2020 25th International Conference on Pattern Recognition (ICPR)
Milan, Italy, Jan 10-15, 2021

Human Segmentation with Dynamic LiDAR Data

Tao Zhong*, Wonjik Kim†, Masayuki Tanaka‡ and Masatoshi Okutomi§
Department of Systems and Control Engineering, School of Engineering,
Tokyo Institute of Technology, Meguro-ku, Tokyo 152-8550, Japan
Email: *tzhong@ok.sc.e.titech.ac.jp, †wkim@ok.sc.e.titech.ac.jp, ‡mtanaka@sc.e.titech.ac.jp, §mxo@sc.e.titech.ac.jp

***Abstract*—Consecutive LiDAR scans compose dynamic 3D sequences, which contain more abundant information than a single frame. Similar to the development history of image and video perception, dynamic 3D sequence perception starts to come into sight after inspiring research on static 3D data perception. This work proposes a spatio-temporal neural network for human segmentation with the dynamic LiDAR point clouds. It takes a sequence of depth images as input. It has a two-branch structure, i.e., the spatial segmentation branch and the temporal velocity estimation branch. The velocity estimation branch is designed to capture motion cues from the input sequence and then propagates them to the other branch. So that the segmentation branch segments humans according to both spatial and temporal features. These two branches are jointly learned on a generated dynamic point cloud dataset for human recognition. Our works fill in the blank of dynamic point cloud perception with the spherical representation of point cloud and achieves high accuracy. The experiments indicate that the introduction of temporal feature benefits the segmentation of dynamic point cloud.**

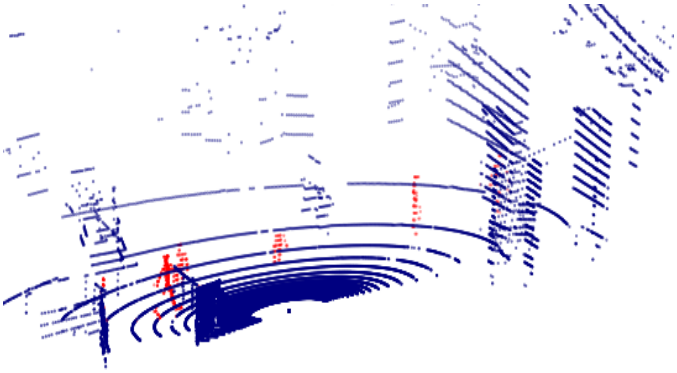


Fig. 1: An example of segmentation results on generated data. Blue points are classified as background, and red points are classified as human.

I. Introduction

Accurate and fast 3D environment perception is of great importance in robotics field, as real-time geometric information of pedestrians is necessary for the following motion planning. Light Detection and Ranging (LiDAR) is especially attracting research and industry interest because of its large detection range, robustness under various conditions, descending price, etc. While early researches [1] used handcraft feature in point cloud processing, learning-based methods are also able to extract the features these days.

In most of the current datasets, the point cloud generated within one scan is formatted as one frame. Therefore, when the detection or segmentation is done frame by frame, each frame is treated as independent static 3D data. Many researchers have successfully applied deep learning methods to static point cloud perception. They [2], [3], [4], [5], [6] extract spatially local and global features based on various representations of point cloud, such as point-wise method, voxel-based method and spherical representation method, etc. Subsequently, the features are used for classification, object detection, and semantic segmentation.

In the real world, LiDAR rotates constantly, generating consecutive sequences of point cloud frames. These 3D videos are dynamic and thus contain abundant spatial and temporal cues. However, compared to the static point cloud, there is so far few researches regarding the dynamic point cloud. One of the reasons is the lack of annotated training data, since labeling millions of points in sequences costs undoubtedly a tremendous amount of time and resources. Recently, the release of several synthetic and real datasets [7], [8], [9] for dynamic point cloud is accelerating some related researches. The point-wise method [10] and voxel-based method [11], which were already proved effective in static point cloud domain, have been extended to dynamic point cloud domain. They have demonstrated better performance than before, and showed the benefits of spatio-temporal features.

The proposed architecture in this work is inspired by the two-stream networks for unsupervised video object segmentation [12], which constructed communication of spatial and temporal features. With the help of a dataset that provides both per-point classification annotation and velocity annotation for synthetic LiDAR data, we train a network that can jointly predict human segmentation and velocity map in the dynamic point cloud. Consequently, different from the previous works that make use of the temporal information implicitly, we are able to extract it explicitly and leverage it in the segmentation task. Moreover, the proposed model is based on the spherical representation method, taking a sequence of 2D range images of point cloud sequences as input. To the best of our knowledge, our work is the first one that study the dynamic point cloud segmentation with this representation.

The experiments show that the proposed model achieves higher accuracy than the existing state-of-the-art methods on the human segmentation dataset. The utilization of temporal cues effectively improves the performance of the human detection. We also investigate the effect of the length of frames which we use for human detection.

978-1-7281-8808-9/20/$31.00 ©2020 IEEE

2020 25th International Conference on Pattern Recognition (ICPR) | 978-1-7281-8808-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICPR48806.2021.9413014

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.

All data and sample code are available online[1].

II. Related Works

A. Static point cloud perception

According to the way of handling the unordered point cloud, recent learning-based methods can be divided into two types: point-based networks [3], [13], [14] and projection-based networks [2], [4], [15], [6].

Point-based methods aim to directly process raw points. As the groundbreaking work, PointNet [3] takes the coordinates of points as input. The global feature is obtained from the point-wise feature with a symmetric function. PointNet++ [13] further takes the local structure into consideration by grouping spatially close points. The point-based method is good at extracting permutation-invariant features, but it also comes with relatively high computation and memory requirements.

Projection-based methods first convert point cloud into intermediate representations. Projecting point cloud onto multi-view planes [4] makes implementing 2D convolution on each view possible. View-wise features are then aggregated into the global feature. Voxelization of point cloud [16] generates 3D tensors, on which 3D convolution can be applied. Another widely used method is spherical representation [15], [17], [18], [6], which projects point cloud onto a spherical plane. The obtained depth image can then be fed into common convolutional networks. SqueezeSeg [15] is a typical and efficient model of this kind. Regrettably, the temporal relationship between frames is ignored when they are considered separately. Based on spherical representation, our work captures the spatiotemporal cues in sequences of depth images.

B. Dynamic point cloud perception

Due to the difficulty of annotation, most of the dynamic point cloud datasets are generated with simulation. Kim et al. developed a pipeline for single frame [19] [20] and sequence of point cloud data generation [7]. SYNTHIA [8] dataset provided many road scene videos generated by a game engine. Behley et al. [9] annotated all the frames of KITTI's [21] odometry dataset and released the SemanticKITTI dataset.

The methods in static point cloud perception also apply to the dynamic condition with some adjustment. Meteornet [10] is a point-wise method based on PointNet. They extended the spatial neighborhood to a temporal and spatial neighborhood when extracting the local features of points. The features were then used for downstream tasks: action recognition, segmentation, and scene flow estimation. PointRNN [22] applies recurrent models to aggregate local features from different time steps. As for the voxel-based method, Luo et al. [5] constructed 4D tensors by concatenating the occupancy grid of point cloud along the time axis. They then performed 3D convolution on the temporal dimension to capture motion features, which benefited the detection, tracking, and motion forecasting tasks. Choy et al. [11] restrained the computation cost of high-dimensional convolution by substituting the 2D convolution layers of UNet [23] with sparse 4D convolution layers. However, when it comes to the accuracy of pedestrian segmentation, the improvement relative to the static method was not so significant. This indicates that extracting temporal features in such an implicit manner might fail. Compared with the above approaches, we leverage temporal information more explicitly.

C. Video Object Segmentation

In unsupervised video object segmentation task, there is no reference mask as guidance. Therefore, motion cues are necessary when searching for foreground objects. As a representative kind of motion field representation, optical flow [24] is frequently used. Some researchers [25], [26] strengthened the feature by concatenating the warped feature map of previous frames to that of the current frame. Segflow [12] constructed a two-stream network that predicts segmentation and optical flow simultaneously. They proved that the bidirectional communication of feature maps between two streams can boost the performance of both tasks. Our method is inspired by this work, forcing the network to learn motion cues from the velocity estimation task.

III. Proposed Method

To segment humans in the dynamic point cloud with the help of motion information, we construct a two-branch network, one is the segmentation branch, and the other is the velocity estimation branch. It takes a sequence of depth images as input and predicts the class and velocity of each point in the last frame at one stage. In the following, we first explain the structure of each branch and the communication of temporal and spatial features between them. Then we briefly introduce how we generate sequential LiDAR data with segmentation and velocity annotation. The overall architecture of the proposed model is shown in Fig. 2.

A. Segmentation Branch

The input of the segmentation branch is the last frame of the sequence. Therefore, it only extracts the spatial features and the size of the input tensor is $h \times w \times 1$, where h and w are the height and width of the range image. The output is the predicted probability distribution over classes for every pixel in this frame.

The segmentation branch has an hour-glass-shaped structure, consisting of a contraction section, an expansion section and skip connections. Considering that the resolution in the vertical direction is much lower than that in the horizontal direction, in the contraction section, the input frame is only downsampled in the horizontal direction. In our case, every two convolutional layers are followed by a max pooling layer with pool size [1 2], and the kernel size of convolution is 3 $\times$ 3. After three downsampling operations, the spatial feature maps with a size of $h \times \frac{1}{8}w \times C$ are upsampled back to the original image resolution in the expansion section. C denotes the number of feature channels. In order to refine the segmentation result, the feature maps with size of $h \times w \times$

[1]http://www.ok.sc.e.titech.ac.jp/res/LHD/

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.

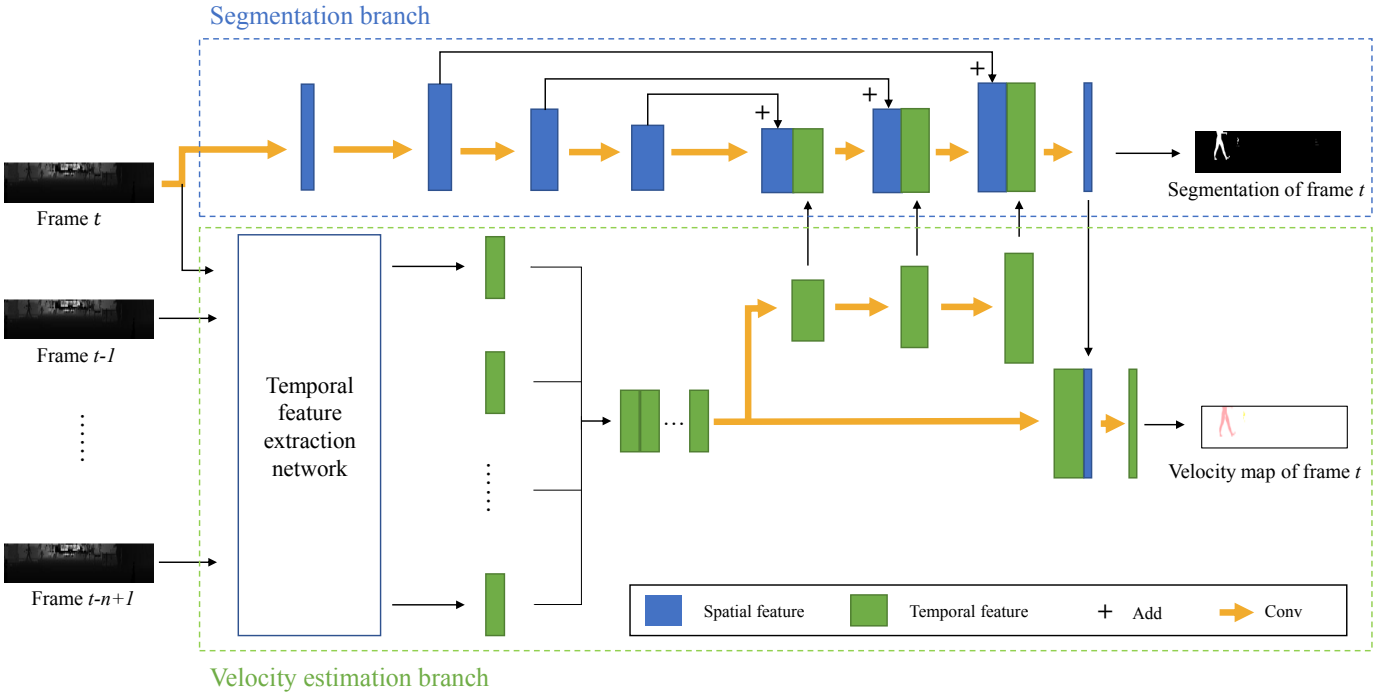


Fig. 2: The proposed network architecture which consists of the segmentation branch and the velocity estimation branch. They communicate spatial and temporal features in the expansion section. Frame t denotes the current scan, while frame $t-1$ to $t-n+1$ denote the last $n-1$ scans.

$\frac{1}{8}C$, $h \times \frac{1}{2}w \times \frac{1}{4}C$, $h \times \frac{1}{4}w \times C$ before pooling layers in contraction section is added to those of the same size in the expansion section via skip connection. The channel number C equals to 512. The activation function of the last layer is softmax.

B. Velocity Estimation Branch

The velocity estimation branch takes multiple consecutive frames as input, extracting temporal features. The size of the input tensor is $n \times h \times w \times 1$, where n represents the number of input frames. The output is the predicted velocity for every pixel in the last frame.

Each input frame first goes through a weight shared feature extraction network. Its backbone is fully convolutional network [27]. The outputs are n feature maps of corresponding input frames, the size of which is $h \times \frac{1}{16}w \times \frac{1}{8}C$.

Since velocity is one kind of motion information, the feature maps in this branch are supposed to encode motion cues. Therefore, we concatenate them along the channel axis and regard the generated tensor as temporal feature maps. In order to make use of these features in the process of segmentation, we propagate the temporal feature maps to the segmentation branch. Specifically, they are up-sampled to match the size of spatial features in the expansion section and concatenated, illustrated by the green arrows in Fig. 2. In reverse, the predicted label map from the segmentation branch is also concatenated to the temporal feature maps as guidance. Finally, the last layer with linear activation decodes them into the $h \times w \times m$ velocity map after several convolutional layers. m denotes the number of dimensions of the velocity vector, which in our case is two to represent the horizontal plane.

C. Loss Function

For the loss function of the segmentation branch, we use the pixel-wise categorical cross-entropy. For the loss function of the velocity estimation branch, we use Mean Square Error (MSE). Hence, the total loss used in the optimization process is the combination of segmentation loss and velocity estimation loss for the background and human. The losses for background and human velocity estimation are computed separately. The defected pixels are not considered in the calculation.

Given n input depth maps $\boldsymbol{d}_t$, $\boldsymbol{d}_{t-1}$, $\cdots$, estimated label map and velocity map for t-th frame can be expressed as

$$[\hat{\boldsymbol{l}}_t | \hat{\boldsymbol{v}}_t] = \boldsymbol{f}(\boldsymbol{d}_t, \cdots, \boldsymbol{d}_{t-n+1}; \boldsymbol{\theta}), \quad (1)$$

where $\hat{\boldsymbol{l}}_t$ is estimated label map for the t-frame, $\hat{\boldsymbol{v}}_t$ is estimated velocity map for the t-frame, and $\boldsymbol{f}(\cdots;\boldsymbol{\theta})$ represents the network with weights $\boldsymbol{\theta}$. The total loss is defined as below:

$$L_t = \frac{\lambda_c}{N_c}\sum_i \mathrm{CE}(\hat{l}_{t,i}, l_{t,i}) + \frac{\lambda_h}{N_h}\sum_{j=\{i|l_{t,i}=h\}} ||\hat{\boldsymbol{v}}_{t,j} - \boldsymbol{v}_{t,j}||_2^2 + \frac{\lambda_b}{N_b}\sum_{j=\{i|l_{t,i}=b\}} ||\hat{\boldsymbol{v}}_{t,j} - \boldsymbol{v}_{t,j}||_2^2, \quad (2)$$

where CE represents a cross-entropy, $l_{t,i}$ is a true label for i-th pixel of t-th frame, $||\cdot||_2^2$ represents L2 norm, $\boldsymbol{v}_{t,j}$ is a true velocity for i-th pixel of t-th frame, h and b represent human and background labels, N_c, N_h, and N_b are the number of

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.

all pixels, human label pixels, and background pixels, and λ_c, λ_h, and λ_b are hyper-parameters.

D. Data Generation

As an extension of 2D optical flow, scene flow [28] denotes the translation vectors of points in 3D space. Unfortunately, as far as we know, there exists so far no point cloud dataset that provides both segmentation and scene flow annotation, due to the difficulty of labeling. Alternatively, we generate sequential LiDAR data with pixel-wise segmentation information and pixel-wise velocity information for training and evaluation, using the pipeline proposed in [7].

They are generated by combining the depth map of real background data and the depth map of synthetic human walking models [29]. At the same time, the speed of LiDAR and human models is recorded. Consequently, each point is automatically labeled with its class and velocity. The points are classified into background and human. The velocity of a human point denotes the relative walking speed of the human model it belongs to, with respect to the origin of LiDAR at x, y axis; the velocity of a background point is the relative speed of the background, i.e. $-v_{LiDAR}$. The forward direction of the LiDAR is designated as x axis.

IV. Experiments

We evaluate the proposed network on a large-scale dataset for human segmentation. The performance is compared with multi-frame methods and a single-frame method. We also perform ablation studies on the influence of the velocity estimation branch and the length of the input sequence.

A. Dataset and Training details

The Automatic Labeled LiDAR Sequence dataset is composed of 1108 generated data sequences and 100 real data sequences. The background data of generated data is collected in the 3rd floor of Miraikan [30] using an HDL-32E LiDAR, and the real data is collected at the same place with pedestrians walking around. Each sequence is composed of 32 frames, the size of each frame is 32 $\times$ 1024. The data is accessible at http://www.ok.sc.e.titech.ac.jp/res/LHD/

There are total 1108 generated sequences and 100 real sequences in our data set. We use 900 generated sequences for training, 100 generated sequences for validation. Then the networks are tested with the remaining 108 generated sequences and 100 real sequences. We train the network using Adam optimizer with an initial learning rate of 3e-5 and a decay rate of 3×10^{-5}. The batch size is set to 1 and 250 steps compose an epoch. The network is trained for 1000 epochs. The hyper-parameters λ_c, λ_h, λ_b in Eq. 2 are set as 1×10^5, 1, and 1001, in order to give more weights on the human points than background points.

B. Evaluation Metrics

In consideration of the overwhelming number of background points, the performance is evaluated with intersection-over-union (IoU) score of human class, where points predicted as humans are regarded as positive. Hence, IoU is calculated with the total number of True Positives (TP), False Positives (FP), and False Negatives (FN) as below:

$$IoU = \frac{TP}{TP + FP + FN} \tag{3}$$

Note that defected pixels are not considered for evaluation.

TABLE I: Ablation study on components of velocity estimation branch. The performance is evaluated with IoU(%) score of human class in different ranges.

Method			Proposed		
Number of frames			4		
Velocity estimation				√	√
Temporal feature propagation			√		√
Generated data	Criteria range(m)	0 to ∞	81.16	77.11	**86.08**
		0 to 4	92.21	90.98	**94.06**
		4 to 8	63.25	59.07	**72.79**
		8 to ∞	33.65	25.75	**41.44**
Real data	Criteria range(m)	0 to ∞	64.23	64.87	**67.29**
		0 to 4	74.59	75.73	**76.93**
		4 to 8	49.84	51.83	**53.79**
		8 to ∞	23.43	**25.06**	24.76

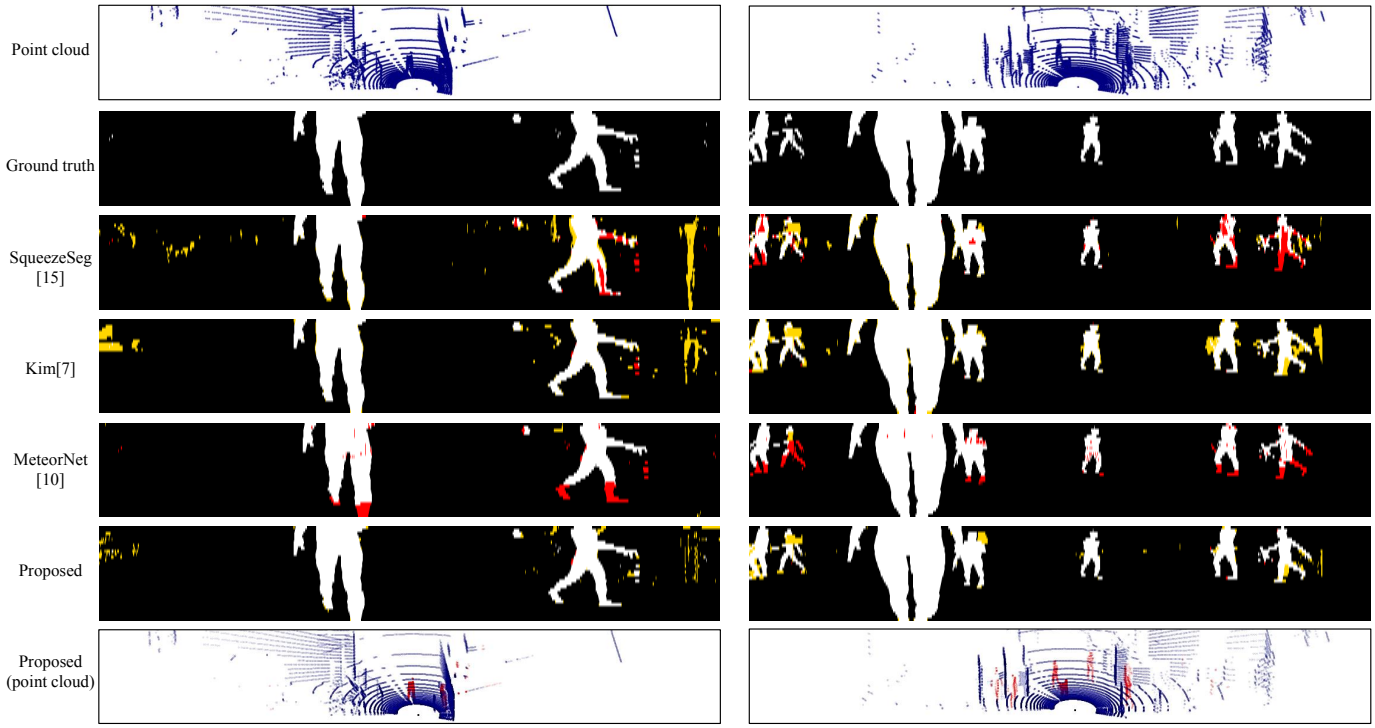
C. Ablation Study

We conduct ablation studies to verify the effect of components in the velocity estimation branch. Furthermore, the IoU score is calculated in several distance ranges in order to verify the effect at different distances. For instance, '0 to 4 [m]' means that only points 0 [m] and 4 [m] far from LiDAR are considered in the calculation.

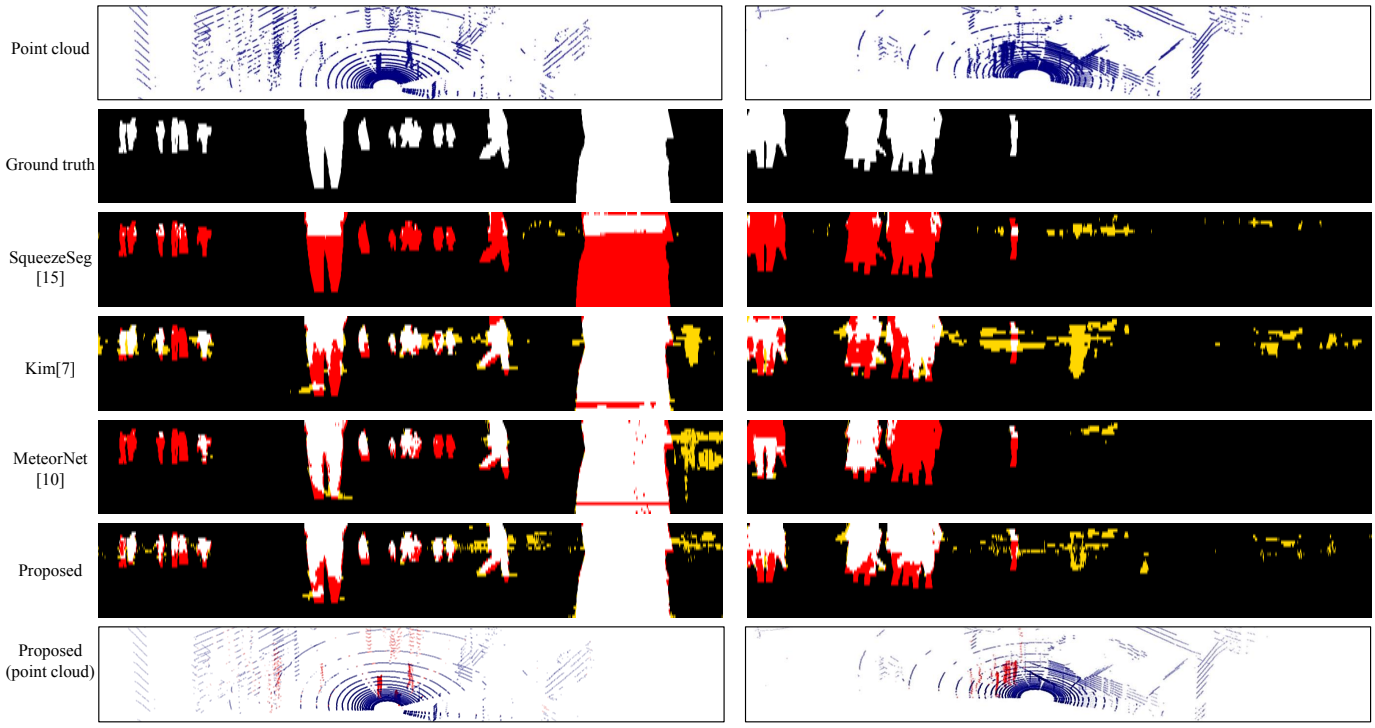
When studying the effect of velocity estimation, we remove the last several layers for decoding the velocity map, i.e., the model only predicts segmentation result. When studying the effect of temporal feature propagation, we cut off the skip connections from the velocity estimation branch to the segmentation branch. As shown in Table I, in the range of '0 to ∞ [m]', temporal feature propagation manages to bring about 9% improvement on generated data and 2.4% improvement on real data, respectively. Especially, in the '8 to ∞ [m]' range of generated data, velocity estimation and temporal feature propagation increase the IoU score by 7.9% and 15.7%, which indicates that motion cues help detection in the distance, where the point cloud is very sparse.

As shown in Fig. 5, the network can estimate the human segmentation and pixelwise velocity map. Then, we can derive the velocity map of the segmented area. The examples of human velocity estimation are illustrated in Fig. 8. According to the Fig. 8, the estimated velocities show similar tendency with ground truth.Asaconsequence,weconcludethatvelocityestimation with LiDAR only is a feasible task.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.



(a) Generated data



(b) Real data

Fig. 3: Comparison of segmentation result on generated data and real data. In 2D plots, white area denotes True Positive and black area denotes True Negative, while red area denotes False Positive and yellow area denotes False Negative.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.

TABLE II: Experiments on the effect of the number of input frames.

Method			Proposed				
Number of frames			1	2	4	8	16
Generated data	Criteria range (m)	0 to ∞	82.01	83.00	86.08	**88.89**	80.73
		0 to 4	91.75	93.17	94.06	**95.10**	90.61
		4 to 8	64.02	67.51	72.79	**78.96**	78.96
		8 to ∞	38.05	34.52	41.44	**46.52**	31.96
Real data	Criteria range (m)	0 to ∞	65.74	66.22	67.29	**67.31**	63.60
		0 to 4	75.85	77.20	76.93	**77.43**	74.28
		4 to 8	52.50	52.77	**53.79**	51.53	46.01
		8 to ∞	23.39	23.57	**24.76**	22.35	23.61

TABLE III: Comparison of networks on the automatic labeled LiDAR sequence dataset for human segmentation. The performance is evaluated with IoU [%] score of human class in the range of '0 to ∞ [m]'.

Method	SqueezeSeg (w/o CRF) [15]	Kim [7]	Meteornet [10]	Proposed
Input	depth	depth	xyz	depth
# of frames	1	4	4	4
Generated	58.87	68.88	73.87	**86.08**
Real	11.35	58.69	52.20	**67.29**
Run time (ms)	**6**	46	840	51

D. Effect of the number of the input frames

The relationship between the number of input frames and the performance is shown in Table II. We evaluate the proposed model with 1-, 2-, 4-, 8-, 16- frame sequences as input. Note that there is no velocity estimation branch when the input is a single frame.

When the length is not larger than eight frames, overall, accuracy increases along with the increase in the number of frames. On generated data, performance in all ranges is improved, and the IoU score in the range of '0 to ∞ [m]' with an 8-frame sequence input is 6.8% higher than with a single frame input. This is attributed to the significantly better prediction accuracy in the range further than 4[m], which ulteriorly proves the advantage of temporal information in dynamic point cloud perception. This tendency is not so clear when it comes to real data. It might be caused by the domain gap between generated data and real data or inaccurate annotations in real data. However, an increase in number of frames also results in an increase in inference time and the accuracy drops significantly when the length reaches 16. Therefore, we consider 4 frame as an appropriate length.

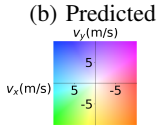
E. Comparison

The performance and runtime of the proposed network is compared to the model proposed by Kim and Meteornet whose input is a 4-frame sequence, as well as SqueezeSeg whose input is a single frame. We implement the SqueezeSeg without the CRF layer because it has better performance in human segmentation according to their paper. The results are shown in Table III. Compared to the model from Kim, the proposed network improves the IoU score of the human class by nearly 17% on generated data and by nearly 6% on real data. It also accomplishes better accuracy than Meteornet and SqueezeSeg. This result shows the effectiveness of our model. Note that compared to the KITTI dataset used by SqueezeSeg and the SYNTHIA dataset used by Meteornet, our data contain neither intensity information nor color information. We used a single GTX 2080 Ti GPU and Intel Core i9 CPU. The run time of projection-based methods is shorter than the point-based method, Meteornet. For 4-frame input sequence, the proposed model runs at 51 ms/seq.

Figure 3 is the 2D and 3D visualization of the segmentation results. Please refer to Figure 1 for the 3D plot of prediction on generated data. It can be seen that the proposed model is able to recognize nearly all the human points of generated data with a few False Positives. On real data that are more difficult, the proposed model demonstrates its ability to restrain False Negative more obviously, which is important in the real robotics application.

(a) Ground truth

(b) Predicted



(c) colormap

Fig. 4: Velocity map of human area.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.

F. Velocity estimation

Our main goal is human segmentation, while the velocity map is also obtained as a by-product. The ground truth and prediction of velocity map of human area is shown in Figure 4, where the tendency of estimated velocities is similar to that of ground truth. This indicates that the velocity estimation branch captures motion information, which is used as cues for segmentation.

V. Conclusion

In this work, we have proposed a two-branch network for dynamic point cloud segmentation, which achieves high accuracy on a dataset for human segmentation. It is able to extract and utilize temporal information via feature propagations between the segmentation and the velocity estimation branches. According to the experimental results, we first conclude that temporal information contained in sequential data is beneficial to segmentation because motion cues can compensate for the sparseness of the input point cloud. Therefore, it is especially helpful for small and far object detection. Secondly, an appropriate increase in the length of sequence can improve the performance by producing more motion cues, while the trade-off between accuracy and computation cost exists.

For future improvement, we would consider how to improve the generalization ability of the model, so that it can be applied to more datasets. Also, we will try to reduce the computation load caused by the increase in the number of frames.

References

[1] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan, "3d object recognition in cluttered scenes with local surface features: a survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 11, pp. 2270–2287, 2014.

[2] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *ArXiv*, vol. abs/1912.12033, 2019.

[3] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[4] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[5] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[6] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[7] W. Kim, M. Tanaka, M. Okutomi, and Y. Sasaki, "Learning-based human segmentation and velocity estimation using automatic labeled lidar sequence for training," *IEEE Access*, vol. 8, pp. 88 443–88 452, 2020.

[8] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3234–3243.

[9] 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 *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9297–9307.

[10] X. Liu, M. Yan, and J. Bohg, "Meteornet: Deep learning on dynamic 3d point cloud sequences," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9246–9255.

[11] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[12] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, "Segflow: Joint learning for video object segmentation and optical flow," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 686–695.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[14] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," *arXiv preprint arXiv:1911.11236*, 2019.

[15] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893.

[16] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 922–928.

[17] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4376–4382.

[18] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[19] W. Kim, M. Tanaka, M. Okutomi, and Y. Sasaki, "Automatic labeled lidar data generation based on precise human model," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 43–49.

[20] ——, "Automatic labeled lidar data generation and distance-based ensemble learning for human segmentation," *IEEE Access*, vol. 7, pp. 55 132–55 141, 2019.

[21] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[22] H. Fan and Y. Yang, "Pointrnn: Point recurrent neural network for moving point cloud processing," *arXiv preprint arXiv:1910.08287*, 2019.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[24] B. K. Horn and B. G. Schunck, "Determining optical flow," in *Techniques and Applications of Image Understanding*, vol. 281. International Society for Optics and Photonics, 1981, pp. 319–331.

[25] D. Nilsson and C. Sminchisescu, "Semantic video segmentation by gated recurrent flow propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6819–6828.

[26] X. Zhu, Y. Xiong, J. Dai, L. Yuan, and Y. Wei, "Deep feature flow for video recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2349–2358.

[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[28] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, "Three-dimensional scene flow," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2. IEEE, 1999, pp. 722–729.

[29] M. Mochimaru, M. Kouchi, N. Miyata, M. Tada, Y. Yoshida, K. Aoki, K. Kawachi, and Y. Motomura, "Dhaiba: Functional human models to represent variation of shape, motion and subjective assessment," SAE Technical Paper, Tech. Rep., 2006.

[30] Miraikan, https://www.miraikan.jst.go.jp, accessed: 2020 March 17.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 28,2021 at 23:56:09 UTC from IEEE Xplore. Restrictions apply.