Real-time Large-scale Surface Reconstruction of Lidar Point Cloud

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

***Abstract*— We present a large-scale surface reconstruction method for lidar point cloud, which achieves better surface reconstruction result and real-time performance. The problem is challenging since lidar data is sparse and massive. So we propose a new surface lattice 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 map point cloud. We implemented parallel computation in the reconstruction process and achieve a real-time performance.Compared with other state-of-art reconstruction methods, our method has a great advantage in terms of both quality and speed, which meets the needs of mobile robotic mapping and real-time visualization.**

I. INTRODUCTION

Real-time large-scale 3D scene reconstruction 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 higher measurement accuracy and higher resolution of angles, distances and speeds. Its data collection is independent of weather and light. With large effective distance, it is useful in a large complex scene: over 100×100 m^2. It is suitable for embedding in current robot platforms.

Lidar collects unorganized point cloud and current surface reconstruction methods based on the lidar point cloud are not real-time. Real-time surface reconstruction allows us to know the current reconstruction result timely. For an incomplete surface reconstruction, we can move the sensor to an appropriate location and fill the vacant part of it.

In addition to the real-time problem, due to the inaccurate external parameters calculation, sensor errors and noise, the point cloud stratifies after several scans for a surface. One common way is to cluster point cloud and get a cluster centroid to represent these points [1][2], but this direct fusion method is unable to accurately reconstruct the precise surface of an object. Meanwhile, we found that reconstructing surface after point cloud registration loses lots of useful information. We use the information between frames to solve the stratification problem.

In this paper, a real-time large-scale lidar surface reconstruction method is presented, which achieves better surface reconstruction result and real-time performance. We take the advantage of the pose information of each point cloud frame to update each frame's implicit surface. Our method has a great advantage in denoising compared to the direct drop-down or discarding excessive points in other systems [3]. By repeatedly updating, we get the implicit surface of the object. We draw the isosurface of the point with zero value to get a surface of the scene. Parallel computation is implemented to achieve real-time performance. The paper makes main contributions as follows,

- We present a real-time 3D reconstruction pipeline for large scale Lidar point cloud.
- We propose a novel surface lattice data structure and employ a novel line of sight algorithm to update the implicit surface to achieve both real-time performance and memory efficiency.
- We implement parallel computing to update the implicit surface faster.

The rest of this paper is organized as follows. In Sect. 2, we discuss related work. Sect. 3 formulates the problem formally and explain the notation we used. The surface reconstruction algorithm is presented in Sect. 4 and 5. Experimental results are shown in Sect. 6. Finally, a discussion and a conclusion are made in Sect. 7.

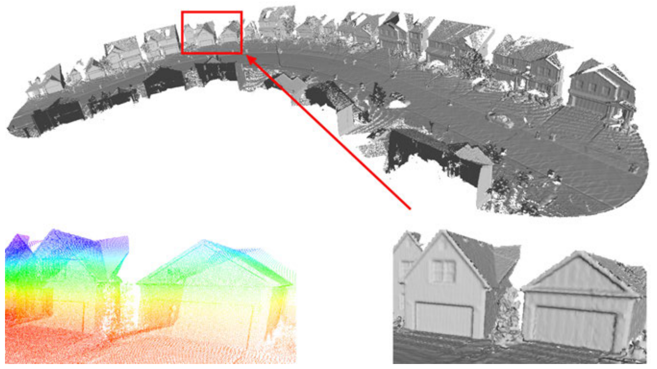
Fig. 1. Our proposed real-time large-scale surface reconstruction method. The first row is a large-scale reconstruction result. The second row are point cloud and detail of reconstruction respectively

II. RELATED WORK

Lidar is a useful range sensor to improve robot ability to create a large-scale unknown environment map and localize within this map [4][5].

The real time 3D reconstruction approaches [6][7][8], which use range sensors such as Kinect, are confined to a certain small region. So T. Whelan et al. [9] and H. Roth et al. [10] use volume movement to achieve larger reconstruction, but still can not handle with large outdoor scenes and comes with a number of limitations such as light dependence.

*Indicates equal contribution. †They are with School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China. Emails: jianzhang@smail.nju.edu.cn, anyirao@smail.nju.edu.cn, ConghuiGeng@smail.nju.edu.cn, allanyu@nju.edu.cn, nackzhou@nju.edu.cn and coff128@nju.edu.cn.

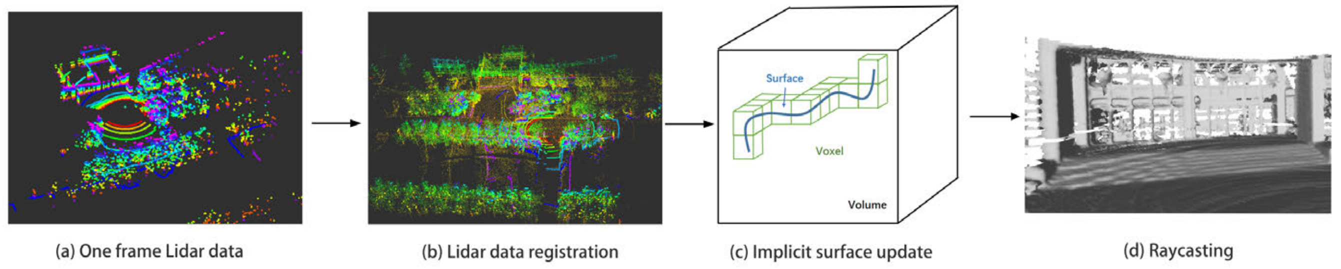

Fig. 2. Overview of our pipeline from point cloud to a raycasted 3D scene. (a) is one frame lidar point cloud as input; (b) is multi-frame point cloud after lidar mapping; (c) is the surface lattice data structure and implicit surface update process; (d) is real-time reconstruction result.

A representative 3D reconstruction large-scale approach using images is presented in [11]. This method produced visually compelling results with a strong assumption of building regularity and Manhattan-world assumption.

Verma et al [12], Zhou et al [13] and Poullis et al [14] create 3D scenes from lidar data. They used classification to remove noise, segmentation to separate individual building patches and ground points, and generate mesh models from building patches. These 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 [15] extend dual contouring [16] into a 2.5D method and produce 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 not easy to deal with other complex scenes, such as large-scale street scenes.

Recent state-of-art lidar large-scale surface reconstruction needs to be done off-line [17][18][19]. Marton et al's triangulation surface reconstruction [3] is a fast incremental method but not real-time. Small object reconstruction can be real-time but sensors need to be very close to objects [20], which can not apply to a large-scale environment. Our primary goal is to achieve real-time performance for large scale scene since real-time is important to fill incomplete part of an object in time.

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. 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. Thirdly, 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 will update them incrementally. At last, the reconstructed surface is raycasted for real time visualization.

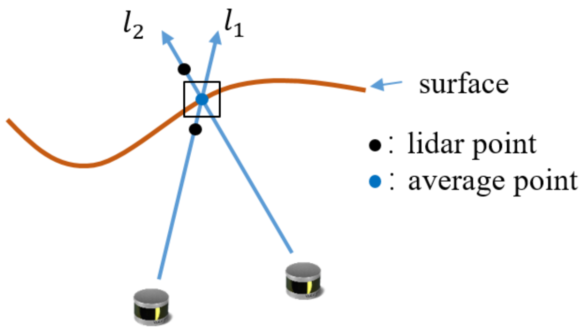

Fig. 3. Implicit surface reconstruction. we use TSDF weighted moving average computation. The first update l_1 and the next update l_2 combine to get an average surface point, which is close to the real point

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 proposed a voxel based line of sight update algorithm, as shown in figure 3 to solve the problem. We get the line of sight through the 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 $P_k^L = \{p_{k1}^L, p_{k2}^L \cdots p_{kn}^L\}$ in lidar coordinate system $\{L\}$, $k \in Z^+$, where k represent lidar frame number, and pose H_k corresponding to kth frame. We use $p_{k,i}^L \in R^3$ to indicate one point in each frame. The TSDF(truncated signed distance function) value is calculated from current frame's lidar point P_k^L, and 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.

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

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 Lattice

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 [6], 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 lattice. The key idea of our implicit surface update method is to generate the line of sight $\overline{op_{ki}}$ from the current lidar point $p_{k,i}^L$ 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.

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

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 $p_{k,i}$.

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.

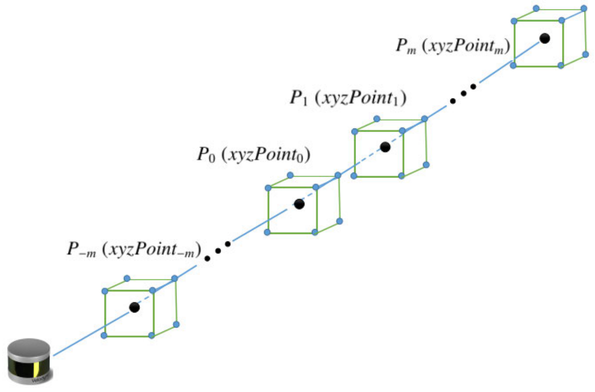

Fig. 4. Generate line of sight. m points and their coordinates in the front of and behind the point P_0

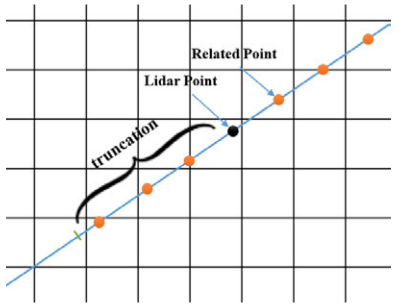

Fig. 5. For example, We get m=3 points and their coordinates in the front of and behind the point. Because $\frac{\partial x}{\partial t} > \frac{\partial y}{\partial t}$. We normalized our ray vector with $\frac{\partial x}{\partial t}$, so we get $2m$ points along x axis rather than y axis

D. Lidar Mapping

Lidar points are received at different times, which result 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 similarity to J. Zhang et al. [21]. 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 through matching different frame's feature points. Therefore, we use sensor motion map and register each frame's point cloud in the world coordinates.

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

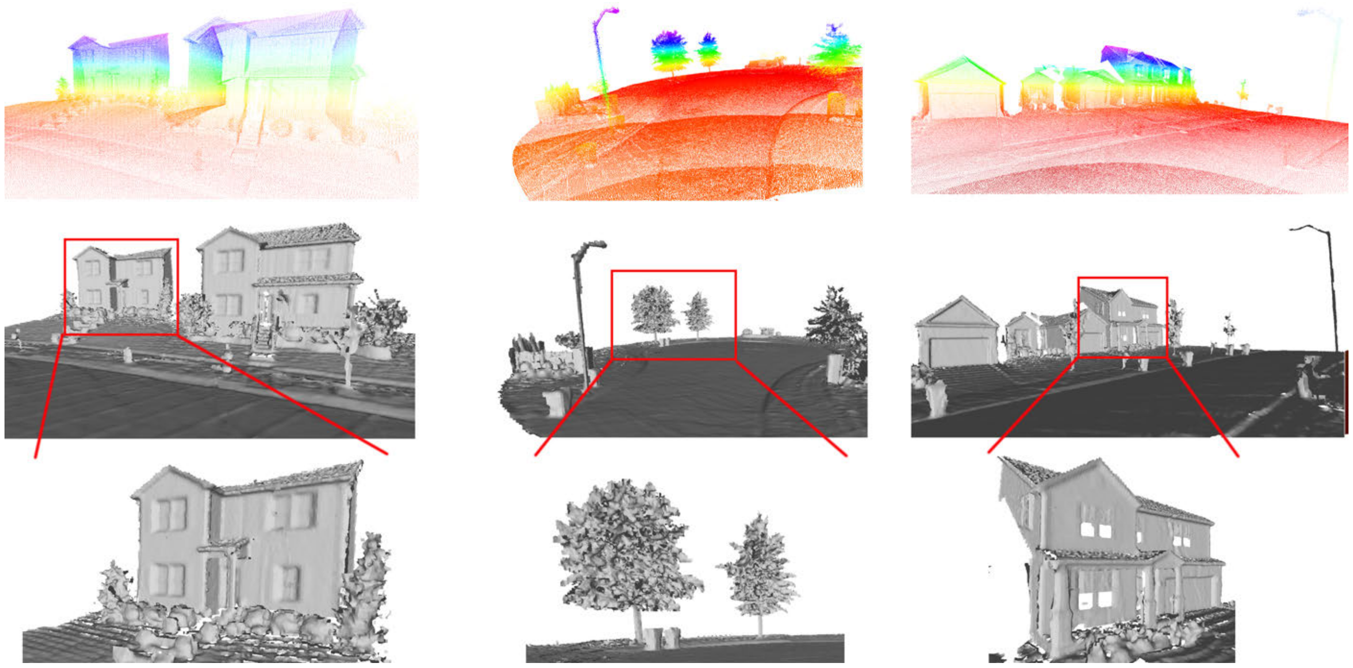

Fig. 6. Surface reconstruction results. From top to bottom: point clouds, surface reconstruction results and their details. Each column corresponds to different environment

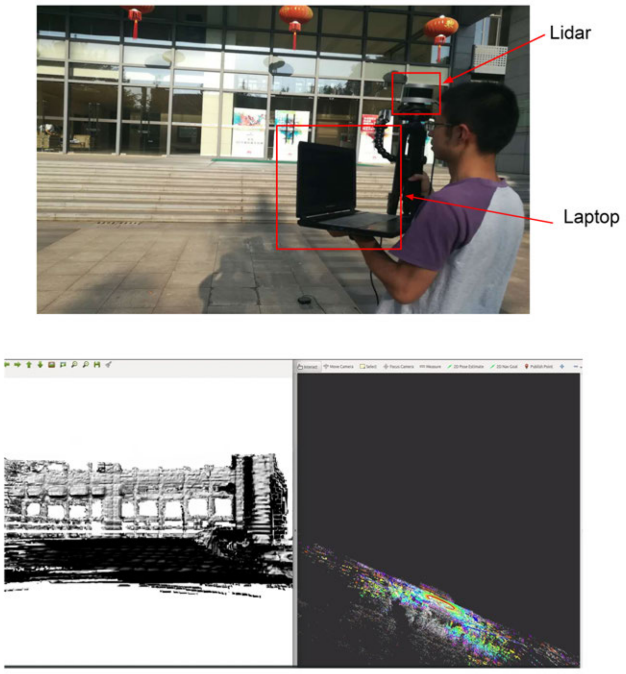

Fig. 7. Our system setup and real-time reconstruction. Our laptop reconstructs a building in real-time. In the laptop screen, the point cloud is on the right, while the surface is on the left. We use a go-pro camera to compare our result with image.

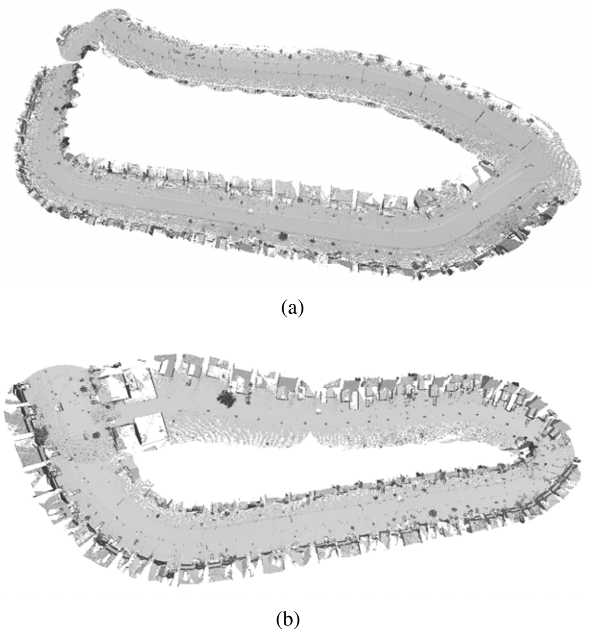

Fig. 8. Large-scale reconstruction results. we reconstruct two loop street (a) containing about 30 houses and (b) containing about 50 houses.

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 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 [22].

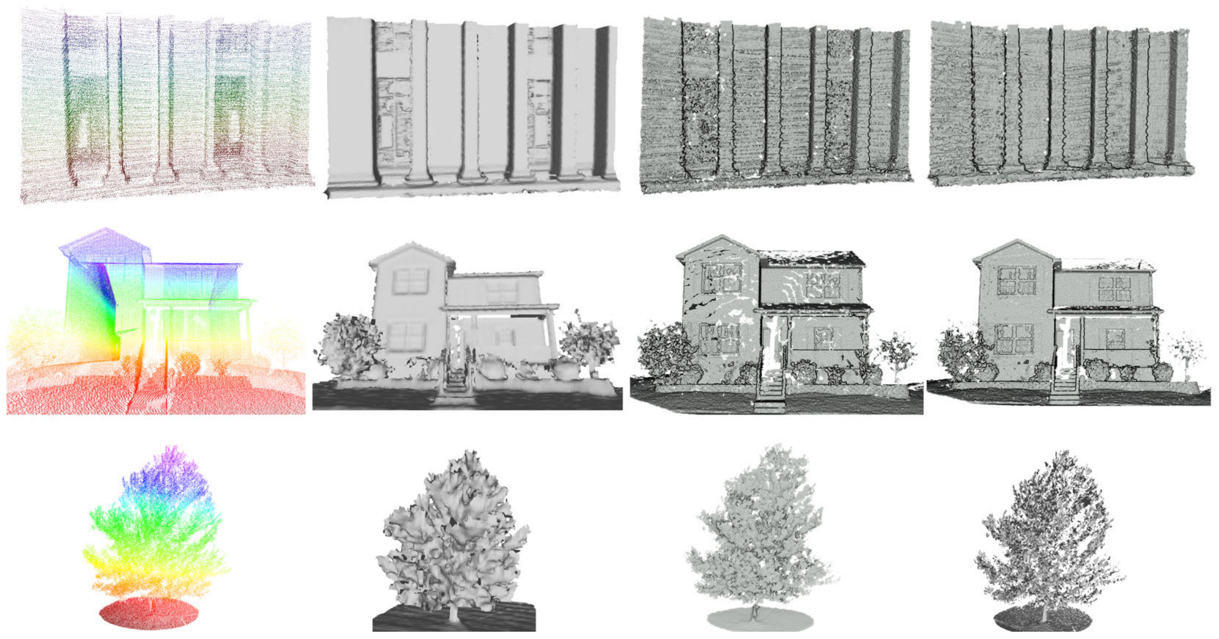

Fig. 9. Comparison of reconstruction result between several methods. The first column is point cloud, the second column is our method, the third column is Zoltan Csaba Marton et al's method[3], and the fourth column is Peer Stelldinger's method[19]. The first row presents different algorithms' reconstruction result using our VLP-16 dataset and the second and the third row is the result of SDR dataset.

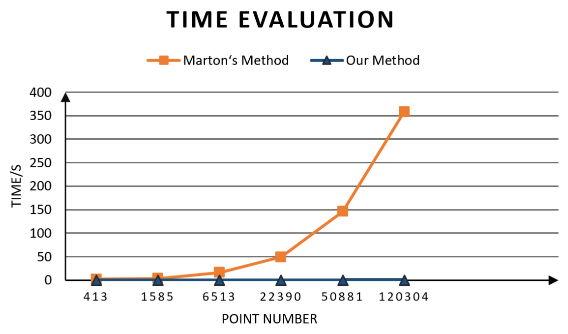

Fig. 10. Time evaluation between Zoltan-Csaba Marton et al's [3] and our methods. when per frame processing point increases, our method keeps at a low time cost, whereas Marton et al's approach increase exponentially, which is unable to reconstruct large-scale point in real-time

IV. EXPERIMENTS AND COMPARISON

Our system is illustrated in figure 7, including a 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. Here is a link to one experiment's video https://www.youtube.com/watch?v=_6JFl3p75qg&feature=youtu.be.

A. Dataset

We use two datasets to compare our algorithm and the other two state-of-art algorithms. One is Lin et al's dataset[23], 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 thousands of points. The VLP-16 dataset is sparse, and every frame is consisted of about two-thousand points.

B. Reconstruction Results

The reconstruction result is 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. We show an overview of our large-scale reconstruction, as shown in figure 8.

C. Comparison of Reconstruction performance

We compared our approach with two state-of-art algorithms: the first is Marton et al's method [3] and the second is Peer Stelldinger's method [19]. We have used Zoltan-Csaba Marton et al.'s provided method on PCL library and implemented Peer Stelldinger's algorithm ourselves 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.

Noisy points cause roughness and holes in reconstruction results. 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. Besides this, Zoltan-Csaba Marton et al's method also discards those newly collected points which overlap with existing surface mesh. This also discards useful points, so it cannot accurately reconstruct surface with few errors.

Peer Stelldinger's algorithm is not incremental and slow. It has low robustness when handling large-scale noisy point cloud because it reconstructs at the end of point registration and loses much useful information. It is unable to complete the incomplete part of reconstruction in time and does not work on noisy points, outlier points, misaligned scans, missing data. Thus, our method achieves both time efficiency and the most visual realism of reconstruction results.

D. *Real Time Performance*

Our surface reconstruction algorithm costs about 0.2 seconds to process 10-frame lidar data. It is very fast and efficient, noting that we only use a normal laptop. We also compare our method with Zoltan-Csaba Marton et al's method in terms of speed. We record processing time according to different frame size and draw Reconstruction Time-Points Number graph, as shown in figure 10. The Zoltan-Csaba Marton et al's method searches a wide range of points for each point to triangulate, and it needs to search more points when the total number of points increases, so the reconstruction time is a non-linear growth about points number. Our method is free of searching. Although Zoltan-Csaba Marton et al's method is faster handling less than 10 thousand points, as the number of points increases, it consumes much more time. Note 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. Therefore, our approach has an advantage in reconstructing large-scale point cloud.

V. CONCLUSION

In this paper, we propose a large-scale surface reconstruction method using 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 a very high calculation speed to meet mobile robotics needs and achieves a better surface reconstruction result. It is capable of completing the incomplete part according to current reconstruction result. Compared with other state-of-art reconstruction methods, our method has a great advantage in terms of both quality and speed, which meets the needs of mobile robotic mapping and real-time visualization.

REFERENCES

[1] T. K. Dey and S. Goswami, *Provable surface reconstruction from noisy samples*. Elsevier Science Publishers B. V., 2006.

[2] H. Si, "Tetgen, a delaunay-based quality tetrahedral mesh generator," *ACM Trans. Math. Softw.*, vol. 41, no. 2, pp. 11:1–11:36, Feb. 2015. [Online]. Available: http://doi.acm.org/10.1145/2629697

[3] 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.

[4] R. Wang, "3d building modeling using images and lidar: A review," *International Journal of Image and Data Fusion*, vol. 4, no. 4, pp. 273–292, 2013.

[5] M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva, "A survey of surface reconstruction from point clouds," in *Computer Graphics Forum*, vol. 36, no. 1. Wiley Online Library, 2017, pp. 301–329.

[6] 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.

[7] M. Kaess, M. Fallon, H. Johannsson, and J. Leonard, "Kintinuous: Spatially extended kinectfusion," *CSAIL Tech. Rep.*, 2012.

[8] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

[9] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense rgb-d slam with volumetric fusion," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 598–626, 2015.

[10] H. Roth and M. Vona, "Moving volume kinectfusion." in *BMVC*, vol. 20, no. 2, 2012, pp. 1–11.

[11] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan, "Image-based street-side city modeling," *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 114, 2009.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] P. Stelldinger, "Topologically correct surface reconstruction using alpha shapes and relations to ball-pivoting," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*. IEEE, 2008, pp. 1–4.

[20] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 169, 2013.

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

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

[23] 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.