to 3, we found that the accuracy drops consistently on future motion vector regression, but we don't necessarily always need that much proxy points. Using 9 or 12 proxy points could still work very well for those slow-moving points.

*The first two authors contributed equally. Part of this work was done when JW and XL were interns at Mitsubishi Electric Research Labs.

3. Point-STC Kernel

We leverage proxy point sets to better capture local geometric features and make the training more stable and effective. As originally introduced by [5], these proxy points could act like anchors or grid to learn 3D features in a convolutional way in 3D space. Ideally we want these proxy points to be as far as possible from each other. So in practice, we formulate it as an optimization problem where the optimal points are bounded within a unit ball. Below is a detailed description of the implementation. Following [5], we first formulate it as:

$$\underset{\delta_i,\delta_j}{\operatorname{argmax}} \sum_i \sum_{j\neq i} (\delta_i - \delta_j)^2 \tag{1}$$

$$|\delta_i| < 1, |\delta_j| < 1 \tag{2}$$

δ_0 is fixed at the origin during the iterative process, we further add the regularization item $\|\delta_i\|$ as an attractive potential to avoid those proxy points diverging indefinitely during optimization [5]. We randomly generate a hundred seed sets of proxy points, and then use the analytical form of the gradient to gradually optimize over all the points. The maximum iteration is set to 10000, and we stop the iteration when the gradient norm changes is smaller than $1e-5$. We select the set that converges best from all one hundred seed sets. To match with the scale in each layer, and better capture all neighborhoods, the proxy point sets are re-scaled with respect to the neighborhood sampling radius by multiplying with $0.66r_i$, r_i is the neighborhood sampling radius. A random rotation is applied to the proxy points during the initialization of each Point-STC layer. Visualizations of the proxy points are shown in Figure 3.

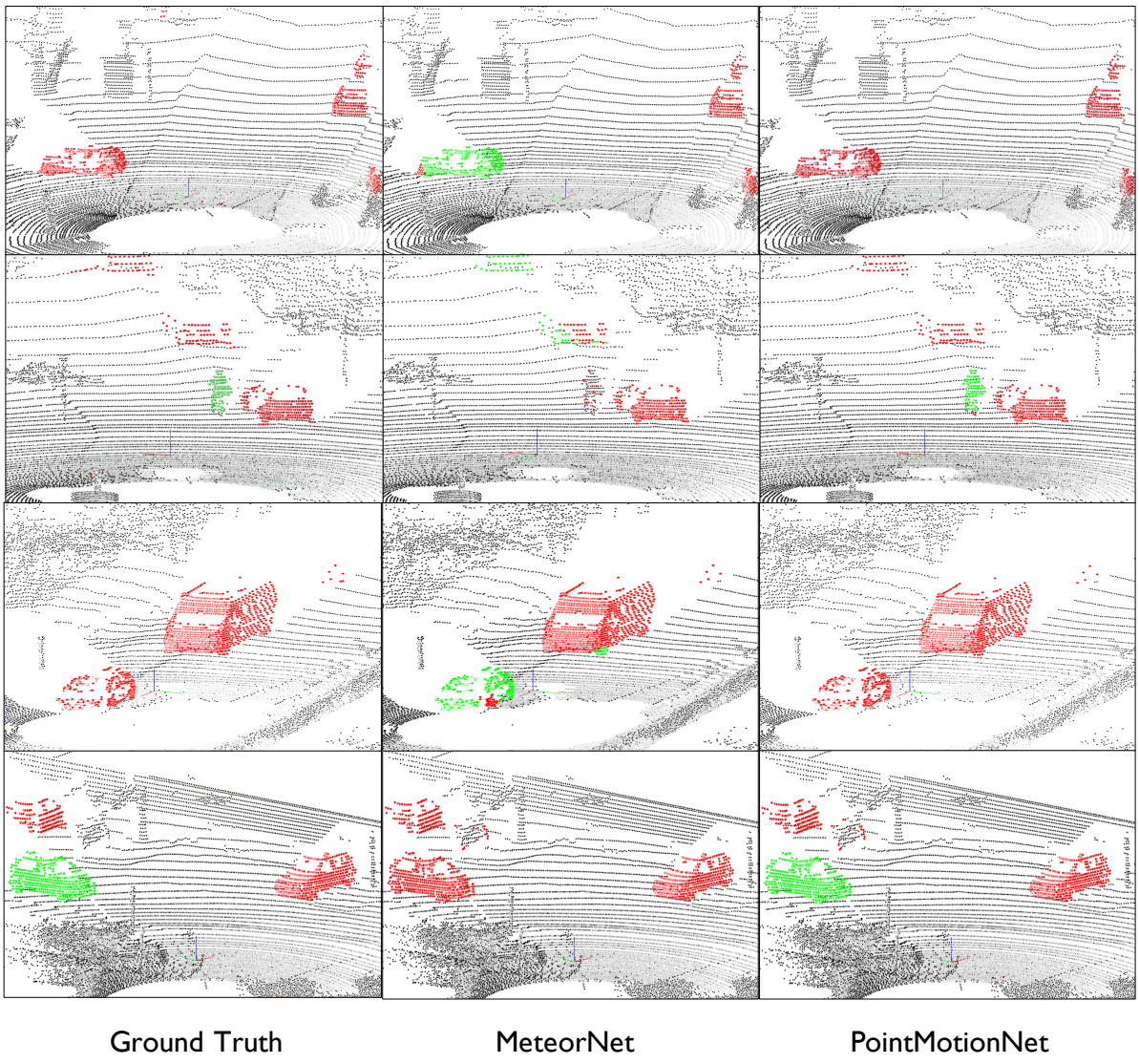

Figure 1. PointMotionNet (right) visually outperforms MeteorNet (middle) in motion state estimation on SemanticKITTI. For six foreground classes, moving points are in green and static ones are in red; for the remaining background classes, points are in grey.

4. Implementation Details

4.1. Argoverse Motion Vector Dataset Creation

Since Argoverse dataset [2] doesn't provide motion vectors information, we generate motion vectors annotations by leveraging the labels from Argoverse 3D Tracking v1.1. We end with 6561 samples for training, and 2507 samples for validation. The preprocessing could be divided into two steps:

Ego-motion compensation. Each original frame is taken from its own local coordinate while the vehicle is moving forward. After selecting a short video clip containing several sequential scans, we set a frame as current frame, and synchronize all other frames to current frame, together with the original 3D bounding boxes. More specifically, given the vehicle poses registered in global map, M_t of current frame S^t_{local}, $M_{t+\tau}$ of another frame $S^{t+\tau}_{local}$, we could compute the synchronized point cloud coordinates $S^{t+\tau}_{synced} = M^{-1}_{t+\tau} M_t S^{t+\tau}_{local}$, here M is 4 $\times$ 4 homogeneous transformation matrix, while S is padded with 1 to create homogeneous coordinates. The same operation is applied to the center and 6 corners of each 3D bounding box in frame $t + \tau$ to get synchronized pose in current frame;

Motion vector assignment. Using the unique tracker ids, we could associate poses between current frame and synchronized future frames for each instance belonging to current frame, and compute the relative 6D pose transfor-

Approach	25-class mIoU	12-class mIoU	car (S)	bicycle	motorcycle	truck (S)	other-vehicle (S)	person (S)	bicyclist (S)	motorcyclist (S)	road	parking	sidewalk	other-ground	building	fence
TangentConv [4]	34.1	20.7	84.9	2.0	18.2	21.1	18.5	1.6	0.0	0.0	83.9	38.3	64.0	15.3	85.8	49.1
DarkNet53 [1]	41.6	24.2	84.1	30.4	32.9	20.0	20.7	7.5	0.0	0.0	91.6	64.9	75.3	27.5	85.2	56.5
SpSeqnet [3]	43.1	25.5	88.5	24.0	26.2	29.2	22.7	6.3	0.0	0.0	90.1	57.6	73.9	27.1	91.2	**66.8**
KP-Conv [5]	51.2	**40.5**	93.7	44.9	47.2	42.5	**38.6**	**21.6**	0.0	0.0	86.5	58.4	70.5	26.7	90.8	64.5
Cylider3D [6]	51.5	31.4	**93.8**	**67.6**	**63.3**	41.2	37.6	12.9	0.1	0.1	90.4	**66.3**	74.9	**32.1**	**92.4**	65.8
PointMotionNet	**54.8**	39.2	**93.8**	59.3	59.9	**44.2**	38.1	13.7	**23.1**	**23.4**	**91.7**	65.5	**76.1**	24.2	90.1	63.5

Table 1. PointMotionNet outperforms in multisweep semantic segmentation on SemanticKITTI test set. Note that S represents the static foreground category, and M denotes the moving foreground category.

Approach	25-class mIoU	12-class mIoU	vegetation	trunk	terrain	pole	traffic-sign	car(M)	bicyclist(M)	person(M)	motorcyclist(M)	other-vehicle(M)	truck(M)
TangentConv [4]	34.1	20.7	79.5	43.2	56.7	36.4	31.2	40.3	1.1	6.4	1.9	30.1	42.2
DarkNet53 [1]	41.6	24.2	78.4	50.7	64.8	38.1	53.3	61.5	14.1	15.2	0.2	28.9	37.8
SpSeqnet [3]	43.1	25.5	84.0	66.0	65.7	50.8	48.7	53.2	41.2	26.2	36.2	2.3	0.1
KP-Conv [5]	51.2	**40.5**	84.6	70.3	66.0	57.0	53.9	69.4	0.5	0.5	**67.5**	**67.4**	**47.2**
Cylider3D [6]	51.5	31.4	**85.4**	**72.8**	68.1	**62.6**	61.3	68.1	0.0	0.1	63.1	60.0	0.4
PointMotionNet	**54.8**	39.2	84.9	70.9	**70.1**	62.2	**64.0**	**71.4**	**62.2**	**61.8**	15.1	29.9	10.5

Table 2. PointMotionNet outperforms in multisweep semantic segmentation on SemanticKITTI test set. Note that S represents the static foreground category, and M denotes the moving foreground category.

proxy points(k)	# frames	$\text{mIoU}_{\text{total}} \uparrow$	$\text{mIoU}_{\text{static}} \uparrow$	$\text{mIoU}_{\text{moving}} \uparrow$	Static $\downarrow$	Speed $\leq$ 5m/s $\downarrow$	Speed $>$ 5m/s $\downarrow$
15	3	**0.985**	0.994	**0.976**	**0.001**	0.416	0.445
15	*2*	0.980	0.992	0.969	0.001	0.453	0.568
15	*4*	0.982	**0.996**	0.969	0.001	**0.341**	**0.422**
12	3	0.984	0.994	0.975	0.002	0.395	0.489
9	3	0.982	0.992	0.971	0.002	0.406	0.559
6	3	0.981	0.992	0.970	0.002	0.461	0.543
3	3	0.964	0.985	0.942	0.003	0.588	0.847

Table 3. Additional Ablation Study of **PointMotionNet** on Argoverse Validation Set. We further test different number of proxy points k.

mation between them. Then for each bounding box in current frame, we will first rotate it to align with xyz axis, and the same transformation is applied to the whole point cloud. So we could check the membership with the aid of aligned axis to find all points that are inside the 3D bounding box. Finally we apply the relative transformation to all points belonging to current bounding box, the substraction between transformed coordinates and original coordinates will result in the 3D motion vector annotations. We apply the same process to all 5 future frames.

	Argoverse	SemanticKitti
Proxy points	15	15
Frame number	3	2
Parameters	26.7M	23.2M
Training (sample/sec)	4.8	4.9
Inference (sample/sec)	17.2	28.5
Avg pts/sample (during training)	15k	21.7k
Avg GPU mem./sample (during training)	10.0G	7.5G
Avg pts/sample (during inference)	8.6k	8.3k
Avg GPU mem./sample (during inference)	3.0G	2.7G

Table 4. **Model sizes and running speed.**

4.2. Model sizes and speed

We split a large-scale point cloud into multiple patches, which are small-scale point clouds, and handle each patch locally. During data loading stage, each patch may have different number of points, we concatenate different patches and use batch size 1 to avoid zero padding. We consistently test our data loading and model inferring speed on Nvidia V100 GPU, as shown in Table 4.

4.3. Architecture specs

As shown in Figure 4, we draw the detailed architecture of PointMotionNet with specific layer dimensions. There are 4 down-sampling and up-sampling stages. The pyramid architecture enables spatiotemporal feature learning at multiple scales. Note that most of the weights come from the spatiotemporal feature encoder, while light-weight modules are used in decoding stage, in this way we could allow more effective usage of computation resources.

5. Additional Qualitatively Visualization on Argoverse and SemanticKITTI

We report additional visualization results of our PointMotionNet on the Argoverse and SemanticKITTI in Figure 1 and Figure 5, respectively.

6. Limitations

A typical failure case of our prediction is that points belonging to the same instance might not have the similar predicted motion vectors, as shown in Figure 2. It's possible to further leverage instance labels to ease this issue, and make the predictions more consistent. In addition, PointMotionNet is a fully-supervised method, requiring large-scale annotated data. One future direction is to explore semi-supervised learning to ease the problem.

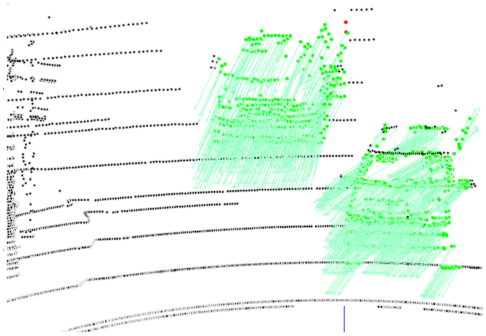

Figure 2. Failure case. Some motion vectors on the right car are not well predicted, making the predictions on the same instance not quite consistent.

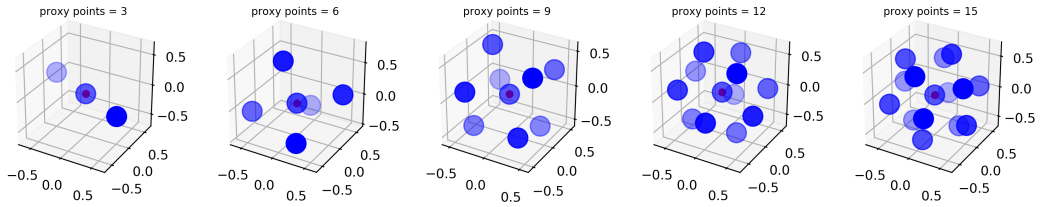

Figure 3. **Distribution of proxy points in 3D space**. The center small darker point denotes the origin, all those blue points are proxy points.

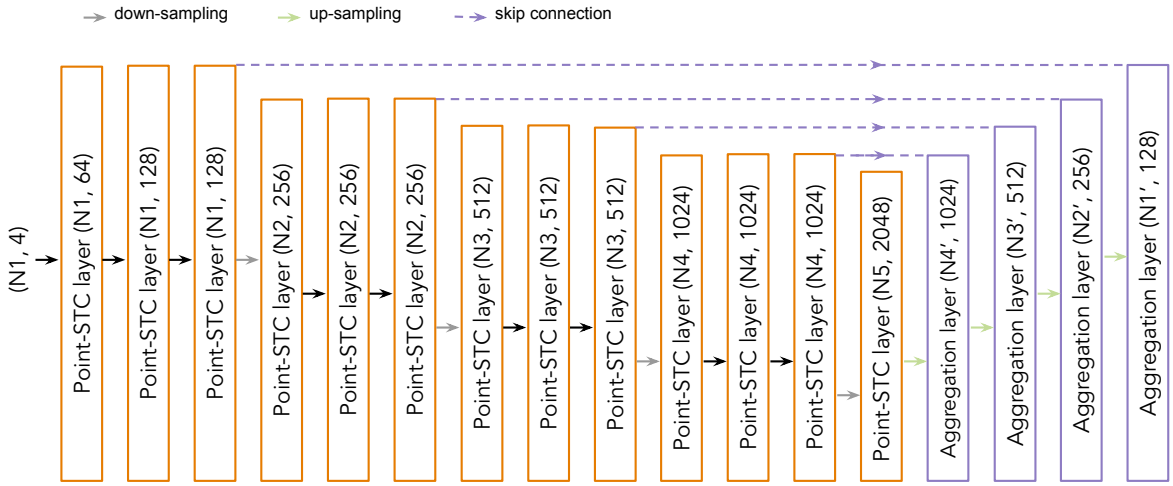

Figure 4. PointMotionNet architecture

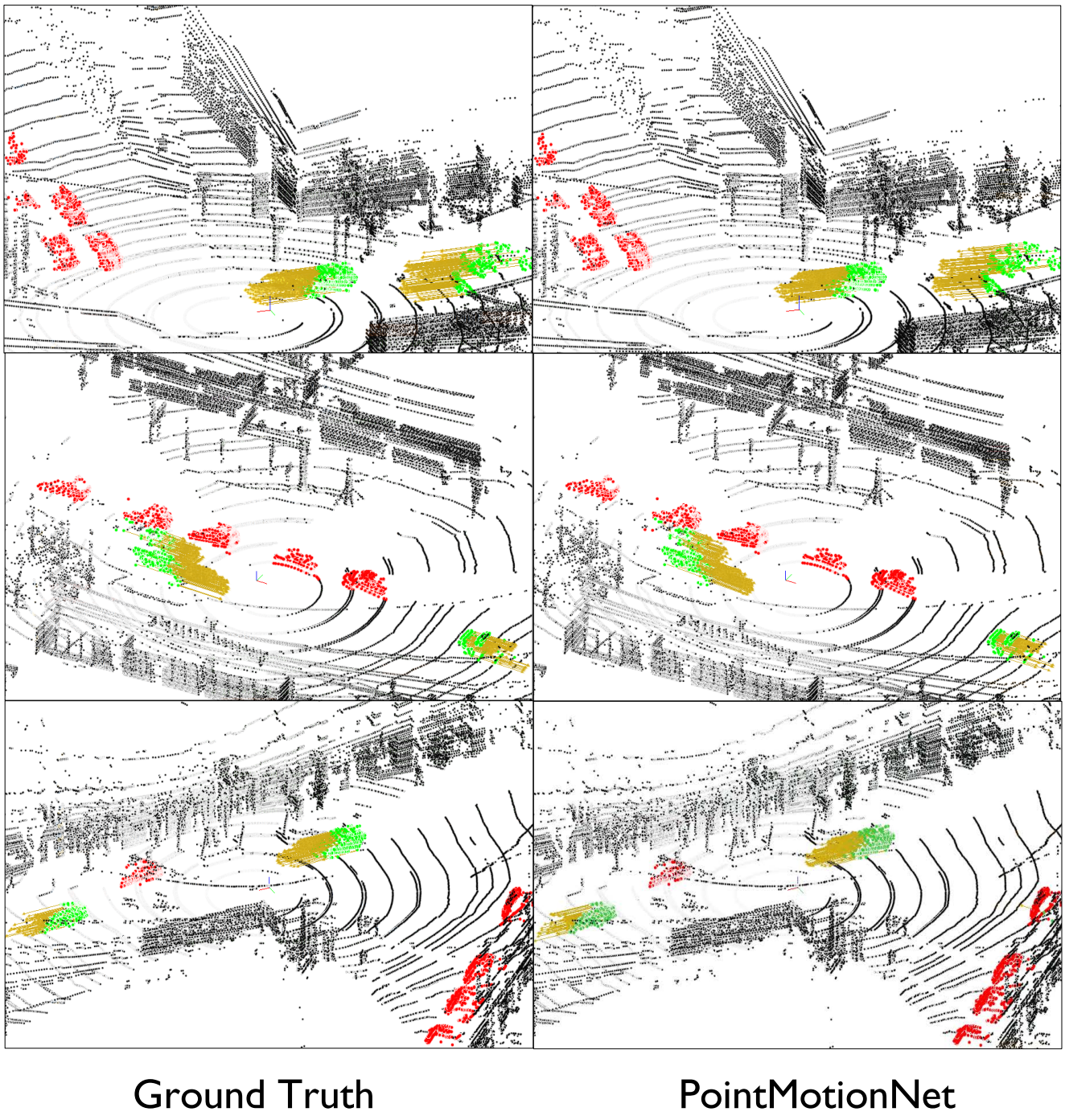

Figure 5. PointMotionNet (right) vs Ground Truth (left) in point-based motion prediction on Argoverse. Green moving points are associated with brown motion vectors and the static points are in red; the remaining points are background, marked in grey.

References

[1] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019. 1, 3

[2] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[3] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4583, 2020. 3

[4] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 3

[5] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Franccois Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. 1, 3

[6] 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. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1, 3