SDFMap: A Real-time 3D Semantic Mapping for LiDAR Point Cloud

Jian Zhang*, Runhao Luo*, Conghui Geng, Yu Zhou, Yao Yu, Sidan Du, Pei Li

***Abstract*— We present a real-time SDF(Signed Distance Function) based 3D semantic mapping for LiDAR point cloud, which achieves accurate and continuous surface reconstruction result and high semantic segmentation accuracy. Real-time large-scale scene understanding and 3D structure perception are very challenging but important. LiDAR data is sparse and massive and is difficult to use for 3D reconstruction. So we propose a novel line-of-sight algorithm to update implicit surface incrementally and a novel semantic fusion method to segment in real time. Besides, a new network named Frustum Fusion Network(FFNet) is proposed to improve semantic segmentation accuracy based on LiDAR point cloud and RGB images. Our method takes advantage of the information in collection processes to reduce noise in both reconstruction and semantic segmentation. We implemented parallel computation in the reconstruction and semantic fusion process, which achieve a real-time performance. We demonstrate our approach in CARLA dataset, Apollo dataset and our dataset. Compared with the state-of-art mapping methods, our method has a great advantage in terms of both quality and speed, which meets the needs of robotic mapping and navigation.**

I. INTRODUCTION

When robots enter unfamiliar environment, it is very important to perceive the 3D structure and recognize objects in real time. Reconstructing precise and continuous surface in real time allows robots respond accurate and fast. At the same time, fusing semantic information into the 3D map is also a necessary part from the perception standpoint. Building a system which can automatically perform real-time large-scale semantic mapping is very meaningful.

For many robotic applications[1][2], they should always be provided with the precise distance of relevant objects. Among 3D sensing devices, LiDAR (Lighting Detection And Ranging) holds a high measurement accuracy and higher resolution of angles, distance and speed. However, reconstruction based on LiDAR is almost off-line[3][4][5], because LiDAR sensors collect ten thousands of points per frame and it will collect a huge number of points just in several minutes. So an efficient incremental reconstruction method is the key to exploit the high accuracy of LiDAR and avoid the problem of huge amount of data accumulation. On the other hand, sparse online reconstruction[6][7][8][9][10] has always been an important research field because of its many advantages such as lower computation and equipment cost. But in some applications such as self-driving, many objects of interest are possible to be ignored and this situation will lead to serious problem. Semantic information is also a vital aspect of perception which can help robots understand surroundings.

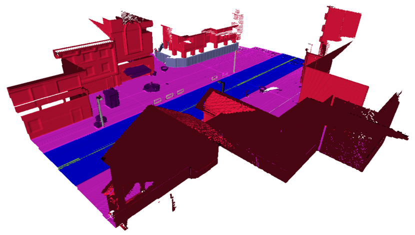
Fig. 1: The result of our system: 3D semantic map

Fusing it into the existing 3D map suffers a variety of limitations. Many Approaches[11][12] just work indoors and others[13][14] use a significant time in semantic fusion. These methods achieve precise segmentation accuracy but are not online.

In this paper, a novel real-time SDF(Signed Distance Field) based 3D semantic mapping framework for LiDAR point cloud is presented, which use a line-of-sight algorithm to assign SDF value, FFNet to parse scene and Bayesian way to fuse semantic 3D map with multiframe in data collection processes. By fusing data over time, our method achieves the 3D surface estimation with subvoxel precision and high accuracy of semantic segmentation. Our line-of-sight algorithm also can update the SDF value of voxel incrementally. Parallel computation is implemented to achieve real-time performance.

The paper makes main contributions as follows,

- We present a new real-time SDF based 3D semantic mapping for LiDAR point cloud. Compared with other methods, our framework can achieve smoother 3D surface estimation with subvoxel accuracy, higher accuracy of semantic segmentation and real time performance.
- We propose a novel line-of-sight algorithm to update the implicit surface incrementally and reduce noise considering LiDAR characteristics.
- We propose a joint 2D-3D segmentation network (FFNet) for scene parsing, which exploit fusing the spatial information and texture information to parse scene.
- We incrementally fuse the 3D semantic map over time using the results of FFNet as input and improve the accuracy of semantic segmentation compared with using single frame data.

The rest of this paper is organized as follows. Related work is included in Sect. II. Sect.III presents our line-of-sight algorithm, FFNet and semantic fusion module. Experimental results and comparisons are shown in Sect. IV. Finally, a discussion and a conclusion are made in Sect. V.

*Indicates equal contribution.

The authors are with School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China.

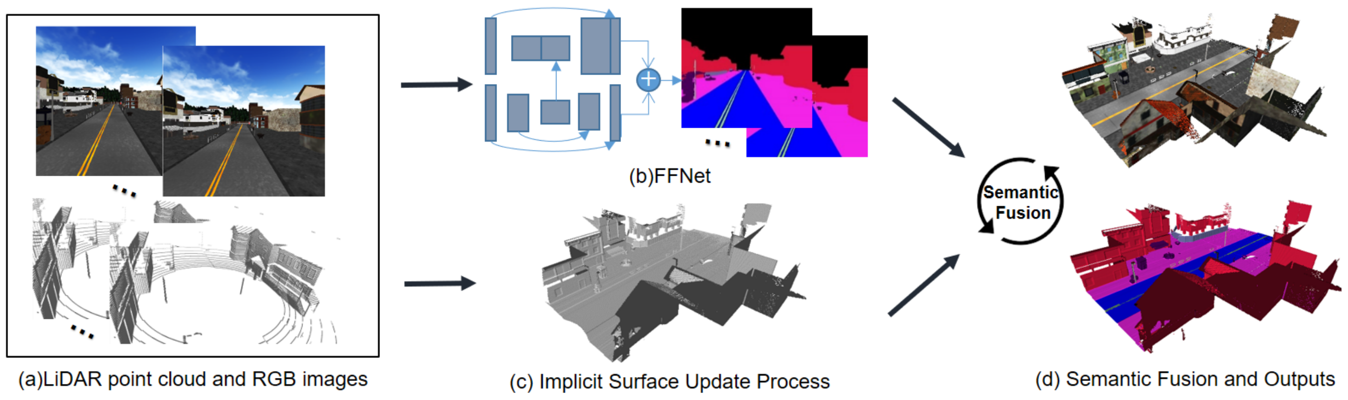
(a)LiDAR point cloud and RGB images
(b)FFNet
(c) Implicit Surface Update Process
Semantic Fusion
(d) Semantic Fusion and Outputs

Fig. 2: Overview of our pipeline. (a) is multi-frame LiDAR point cloud and RGB images as input; (b) is FFNet which outputs semantic results based on LiDAR point cloud and RGB images; (c) is the implicit surface update process; (d) is semantic fusion module and 3D semantic mapping results.

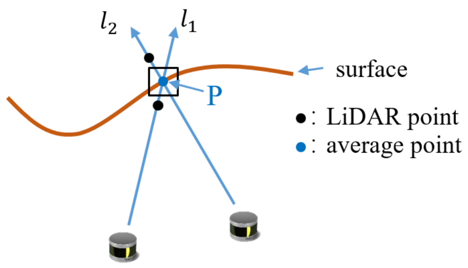

Fig. 3: Implicit surface reconstruction. we use TSDF weighted moving average computation. l_1 and l_2 are different lines of sight. The first update l_1 and the next update l_2 combine to get an average surface point, which is close to the real point.

II. RELATED WORK

3D semantic mapping and surface reconstruction, due to their vast applications in many areas, have long been popular in many research communities, from computer graphics, computer vision to robotics.

A. 3D Reconstruction

In the computer graphics and computer vision fields, researchers proposed many methods to achieve precise surface reconstruction for LiDAR point cloud. Verma et al. [15], Zhou et al. [16] and Poullis et al. [17] created 3D scenes from LiDAR data. They used classification to remove noise, segmentation to separate individual building patches and ground points, and generated mesh models from building patches. These approaches are not generalized well to handle objects with arbitrary shapes since they rely on predefined patterns. Without exception, these methods must be done off-line. KinectFusion [18] uses TSDF (Truncated Signed Distance Function) to model surface using RGBD data. However, this method cannot deal with the LiDAR data.

In robotics field, Hornung et al. [19] proposed OctoMap which uses octree data structure to realize memory-efficient. However, OctoMap lacks detail near the surface and does not have an arbitrary resolution. Lovi et al. [6] and Hoppe et al. [7] realize real-time incremental reconstruction. However, these sparse reconstruction methods can not provide a high precision map, which is important for autonomous driving application.

B. Semantic Mapping

Recently, semantic mapping has been more and more popular. Approaches like [11][12][20] reconstruct 3D semantic model using RGB-D sensors which is easily affected by the environment and limited by the effective distance of the sensor. Sengupta et al. [13], Hane et al. [14] and Kundu et al. [21] realize large-scale semantic mapping. But their methods are off-line and are not incremental. Vineet et al. [22] use stereo fusion to reconstruct semantic scene, which is incremental and near real-time. They use random forest classifier to extract 2D features and volumetric CRF to refine results, which consume significant time in these processes.

III. METHOD

The pipeline of our system is shown in figure 2. The input of our system is the LiDAR point cloud and RGB images. Through registration algorithm, a pose can be estimated and used to map and reconstruct. After obtaining the pose information, voxels will be created only if they are near the surface and within the volume range. Our line-of-sight algorithm updates them incrementally. At the same time, the FFnet will process the current frame data and output the semantic segmentation results. At last, the semantic fusion module will fuse the semantic results over time into reconstructed 3D map and output 3D semantic map.

A. 3D Surface Reconstruction

Current methods[23][18] of integrating depth sensor data into a TSDF are based on depth images. When it comes to LiDAR, it will be different. What LiDAR collects is unorganized point cloud data and we can not directly use projection method to update its implicit surface. We propose a voxel based line-of-sight update algorithm, as shown in Fig. 3, to solve the problem. The line of sight is gotten

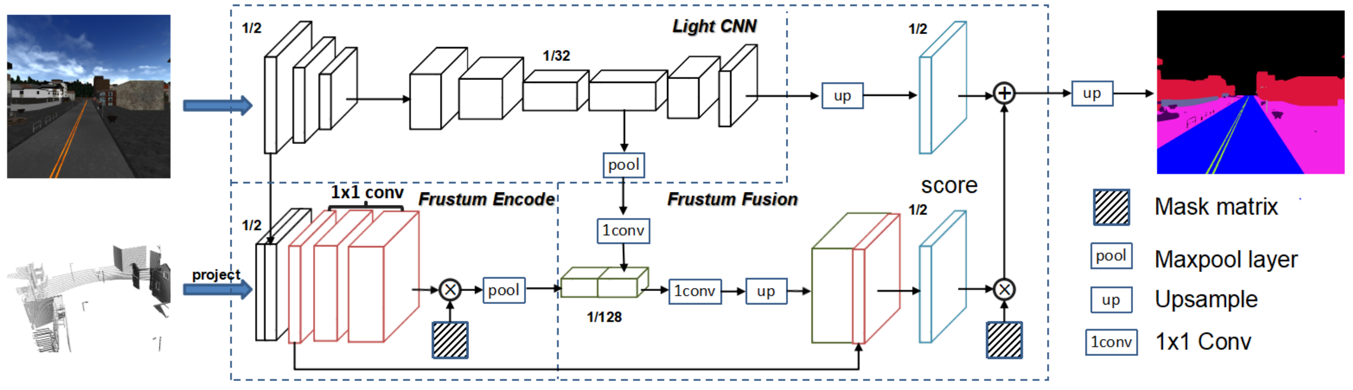

Fig. 4: Network achitecture of our FFNet.

from current sensor posture and LiDAR points P in world coordinate. We get the voxels which the line of sight cross through, update the associated voxel values, fuse them over time and get the continuous implicit surface of an object.

The input to our algorithm is an unorganized 3D point cloud set $P_k^L = \{p_{k,1}^L, p_{k,2}^L, \cdots, p_{k,n}^L\}$ in LiDAR coordinate system $\{L\}$, $k \in Z^+$, where k represents LiDAR frame number, $p_{k,i}^L \in R^3$ indicates ith point in kth frame. LiDAR kth frame's pose in world coordinate system is H_k.

The key idea of our implicit surface update method is to generate the line of sight $\overline{o_k p_{k,i}}$, where o_k is the sensor origin and $p_{k,i}^L$ is the current LiDAR point, and then to find the relevant voxels $V_{p_{k,i}}$ from the line of sight. We transform the point cloud into world coordinate system using k_{th} posture, which contains a 3×3 rotation matrix R_k and 3×1 translation vector T_k.

$$p_{k,i} = \begin{bmatrix} R_k & T_k \end{bmatrix} p_{k,i}^L \tag{1}$$

o_k is equal to T_k and the line of sight is a three dimensional vector, as shown in equation 2.

$$\overrightarrow{o_k p_{k,i}} = p_{k,i} - o_{k,i} \tag{2}$$

We use the line of sight to sweep relevant voxels in space and update their TSDF values and weights. To avoid missing voxels when searching the surrounding voxels of each line of sight. We find maximal axis and take the maximal axis as standard and normalize other two directions to get the normalized direction vector $\hat{\overrightarrow{op_{k,i}}}$, as shown in equation 3.

$$\hat{\overrightarrow{o_k p_{k,i}}} = \frac{\overrightarrow{op_{k,i}}}{max(\overrightarrow{op_{k,ix}}, \overrightarrow{op_{k,iy}}, \overrightarrow{op_{k,iz}})} \tag{3}$$

We then use the line normalized direction vector $\hat{v}$ to find related points $P_{\overline{op_{k,i}}}$ in the front of and behind the original point $O_{k,i}$ respectively, as shown in equation 4.

$$P_{\overline{op_{k,i}}} = O_{k,i} + m \cdot \hat{\overrightarrow{op_{k,i}}} \tag{4}$$

where m is a parameter. In our system, m is set to integers less than 5 in absolute value. Through these relevant points, we get surrounding voxels $V_{k,i}$ corresponding to the point $p_{k,i}$ and $v_{k,i} \in V_{k,i}$.

LiDAR probably collects points from a long distance, which contain more noise compared with close points. We use a dynamic weight to account for noise data far from the sensor. As shown in equation 5, a is a parameter which is related to the size of reconstruction scene. According to our experiment, we set the value of a to 5.

$$w_{v_{k,i}} = \frac{a}{a + ||\overrightarrow{o_k p_{k,i}}||} \tag{5}$$

TSDF is updated as equation 6, 7.

$$W_k(v_{k,i}) = max(W_{k-1}(v_{k,i}) + w_i(v_{k,i}), W_{max}) \tag{6}$$

$$D_k(v_{k,i}) = \frac{W_{k-1}(v_{k,i})D_{k-1}(v_{k,i}) + w_k(v_{k,i})d_k(v_{k,i})}{W_k(v_{k,i}) + w_k(v_{k,i})} \tag{7}$$

where $D_{k-1}(v_{k,i})$, $W_{k-1}(v_{k,i})$ are the TSDF values and their weight of all voxels in k-1 frame as shown in equation 8.

$$\begin{aligned} D_{k-1}(v_{k,i}) &= \frac{\sum w_{k-1}(v_{k,i})d_{k-1}(v_{k,i})}{\sum d_{k-1}(v_{k,i})} \\ W_{k-1}(v_{k,i}) &= \sum w_{k-1}(v_{k,i}) \end{aligned} \tag{8}$$

Voxel's signed distance function is $d_1(v_{k,i})$, $d_2(v_{k,i})$, $\cdots$ $d_n(v_{k,i})$ with corresponding weight $w_1(v_{k,i})$, $w_2(v_{k,i})$, $\cdots$ $w_n(v_{k,i})$.

For parallelization, we use a spatial hashing based data structure to manage the space, which allows us to insert and update voxels in parallel. A memory pool is also created to reduce fragmentation of memory and improve efficiency. We only insert voxels near the zero isosurface and do not waste memory on empty space.

B. *FFNet*

The overall framework of our Frustum Fusion Network is illustrated in Fig. 4. Owing to that we use point cloud data for reconstruction, we have segmentation tasks on both image and point cloud. Naively getting prediction of point cloud from segmented image via projection will leave geometry information unused. Furthermore, CNN architecture often downsample input a lot to reduce time consumption and yield coarse prediction maps missing many small but important details, which is not consistent with accurate geometry information in our reconstruction system. Instead, We proposed a novel Frustum Fusion Network for joint segmentation of image and point cloud. Our network is composed of three parts: a light CNN network to extract feature map, a frustum block encode unit to aggregate information of sparse points in each frustum block, and a fusion unit to combine image and geometry information to make a joint prediction.

Light CNN for image Due to the inefficiency of CNN to restore high resolution prediction, we apply a light weight CNN architecture to achieve high level semantic information. As shown in Fig.4, three 3x3 Conv layers with stride 2 are used to downsample the input image to 1/8 resolution, the output is then feed into PSPNET50 without initial block to get feature map and image score map $\mathbb{S}_p^I$.

Frustum block encode unit Point cloud data is often divided into voxels in Euclidean space for easy processing. However, this will yield many empty voxels and make it hard to build connections between geometry information and image feature especially high level feature. To address this issue, we proposed a frustum block encode unit to aggregate information of points in each frustum block according to 2D image grids. First, we project point cloud to camera coordinate to get a sparse map and a mask matrix M_p, which is a binary matrix indicating if there is a point corresponding to pixel position p. We also augment each point with relative offset and low level image feature. Applying traditional CNN modules on this sparse map will bring a lot of noise, instead we use several 1×1 Conv layers to process each pixel of the projected map in parallel to get encoded local feature map $\mathbb{F}_p^L$. Then We multiply $\mathbb{F}_p^L$ with M_p followed by a Maxpool layer to get global feature $\mathbb{F}_p^G$ for each block, here the size and stride of the Maxpool layer is equal.

Frustum fusion unit In this module, we fuse the information obtained from image and point cloud to get a joint 2D and 3D segmentation result. As shown in Fig. 4, The above global frustum block feature is concatenated with image high level feature and passed through a 1x1 Conv to get joint global feature. The joint global feature is then upsampled and concatenated with local feature map to get a score map $\mathbb{S}_p^P$ for point cloud. The final score map is calculate by:

$$\mathbb{S}_p^F = \mathbb{S}_p^I + M_p \times \mathbb{S}_p^P \tag{9}$$

Then we use a softmax loss with image ground truth to train the network end to end. Prediction for point cloud is then obtained via transform, which is the input to our incremental fusion module described in the next section.

C. *Incremental Semantic Label Fusion*

Output from the FFNet does not exploit the time information and would have segmentation error due to uncertainty of sensors and environments. Fusing the semantic information over time will reduce noise and increase accuracy of segmentation. The semantic information is fused in a Bayesian way[26][19]. We denote the class distribution of a voxel $v_{k,i}$ as $l_{v_{k,i}}(l_{v_{k,i}} = l^1_{v_{k,i}}, l^2_{v_{k,i}} \cdots l^j_{v_{k,i}} \cdots$, j represents jth class, $v_{k,i}$ represents kth frame's ith point). $l^j_{v_{k,i}}$ has two states $\{yes, no\}$ and $Z_k = \{I_k, S_k\}$ represents kth frame's observation, which includes images I_k and relevant voxels' distance S_k from LiDAR sensor. We assume that semantic results become less credible at a further distance and $P(l^j_{v_{k,i}}|Z_k) \sim P(l^j_{v_{k,i}})$. Semantic information $P(l_{v_{k,i}}|Z_{1:k})$ is updated as equation 10 within a valid distance.

$$P(l_{v_{k,i}}|Z_{1:k}) = \\ [1 + \frac{1-P(l_{v_{k,i}}|Z_k)}{P(l_{v_{k,i}}|Z_k)} \frac{1-P(l_{v_{k,i}}|Z_{1:k-1})}{P(l_{v_{k,i}}|Z_{1:k-1})} \frac{P(l_{v_{k,i}})}{1-P(l_{v_{k,i}})}]^{-1} \tag{10}$$

where $P(l_{v_{k,i}})$ is a prior probability. The common assumption of a uniform prior probability leads to $P(l_{v_{k,i}}) = 0.5$. With

$$L(l_{v_{k,i}}) = log[\frac{P(l_{v_{k,i}})}{1-P(l_{v_{k,i}})}], \tag{11}$$

the final update method is

$$L(l_{v_{k,i}}|Z_{1:k}) = max(min(L(l_{v_{k,i}}|Z_{1:k-1}) + L(l_{v_{k,i}}|Z_k), l_{max}), l_{min}). \tag{12}$$

IV. EXPERIMENTS AND COMPARISONS

A. *Dataset*

We verify the effectiveness of our method for both 3D semantic segmentation and reconstruction in three datasets.

CARLA Dataset. We generate a virtual dataset with CARLA[24] simulation platform, which is built for realism in the rendering and physics simulation and allows for flexible configuration of the agent's sensor suite. In our experiment, we set cameras with 1024×1024 resolution to get RGB and semantic segmentation images. A virtual 64-line 10HZ LiDAR sensor is also added to obtain point cloud. With sensors attached to a auto-piloted car in the virtual city, We simulate and collect sensor data with corresponding poses at 10 fps. Totally 900 frames data is generated and every $10th$ frame of the sequence is selected to yield a 90 frames test set.

Apollo Dataset. We also evaluate our approach on Apollo Dataset[25], which is challenging for real-world large-scale scenes mapping. We use the Record005 sequence data of Road02 portion.

VLP-16 Dataset. The VLP-16 dataset is collected by ourself. Our system's operating frequency is 10HZ, which includes a VLP-16 LiDAR, two cameras and a laptop.

B. *3D Reconstruction Results*

1) Qualitative Results: We show some qualitative results for our reconstruction approach and achieve a good result on both virtual dataset and real datasets. As shown in Fig. 5, our approach not only get good results on large objects such as road and buildings but also have the ability to accurately recover small object such as street light, boxes and so on.These results prove that the line-of-sight algorithm is very effective on LiDAR point cloud.

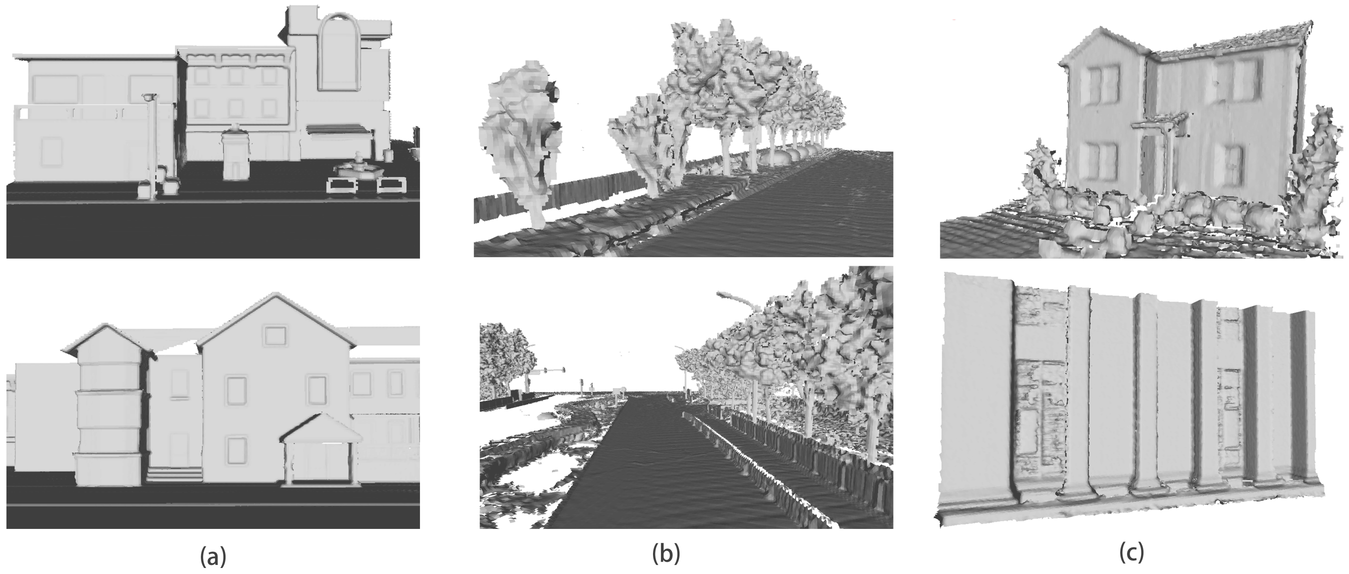

Fig. 5: Surface reconstruction results. Each column corresponds to different datasets. (a)is from CARLA simulator [24]. (b) is from Apollo data set[25]. (c) is VLP-16 dataset, which is collected from our system.

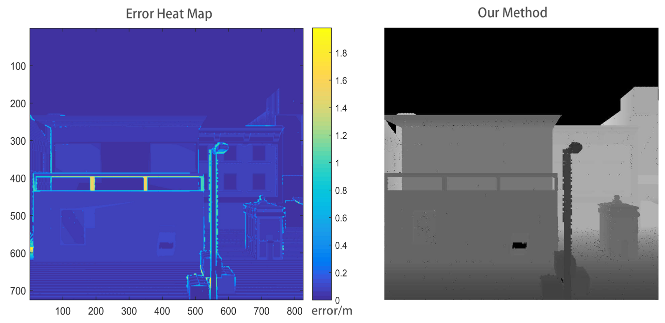

Fig. 6: Reconstruction error of our method.

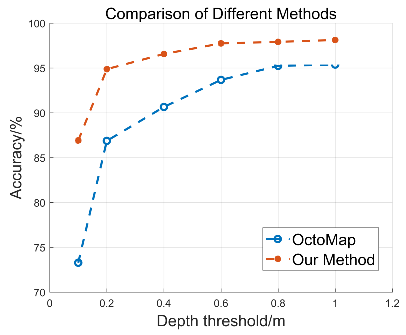

Fig. 7: Qualitative comparison of reconstruction result between our methods and OctoMap[19] under different depth threshold.

TABLE I: **Comparison on CARLA dataset**: FFNet-PF denote the results rendered after our incremental probability fusion.

Method	Build	Fence	Poles	Roadl	Roads	Sidew	Walls	mIoU
ICNET[27]	97.56	52.26	76.05	82.67	99.26	**98.61**	96.60	86.14
FFNet	97.88	54.34	78.23	83.33	99.25	98.34	96.80	86.88
FFNet-PF	**99.18**	**64.50**	**83.79**	**87.55**	**99.52**	98.41	**98.01**	**90.14**

2) Performance Comparison: To evaluate the errors introduced by line-of-sight algorithm, we reconstruct a scene in CARLA simulator as shown in Fig. 6. The error of 3D reconstruction mainly appears at the edge of small objects such as railing and street lights. As the approach of Sengupta et al. [13], we render depth images from our reconstructed 3D model. By comparing them with the ground truth, error heat maps can be calculated. We think one pixel is accurate if the error is less than a fixed threshold and we can get the accuracy of an image. When the depth threshold is 0.2m and 0.1m, our approach achieves 94.87% and 86.91% accuracy respectively in average.

We compared our approach with OctoMap[19]. OctoMap uses a discrete cut-off probability and its precision is limited by the minimum voxel size. Compared with OctoMap, one advantage of our method is that we can reach the subvoxel precision. We use the same accuracy evaluation method. When the depth threshold is 0.2m and 0.1m, OctoMap achieves 86.85% and 73.28% accuracy respectively in average. As shown in Fig. 7, our approach outperforms OctoMap in all cases.

C. Semantic Mapping Results

We quantitatively evaluate the accuracy of our semantic reconstruction result on our simulated CARLA dataset. Our FFNet cost 35ms on average to infer each frame on simulated dataset. The prediction result of our fusion system is rendered from 3D semantic map. We use pixel-wise interaction over reunion (IoU) as our metric. For a fair comparison, pixels not included in rendered map are not calculated. We compared our system with real-time 2D approach of Zhao et al. [27]. The results are summarised in table I and visualized comparisons is illustrated in Fig. 8. We first observe that our joint segmentation network utilizing point cloud data achieve a slight improvement on poles and fence class against

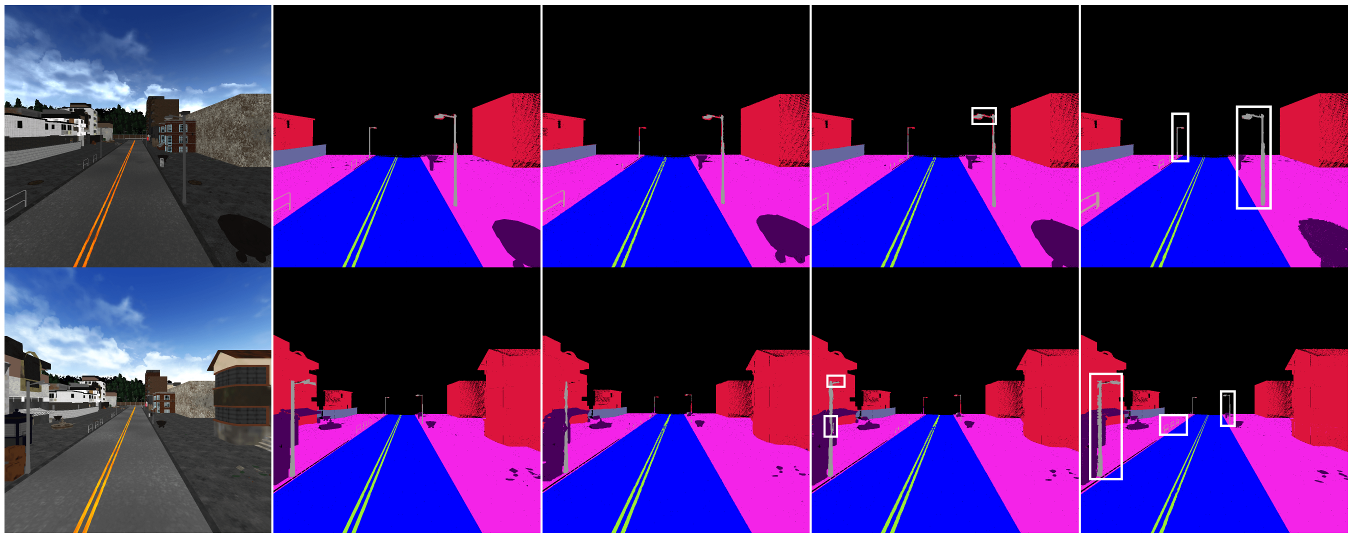

Fig. 8: Visual results on CARLA. Columns from left to right are RGB image, ground truth, ICNET[27] result, FFNet result and FFNet with probability fusion result. We ignore points not included in rendered map.

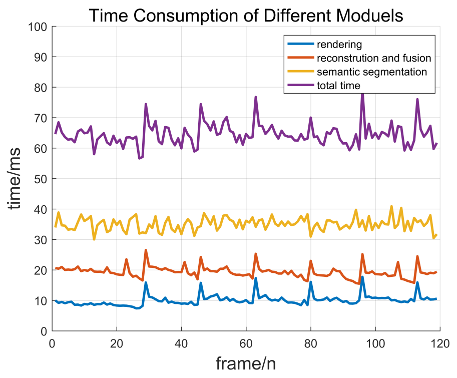

Fig. 9: Time Consumption of different modules, including rendering, FFNet, reconstruction and fusion. The total time is about 65ms in average.

[27] owing to the advantage of geometry information. Our approach of joint segmentation network with incremental probability fusion get better performance upon all classes except sidewalks and achieve a improvement of 4.0% over [27] on mIoU. As we can see in Fig. 8, many blurred boundaries are eliminated via our incremental semantic fusion approach and the outlines of thin objects improve a lot. The reason of the performance drop on sidewalks class is that there are many objects on sidewalks, which will bring a long time occlusion problem. The above results demonstrate the effectiveness of our incremental semantic fusion system and the ability to reconstruct a highly precise semantic map.

D. Real-Time Performance

Real-time performance means all of our software modules should be completed before the next frame's data coming. In our system, LiDAR's operating frequency is 10HZ and we need to process data within 100 ms per frame. VLP-16 LiDAR collects about 20k points per frame. So we fix the points number per frame to 20k In table II(a). According to the experiment in table II(a), we set the voxel size to 0.1m for the balance between speed and quality. As shown in table II(b), our system is robust when points number changes. Setting the voxel size and points number to 0.1m and 20k, the total time per frame is about 65ms in average. Time spending on the semantic segmentation, reconstruction and rendering is about 35ms, 20ms and 10ms in average, as shown in Fig. 9 . Finally, our system consume less than 0.1s per frame and achieve real-time performance.

(a) Different Voxel Size

Voxel size/m	time/ms
0.2	42
0.15	52
0.10	65
0.08	88
0.06	122
0.04	136
0.02	153

(b) Different Points Number

Points Number(per frame)	time/ms
5k	37
10k	45
20k	65
40k	113
60k	152
80k	213
100k	265

TABLE II: Table II(a) 20k points per frame, Table II(b) Voxel size is 0.1m

V. CONCLUSION

In this paper, we propose a real-time SDF based 3D semantic mapping framework for LiDAR point cloud. A line-of-sight algorithm is used to solve the large noise in the LiDAR data, reconstruct smooth surfaces. A joint 2D-3D segmentation network FFNet is proposed for scene parsing and an incremental fusion method is used to improve semantic segmentation result. Our method provides real-time performance to meet robotics needs and achieves a subvoxel precision surface reconstruction result. Compared with other state-of-art mapping methods, our method has a great advantage in terms of both quality and speed.

References

[1] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. R. Bradski, "Self-supervised monocular road detection in desert terrain," in *Robotics: Science & Systems Ii, August, University of Pennsylvania, Philadelphia, Pennsylvania, Usa*, 2006.

[2] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, and C. Geyer, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, p. 425–466, 2008.

[3] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 3, p. 29, 2013.

[4] S. Giraudot, D. Cohen-Steiner, and P. Alliez, "Noise-adaptive shape reconstruction from raw point sets," in *Computer Graphics Forum*, vol. 32, no. 5. Wiley Online Library, 2013, pp. 229–238.

[5] H. Lin, J. Gao, Y. Zhou, G. Lu, M. Ye, C. Zhang, L. Liu, and R. Yang, "Semantic decomposition and reconstruction of residential scenes from lidar data," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 66, 2013.

[6] D. Lovi, N. Birkbeck, D. Cobza, and M. Jägersand, "Incremental free-space carving for real-time 3d reconstruction," 01 2011.

[7] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof, "Incremental surface extraction from sparse structure-from-motion point clouds," in *British Machine Vision Conference*, 2013.

[8] A. Romanoni and M. Matteucci, "Incremental reconstruction of urban environments by edge-points delaunay triangulation," pp. 4473–4479, 2016.

[9] Y. Ling and S. Shen, "Building maps for autonomous navigation using sparse visual slam features," in *Ieee/rsj International Conference on Intelligent Robots and Systems*, 2017, pp. 1374–1381.

[10] A. Romanoni, D. Fiorenti, and M. Matteucci, "Mesh-based 3d textured urban mapping," in *Ieee/rsj International Conference on Intelligent Robots and Systems*, 2017, pp. 3460–3466.

[11] X. Li and R. Belaroussi, "Semi-dense 3d semantic mapping from monocular slam," 2016.

[12] J. Mccormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," pp. 4628–4635, 2016.

[13] S. Sengupta, E. Greveson, A. Shahrokni, and P. H. S. Torr, "Urban 3d semantic modelling using stereo vision," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 580–585.

[14] C. Hane, C. Zach, A. Cohen, R. Angst, and M. Pollefeys, "Joint 3d scene reconstruction and class segmentation," in *Computer Vision and Pattern Recognition*, 2013, pp. 97–104.

[15] V. Verma, R. Kumar, and S. Hsu, "3d building detection and modeling from aerial lidar data," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2. IEEE, 2006, pp. 2213–2220.

[16] Q.-Y. Zhou and U. Neumann, "2.5 d dual contouring: A robust approach to creating building models from aerial lidar point clouds," *Computer Vision–ECCV 2010*, pp. 115–128, 2010.

[17] C. Poullis and S. You, "Automatic reconstruction of cities from remote sensor data," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2775–2782.

[18] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*. IEEE, 2011, pp. 127–136.

[19] A. Hornung, M. W. Kai, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[20] A. Hermans, G. Floros, and B. Leibe, "Dense 3d semantic mapping of indoor scenes from rgb-d images," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 2631–2638.

[21] A. Kundu, Y. Li, F. Dellaert, F. Li, and J. M. Rehg, *Joint Semantic Segmentation and 3D Reconstruction from Monocular Video*. Springer International Publishing, 2014.

[22] V. Vineet, O. Miksik, M. Lidegaard, M. Nießner, S. Golodetz, V. A. Prisacariu, O. Kähler, D. W. Murray, S. Izadi, and P. Pérez, "Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 75–82.

[23] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 303–312.

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[25] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The apolloscape dataset for autonomous driving," *arXiv: 1803.06184*, 2018.

[26] M. Yguel, O. Aycard, and C. Laugier, *Update Policy of Dense Maps: Efficient Algorithms and Sparse Representation*. Springer Berlin Heidelberg, 2008.

[27] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," *arXiv preprint arXiv:1704.08545*, 2017.