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

Jian Zhang*, Anyi Rao*, Pei Li, Conghui Geng, Yao Yu, Yu Zhou, Sidan Du

***Abstract*— We present a real-time SDF based 3D mapping framework for LiDAR point cloud, which achieves accurate and continuous surface reconstruction result and good visualization performance. The problem is challenging since LiDAR data is sparse and massive. So we propose a new surface grid data structure to discretize space for memory efficiency and a novel line of sight algorithm to update implicit surface incrementally. Our method takes the advantage of the information in collection processes to reduce noise and register point cloud. We implemented parallel computation in the reconstruction process and achieve a real-time performance. Compared with the state-of-art mapping methods, our method has a great advantage in terms of both quality and visualization, which meets the needs of robotic mapping and navigation.**

I. INTRODUCTION

Real-time large-scale 3D mapping and reconstruction for LiDAR point cloud with precise and continuous surfaces is a very popular but challenging problem. This allows robots to plan precise path, and respond faster and more accurate to the surrounding environment.

Among 3D sensing devices, LiDAR (Lighting Detection And Ranging) holds a high measurement accuracy and higher resolution of angles, distances and speeds. Its data collection is independent of weather and light, with large effective distance. Recently, some LiDAR sensors collect ten thousands of points one time and many surface reconstruction methods [1][2] cannot process the data in real time. These methods achieve precise surface reconstruction but are not online which is unacceptable for robotics applications. On the other hand, due to the inaccurate external parameters calculation, sensor errors and noise, the point cloud stratifies after several scans for a surface. Some researchers use probability model to reduce the noise of sensor data and achieve real-time performance, such as Octomap [3]. However, Those methods have a discrete cut-off probability and their precision is limited to the minimum voxel size.

In this paper, a novel real-time SDF(Signed Distance Field) based 3D mapping framework for LiDAR point cloud is presented, which assign SDF value to the voxels near surface and use line of sight algorithm to get the estimated surface position. Our method achieves subvoxel precision since the surface is found as a zero-crossing. We take the advantage of the pose information of each point cloud frame to update related voxels. Because of this, our method has a great advantage in denoising. Our line of sight algorithm also can update the SDF value of voxel incrementally. Parallel computation is implemented to achieve real-time performance.

*Indicates equal contribution.

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

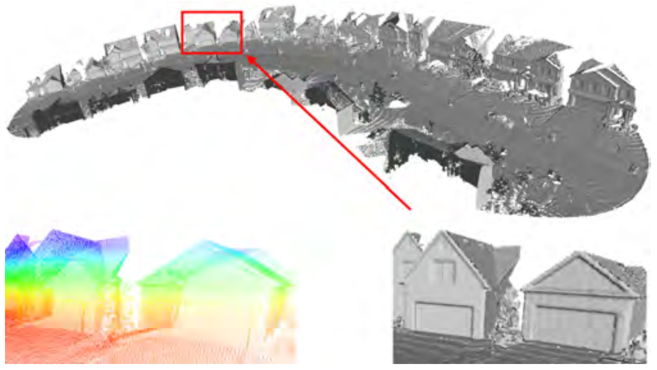

Fig. 1. The first row is a large-scale reconstruction result of our proposed SDFMap. The second row are point cloud and detail of reconstruction respectively.

The paper makes main contributions as follows,

- We present a new real-time SDF based 3D mapping framework for LiDAR point cloud which can achieve both real-time performance and subvoxel accuracy of surface estimation. Compared with other methods, our framework can achieve better visualization performance.
- We propose a novel line of sight algorithm to update the implicit surface incrementally and reduce noise considering LiDAR characteristics.
- We implement parallel computation to update voxels faster.

The rest of this paper is organized as follows. Related work is included in Sect. II. Sect. III formulates the problem formally, explains the notation we used and presents our surface reconstruction algorithm. Experimental results and comparisons are shown in Sect. IV. Finally, a discussion and a conclusion are made in Sect. V.

II. RELATED WORK

3D mapping and surface reconstruction from LiDAR point cloud, due to its vast applications in many areas, have long been an active topic in many research communities, from computer graphics, computer vision, to robotics. In the computer graphics and computer vision fields, researchers proposed many methods to achieve precise surface reconstruction for LiDAR point cloud. Verma et al. [1], Zhou et al. [2] and Poullis et al. [4] 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

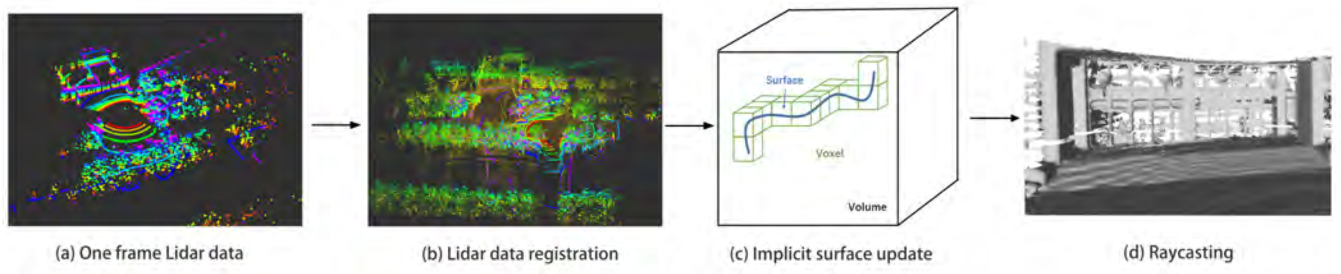

Fig. 2. Overview of our pipeline from point cloud to a raycasted 3D scene. (a) is one frame LiDAR point cloud as input; (b) is multi-frame point cloud after LiDAR registration; (c) is the surface grid data structure and implicit surface update process; (d) is real-time reconstruction result.

approaches are not generalized well to handle objects with arbitrary shapes since they rely on predefined patterns. To deal with arbitrary shapes, Zhou and Neumann [5] extended dual contouring [6] into a 2.5D method and produced 3D models composed of arbitrarily shaped roofs and vertical walls connecting them. Although it holds a high level of details with residential scenes containing roofs, it is hard to deal with other complex scenes, such as large-scale street scenes. Without exception, these methods must be done off-line. Recently, more novel surface reconstruction methods [7][8][9] were proposed. Although these methods are able to reconstruct surface precisely, they require a lot of complicated computation and cannot be used in real time. KinectFusion [10] uses TSDF (Truncated Signed Distance Function) to model surface using RGBD data. However, this method cannot deal with the LiDAR data.

In robotics field, 2D mapping [11] was initially proposed for mobile robot perception and navigation. But the 2D map loses a lot of information and extending 2D map to 3D naively leads to many problems, such as huge memory consumption and low processing speed. Hornung et al. [3] 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. Ramos et al. [12] proposed Hilbert maps and Fridovich et al. [13] proposed AtomMap to deal with these problems. However, these methods have limited accuracy and poor visualization performance to guarantee real-time performance. To solve these problems better, we propose a novel SDF based 3D mapping framework for LiDAR point cloud.

III. SURFACE RECONSTRUCTION

LiDAR collects hundreds of thousands of points per second, or even more. There is a huge amount of cloud data to compute, which brings difficulties to real-time surface reconstruction under the guarantee of high reconstruction accuracy. There are noise, misaligned scans, missing data in LiDAR data. This results in stratification and holes on point cloud, which causes difficulties in reconstructing its surface.

The pipeline of our system is shown in figure 2. The first step is to take current frame LiDAR point cloud as input. Secondly, through LiDAR data registration algorithm, a pose can be estimated and used to map and reconstruct. The above two steps' output will be the input of the third step. 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 last, the reconstructed surface is raycasted for real time visualization.

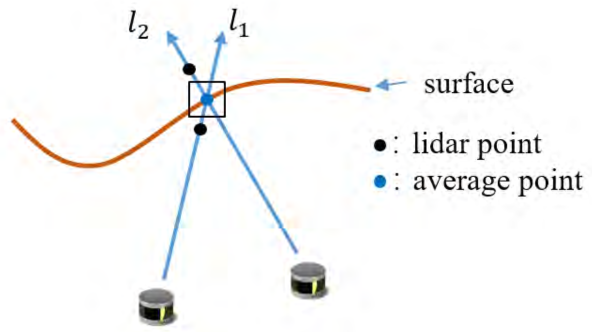

Fig. 3. Implicit surface reconstruction. we use TSDF weighted moving average computation. The first update l_1 and the next update l_2 combine to get an average surface point, which is close to the real point.

A. Sparse Point Cloud TSDF Update

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 figure 3, to solve the problem. The line of sight is gotten 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 many times 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, and pose H_k corresponds to kth frame, $p_{k,i}^L \in R^3$ indicates one point in each frame. The TSDF value is calculated from current frame's LiDAR point P_k^L, and is fused with previous $k-1$ frames, as shown in equation 1.

$$D_k(x) = \frac{W_{k-1}(x)D_{k-1}(x) + w_k(x)d_k(x)}{W_k(x) + w_k(x)} \tag{1}$$

where $D_{k-1}(x)$, $W_{k-1}(x)$ are the TSDF values and their weight of all voxels in k-1 frame as shown in equation 2.

$$D_{k-1}(x) = \frac{\sum w_{k-1}(x) d_{k-1}(x)}{\sum d_{k-1}(x)} \tag{2}$$
$$W_{k-1}(x) = \sum w_i(x)$$

where voxel's signed distance function is $d_1(x)$, $d_2(x)$, $\cdots$ $d_n(x)$ with corresponding weight $w_1(x)$, $w_2(x)$, $\cdots$ $w_n(x)$.

B. Surface Grid

To characterize the implicit surface of an object, we need to compute all the TSDF values around the object. We uniformly divide the space into voxels with the same size and record their TSDF values and weights.

A commonly used way is space voxelization [10], which is to initialize all the reconstructed areas. It wastes massive space in large-scale reconstruction. Our approach only creates voxels in a certain area near the surface to solve the problem, denoted in this paper as surface grid. The key idea of our implicit surface update method is to generate the line of sight $\overline{op_{ki}}$, where o is the sensor origin and $p^L_{k,i}$ is the current LiDAR point, and then to find the relevant voxels from the line of sight, as shown in figure 4. 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, as explained in subsection III-D.

$$p_{k,i} = \begin{bmatrix} R_k & T_k \end{bmatrix} p^L_{k,i} \tag{3}$$

For each LiDAR point $p_{k,i}$, let $O_{k,i}$ be the original point of the line of sight $\overline{op_{ki}}$, and $O_{k,i} = T_k$. The line of sight is a three dimensional vector, as shown in equation 4.

$$\overline{op_{ki}} = p_{k,i} - O_{k,i} \tag{4}$$

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 obtain relevant points in the most dramatic changing direction, as shown in figure 5. We find maximal axis and take the maximal axis as standard and normalize other two directions to get the normalized direction vector $\hat{\overline{op_{ki}}}$, as shown in equation 5.

$$\hat{\overline{op_{ki}}} = \frac{\overline{op_{ki}}}{max(\overline{op_{ki}}_x, \overline{op_{ki}}_y, \overline{op_{ki}}_z)} \tag{5}$$

We then use the line normalized direction vector $\hat{v}$ to find related points $P_{\overline{op_{ki}}}$ in the front of and behind the original point $O_{k,i}$ respectively, as shown in equation 6.

$$P_{\overline{op_{ki}}} = O_{k,i} + m \cdot \hat{\overline{op_{ki}}} \tag{6}$$

where m is a parameter. In our system, m is set as 5. Through these relevant points, we get surrounding voxels corresponding to the point $p_{k,i}$.

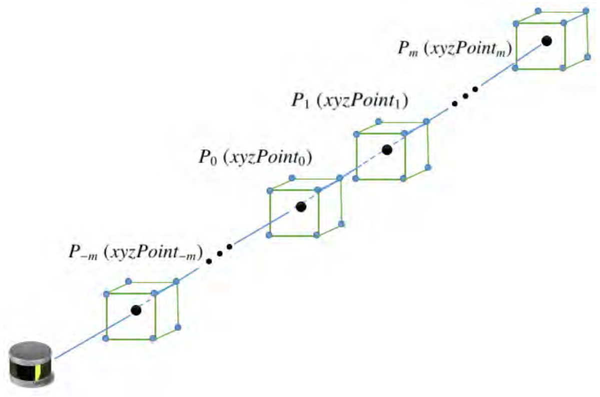

Fig. 4. Generate line of sight. The m points and their coordinates in the front of and behind the point P_0.

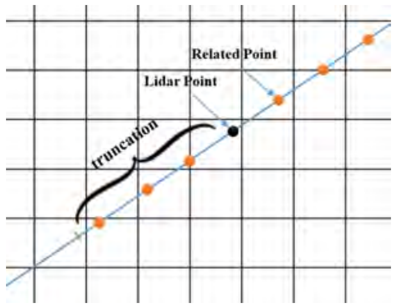

Fig. 5. For example, We get m=3 points and their coordinates in the front of and behind the point. Because $\frac{\partial x}{\partial t} > \frac{\partial y}{\partial t}$. We normalized our ray vector with $\frac{\partial x}{\partial t}$, so we get $2m$ points along x axis rather than y axis.

C. Implicit Surface Parallel Update

Noting the huge amount of LiDAR points and the large computation, we use parallel computation to speed up and achieve real-time reconstruction result. TSDF weighted moving average computation is not an independent process since different lines of sight are likely to index the same voxel. We propose to decompose TSDF value indexing and fusing.

At the first step, we independently calculate the voxel index corresponding to different lines of sight. Points in one frame generates lines of sight $\overline{op_{k1}}$, $\overline{op_{k2}}$, $\cdots$, $\overline{op_{kn}}$ with $(p_{k,1}, d_{k,1}, w_{k,1})$, $(p_{k,2}, d_{k,2}, w_{k,2})$, $\cdots$, $(p_{k,n}, d_{k,n}, w_{k,n})$. After we compute out current frame cloud P_k's TSDF value, we fuse point cloud P_k with the previously calculated 1st to $(k-1)$th frame TSDF values, and we get the current kth frame TSDF.

D. LiDAR Point Cloud Registration

LiDAR points are received at different times, which results in distortion in the point cloud. It is assumed that LiDAR linear velocities are continuous and smooth over time, without abrupt changes. With this assumption, the undistorted point cloud is computed using a linear interpolation same as J. Zhang et al. [14]. We extract different frames' feature points on planar surfaces and sharps edges, and match the feature points to planar surface patches and edge segments respectively. LiDAR sensor motion R_k and T_k are estimated

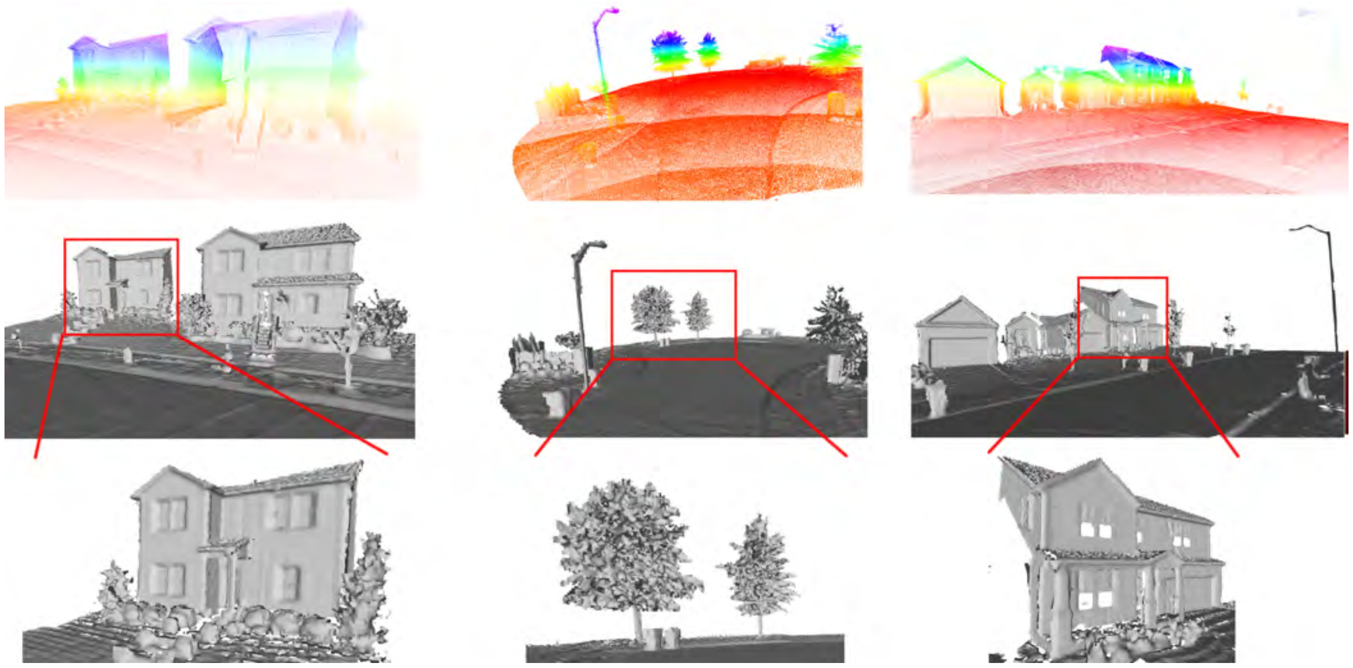

Fig. 6. Surface reconstruction results. From top to bottom: point clouds, surface reconstruction results and their details. Each column corresponds to different environment.

through matching different frame's feature points. Therefore, we use sensor motion map and register each frame's point cloud in the world coordinates.

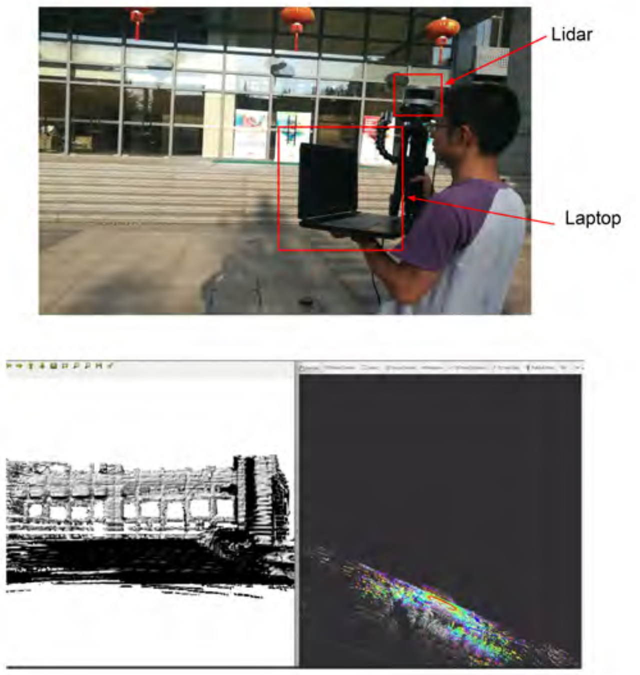

Fig. 7. Our system setup and real-time reconstruction. Our laptop reconstructs a building in real-time. In the laptop screen, the point cloud is on the right, while the surface is on the left. We use a go-pro camera to compare our result with images.

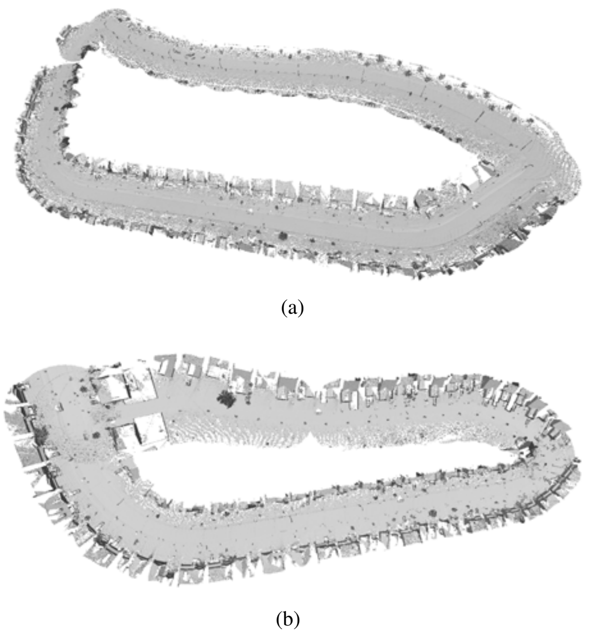

Fig. 8. Large-scale reconstruction results. we reconstruct two loop street (a) containing about 30 houses and (b) containing about 50 houses.

E. LiDAR Raycasting

A GPU-based raycaster generates views of the implicit surface. Each GPU thread walks a single ray and renders a single pixel in the output image in parallel. Given a starting position and direction of the ray, each GPU thread traverses voxels along the ray and extracts the position of the implicit surface by observing a TSDF value zero-crossing. We implement a simple linear interpolation given the trilinearly

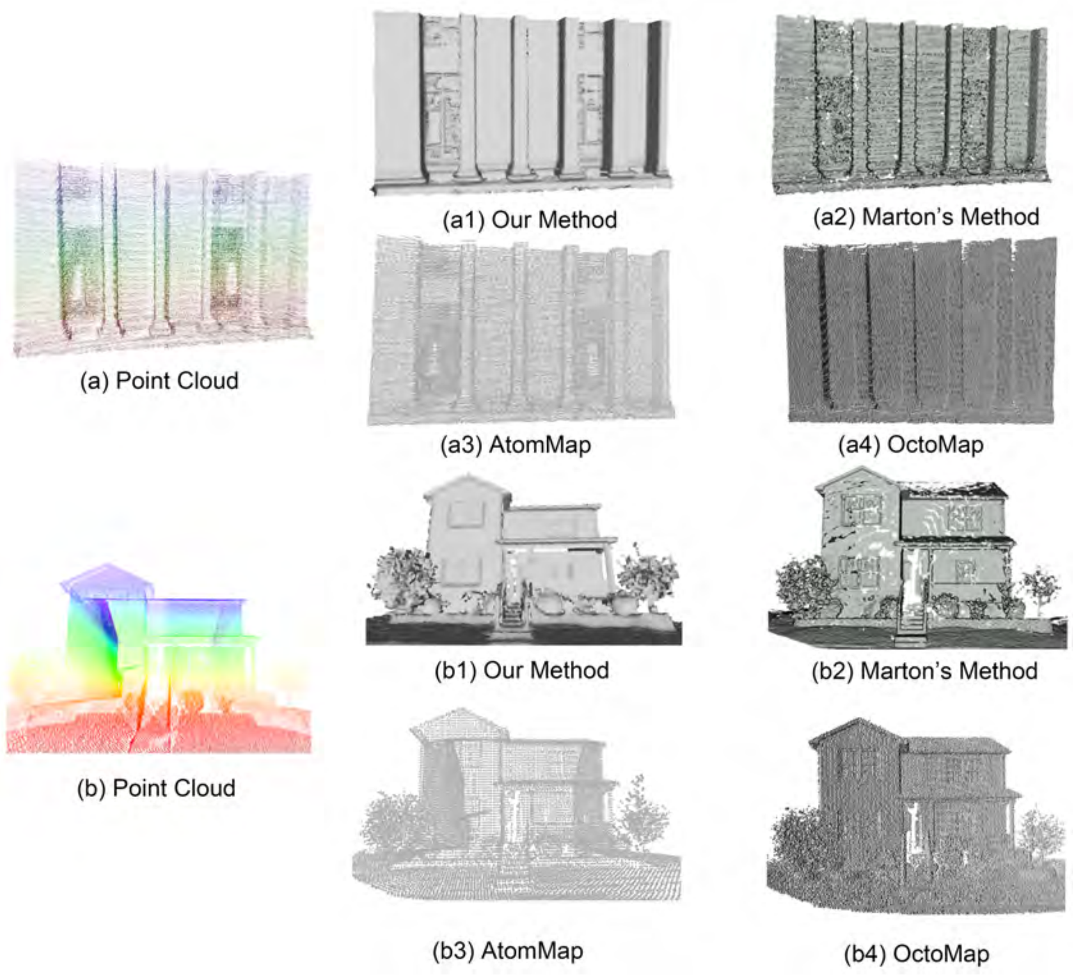

Fig. 9. Qualitative comparison of reconstruction result between several methods. Note that Marton's method, AtomMap and OctoMap are using the existing open source code.

sampled points either side of the zero-crossing to compute the final surface intersection point. The gradient is orthogonal to the surface interface. We compute surface normal from the derivative of the TSDF at the zero-crossing and compute the angle between the normal and our setting light to render the surface. Therefore, each GPU thread that finds a ray/surface intersection can calculate a single interpolated vertex and normal, which can be used for lighting calculations on the output pixel and rendering the surface [15].

IV. EXPERIMENTS AND COMPARISONS

Our system is illustrated in figure 7, including a VLP-16 LiDAR and a laptop. During experiments, the algorithms processing the LiDAR data run on a laptop computer with 2.5GHz quad cores, 8GB memory and 6GB GPU memory, on robot operating system (ROS) in Linux.

A. Dataset

We use two datasets to compare our algorithm and the other three state-of-art algorithms. One is Lin et al's dataset [9], called SDR dataset; the other comes from our collected dataset, called VLP-16 dataset. The SDR dataset is dense, and every frame composes of about ten-thousand points. The VLP-16 dataset is sparse, and every frame is consisted of about two-thousand points.

B. Reconstruction Results

The reconstruction results are shown in figure 6. We achieve a good result on reconstructing residential houses composed of several gables and box structure at different scales and location. The detailed surrounding environment (such as trees and lamps) also achieves a good result. An overview of our large-scale reconstruction is shown in figure 8.

C. Comparison of Reconstruction Performance

We compared our approach with state-of-art algorithms: Armin Hornung et al.'s OctoMap [3], Fridovich et al.'s AtomMap [13] and Marton et al.'s method [16]. Currently, the most commonly used 3D representation is OctoMap, which uses the probabilistic representation to reduce noise. 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

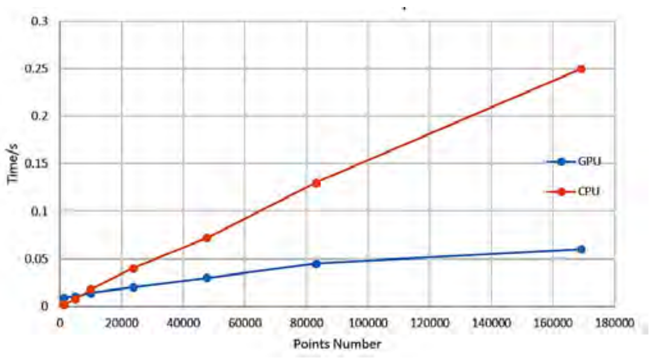

Fig. 10. GPU and CPU time comparison

reach the subvoxel precision, so we have better visualization performance. As shown in figure 9, we use the same voxel size in two methods. Besides, in our method, voxels are fused according to their weights which is very helpful to eliminate noise.

As for AtomMap, Fridovich et al. also acknowledge SDF's promising performance in on-line mapping and planning. AtomMap declares that it uses atoms which is tangent to the surface to represent the map and represents surface more accurately than OctoMap. However, AtomMap's resolution is limited by the atom's size. Although the AtomMap supports a SDF mode, it is more complex than our method when actually used. Compared with the AtomMap, our method is easier to get a continuous surface and has better visualization performance than AtomMap which is shown in figure 9.

We also used Zoltan-Csaba Marton et al.'s provided method on PCL library to perform the evaluation using two datasets. Figure 9 shows the qualitative evaluation on houses varying at surface complexity and scan completeness. We can find that our algorithm reconstructs smoothly and leads to few hole. Zoltan-Csaba Marton et al.'s method smooths reconstruction results according to the angle α between points' normals with typical value 45°. The method needs small angle α to ensure smoothness, but this causes holes in reconstruction since points with small angle α cannot form a triangle. There is a trade-off between holes and smoothness. LiDAR points are quite noisy, sometimes even has stratification. It is hard for the method to achieve both fewer holes and smoothness when handling LiDAR data.

D. Real Time Performance

Our surface reconstruction algorithm processes one million LiDAR points per second. It is very fast and efficient, noting that we only use a normal laptop. Through the use of GPU, we greatly improve the speed of our system to achieve real-time speed and it is shown in figure 10. As the number of points increases, our method maintains a good performance. Noting that a 16-line LiDAR, such as Velodyne VLP-16, generates 20 thousand points per frame and a 64-line LiDAR generates over one hundred thousand points, our approach has an advantage in reconstructing large-scale point cloud.

V. CONCLUSION

In this paper, we propose a real-time SDF based 3D mapping framework for LiDAR point cloud. Implicit surface reconstruction method is used to solve the large noise in the LiDAR data, reconstruct smooth surfaces and smooth joining parts of cloud point. Our algorithm provides real-time performance to meet robotics needs and achieves a subvoxel precision surface reconstruction result. It is capable of completing the incomplete part according to current reconstruction result. Compared with other state-of-art mapping methods, our method has a great advantage in terms of both quality and visualization.

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] Q.-Y. Zhou and U. Neumann, "2.5 d building modeling with topology control," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2489–2496.

[6] T. Ju, F. Losasso, S. Schaefer, and J. Warren, "Dual contouring of hermite data," in *ACM transactions on graphics (TOG)*, vol. 21, no. 3. ACM, 2002, pp. 339–346.

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

[8] 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.

[9] 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.

[10] 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.

[11] A. Elfes, "Occupancy grids: A probabilistic framework for robot perception and navigation," *PhD thesis, Carnegie-Mellon University*, 1989.

[12] F. Ramos and L. Ott, "Hilbert maps: scalable continuous occupancy mapping with stochastic gradient descent," in *Robotics: Science and Systems*, 2015, pp. 1717–1730.

[13] D. Fridovich-Keil, E. Nelson, and A. Zakhor, "Atommap: A probabilistic amorphous 3d map representation for robotics and surface reconstruction," in *IEEE International Conference on Robotics and Automation*, 2017.

[14] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, 2017.

[15] S. O. R. Fedkiw and S. Osher, "Level set methods and dynamic implicit surfaces," *Surfaces*, vol. 44, p. 77, 2002.

[16] Z. C. Marton, R. B. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 3218–3223.