Reconstructing Dynamic Objects via LiDAR Odometry Oriented to Depth Fusion

Conference Paper *in* Lecture Notes in Computer Science · August 2017

DOI: 10.1007/978-3-319-65292-4_55

Hui Cheng
China University of Petroleum - Beijing
320 PUBLICATIONS **4,935** CITATIONS
SEE PROFILE

Chen Chuangrong
Sun Yat-Sen University
3 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Chongyu Chen
DMAI(Guangzhou)
15 PUBLICATIONS **171** CITATIONS
SEE PROFILE

Reconstructing Dynamic Objects via LiDAR Odometry Oriented to Depth Fusion

Hui Cheng, Yongheng Hu, Haoguang Huang, Chuangrong Chen, and Chongyu Chen*

School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, Guangdong, 510006 China,
`chenchy47@mail.sysu.edu.cn`,
Home page: `http://hcp.sysu.edu.cn`

Abstract. LiDAR odometry is the key component of LiDAR-based simultaneous localization and mapping (SLAM). However, the low vertical resolution of LiDAR makes it difficult to produce pleasant mapping results. It is even more challenging to reconstruct the surface of dynamic objects from the raw LiDAR input. To address this problem, existing approaches typically divide it into several subproblems like object detection and tracking and then solve them individually, which greatly increases the complexity of LiDAR odometry as well as the SLAM framework. In this work, we propose to address this problem by improving LiDAR odometry with appropriate modifications to the depth fusion process and several additional lightweight components. Extensive evaluations on KITTI dataset and Velodyne HDL-16E laser scanner demonstrate the effectiveness of the proposed method. The results of the improved LiDAR odometry include abundant information about the dynamic objects, which can be used for many high-level tasks such as object recognition and scene understanding.

Keywords: LiDAR odometry, Depth fusion, 3D reconstruction, Simultaneous localization and mapping

1 INTRODUCTION

Simultaneous localization and mapping (SLAM) has been widely studied for years because of its irreplaceable importance to robot perception. A general assumption of LiDAR-based SLAM approaches [6, 10] is that the static scene is scanned by a moving LiDAR, which is inaccurate when there are dynamic objects in the scene. Moreover, the vertical resolution of LiDAR is very low (e.g. 32 lines), making it difficult to perform surface reconstruction in mapping tasks.

To handle dynamic objects in the scene, researchers have made great efforts to improve the SLAM principles from different aspects, such as object detection and tracking. For example, LiDAR measurements on objects can be considered as outliers of the scene measurements. Therefore, the random sample consensus (RANSAC) [5] paradigm, which is originally designed for outlier detection, is widely used for object detection and removal [4]. Specific cues for LiDAR data, such as occupancy grid [14]

* The corresponding author. Email: chenchy47@mail.sysu.edu.cn

and ray tracing [11], are also adopted for object detection in LiDAR scans. To handle dynamic objects continuously, tracking methods are also employed. Moosmann et al. [8] propose to jointly address tracking and self-localization, which typically requires dense depth input (e.g. 64-line LiDAR data) and an additional component of track management. Yang et al. [16] propose a specific RANSAC oriented to spatiotemporal consistency to build links among moving object tracking, multi-scale segmentation, and LiDAR odometry. Although all the components work in a stable manner, the problem of reconstructing dynamic objects remains unsolved due to the sparsity of the raw LiDAR input. It can be observed that existing solutions tend to add more components to handle dynamic scenes, which will increase the computation amounts. This is not preferred in intelligent systems with limited computational resources.

In this paper, we study on the challenging problem of reconstructing dynamic objects from sparse depth data. The proposed solution, which is called *L iDAR odometry oriented to fusion* (LOOF), is built upon standard LiDAR odometry with the orientation to depth fusion for dynamic objects. Several lightweight algorithms are employed to fully exploit necessary information from the LiDAR odometry pipeline. In particular, spatial clustering and temporal association are employed to exploit spatiotemporal consistency of dynamic objects. Parameters for rendering and fusing depth maps are carefully chosen to be adaptive to the sensor characteristics. To the best of the authors' knowledge, the proposed method is the first solution to reasonable surface reconstruction of dynamic objects from sparse LiDAR data. Evaluations on public datasets verify the effectiveness of the proposed method from the aspects of reconstruction quality, odometry accuracy, and computational efficiency. Abundant information about object segmentations and trajectories can be easily extracted from the results of the proposed solution, which can be used for many high-level tasks such as object recognition, object tracking, and scene understanding.

2 PROPOSED FRAMEWORK

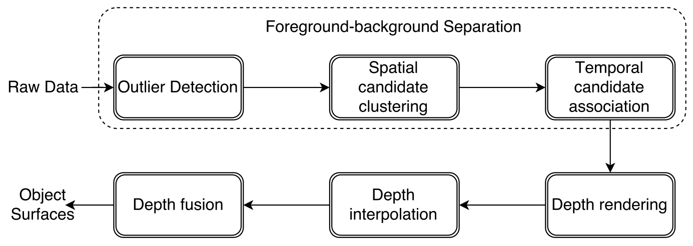

Foreground-background Separation
Raw Data
Outlier Detection
Spatial candidate clustering
Temporal candidate association
Depth rendering
Depth interpolation
Depth fusion
Object Surfaces

Fig. 1. The pipeline of the proposed LiDAR odometry oriented to fusion (LOOF).

In this section, we present our solution of employing lightweight algorithms to LiDAR odometry with orientation of depth fusion. As shown in Fig. 1, The employed algorithms are mainly for foreground-background separation and depth processing.

2.1 LiDAR odometry

Starting with standard LiDAR odometry, the proposed method takes as input a sequence of depth measurements. The set of depth measurements obtained at the same time is defined as a data frame. By exploiting the coordinate information from LiDAR data, we build a multi-dimension map D_i for each data frame, where i is the frame number. In this section, we denote a data point in D_i as $\mathbf{p}(u, v) = \{x, y, z, d\}$, where v is the measurement number related to the scanning azimuth, $u \in \{1, \dots, u_{\max}\}$ is the number of scanning line, d is the depth value, and $\{x, y, z\}$ represents the 3D coordinates of the point in the global coordinate system. For the data provided by mainstream LiDAR, typical values of $u_{\max}$ are 16, 32, and 64. Small value of $u_{\max}$ brings in difficulties to both accurate frame registration and reasonable surface reconstruction.

For the registration between two sequential frames D_i and D_{i+1}, classical point-to-point iterative closest point (ICP) method [1] may produce incorrect registration results. The main reason is that the same LiDAR scanning line is probably on different positions of the object surface in sequential frames, while in computational point of view, the points on the same scanning line will probably be considered as correspondences between two frames. Therefore, we connect the 3D points in D_i as a mesh with normals and adopt standard point-to-plane ICP method [3] for frame registration.

2.2 Foreground-background separation for LiDAR data

To reconstruct dynamic objects, we separate the object measurements from the scene measurements, which is done by an improved RANSAC paradigm with spatial clustering and temporal association.

Adaptive spatial clustering of outliers In the registration between frames of LiDAR data, the depth measurements from the ground usually affects the registration accuracy. Therefore, in this work, we adopt the RANSAC paradigm for two purposes, i.e., ground measurement removal as described in [2] and dynamic object detection. In particular, we divide the data frame into n point sets $\{P_1, \dots, P_n\}$. For every point set P_k, we apply ICP individually to obtain a rigid transforms T_k. After applying T_k to all 3D points, the dynamic points are then detected by exploiting their distances with the corresponding points. In this work, the distance threshold is set to 0.05 m for the frame rate of 10 fps, aiming at detecting dynamic points with a speed no less than 0.5 m/s. The RANSAC iterations stop within 10 iterations. During the iterations, if the average error is smaller than the distance threshold, the iterations will stop immediately.

After the detection of outliers, we cluster the outlier points in the spatial domain, resulting in a preliminary estimation of foreground objects. The density-based spatial clustering of applications with noise (DBSCAN) [15] is employed because of its nice ability in preserving object continuity. There are two key parameters of DBSCAN, i.e., radius of search window r_{win} and the minimal number of points within the searching radius which is denoted as $n_{\min}$. Since the density of laser-scanned measurements decreases as the depth increases, both r_{win} and $n_{\min}$ have to be adaptive to depth. In this work, we propose to compute r_{win} for the current point as:

$$r_{win}(d_2) = s * r_{win}(d_1) \tag{1}$$

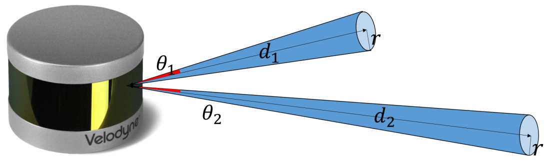


Fig. 2. The relationship between scanning range and object-LiDAR distance with a fixed scanning area.

where the scale

$$s = \frac{\arctan^2(1/d_1)}{\arctan^2(1/d_2)},$$

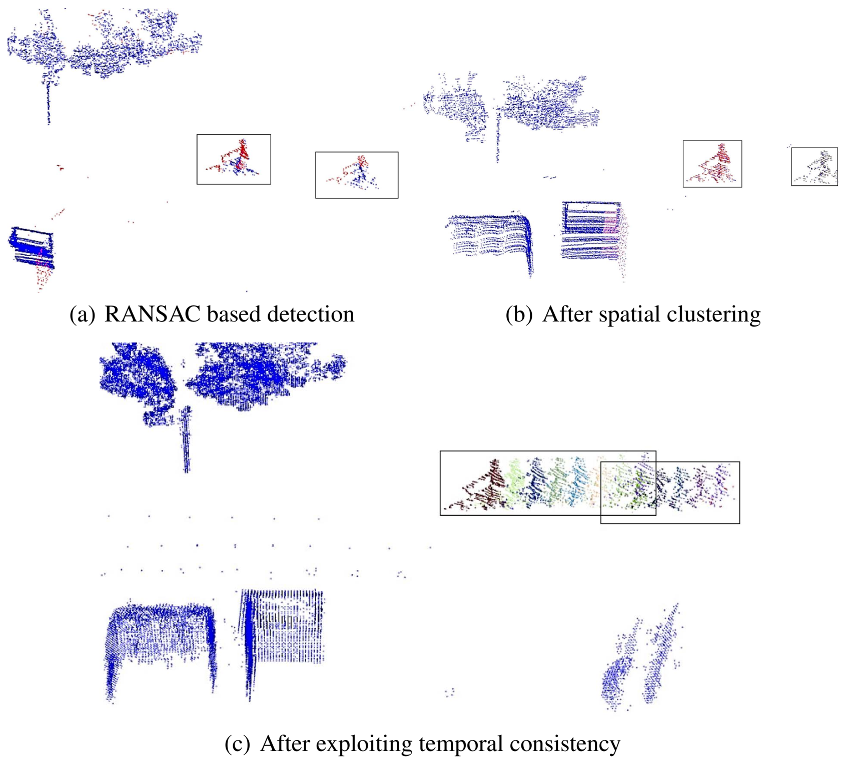
$r_{win}(d_1) = 0.5$ m is the searching radius for the depth of $d_1 = 10$ m, d_2 is the depth of the current 3D point. The reason for using Eq. (1) that the number of LiDAR measurements for the same scanning area become smaller as the distance increase. As illustrated in Fig. 2, for the same scanning area of πr^2, there are θ_1/θ_{int} measurements for the distance of d_1, while there are θ_2/θ_{int} measurements for the distance of d_2, where the scanning interval θ_{int} is a fixed value for a given LiDAR device.

It should be noticed that these separation results are not perfect that some foreground points will be considered as background and vice versa. An example of incorrect labelling is illustrated in Fig. 3 (a) and (b).

Temporal association of object candidates The reconstruction of object surface usually requires continuous scanning from different views. Therefore, we need to build the links for the 3D points from the same object across different frames. Although the spatial clustering results are not perfect, they still provide nice object candidates when being applied to individual frames. Given the imperfect object candidates, we propse to exploit temporal consistency for linking object candidates across frames. In particular, we first divide the sequence into several segments of identical length. Then, close cluster centers across sequential frames are linked together, so that each cluster corresponds to a trajectory. The trajectories that lasts for more than 0.5 second (5 frames) with a breakup smaller than 0.5 second are preserved and marked as foreground. Other trajectories will be removed and the related clusters are marked as background. Then, we compass the points in cubes respectively determined by the maximum and minimum coordinate of the points with foreground label , by doing which the foreground labels are propagated to the surrounding 3D points,resulting in more labeled foreground points. An example of the objects extracted by applying spatial clustering and temporal association is shown in Fig. 3 (c). It is shown that considering the temporal consistency can lead to more accurate foreground separation.

2.3 Depth interpolation and fusion

Inspired by the success of KinectFusion [9], we resort to depth fusion in the form of truncated signed distance function (TSDF) for reconstructing the surface of dynamic objects. All the foreground points are used to produce depth maps in an object-wise manner.


(a) RANSAC based detection (b) After spatial clustering

(c) After exploiting temporal consistency

Fig. 3. Dynamic object detection after different steps, in which blue points represent the scene points and dots in other colors represent dynamic points. There are two moving object in the scene, which are marked by black rectangles. All frames of point clouds are shown in (c), with different colors indicating dynamic points from different frames. It can be observed that the proposed method with temporal considerations can well detect the moving objects while sorting out most points with incorrect labels.

Depth interpolation Considering that the depth measurements are sparse in the vertical direction, we propose to interpolate depth values in the index domain. That is, as for points $\mathbf{p}(u, v) = \{x, y, z, d\}$ in a data frame D_i, in every vertical line, we firstly interpolate the vertical scanning line index u uniformly and acquire the expected vertical index u^*, then we use the initial index u, expected index u^* and the depth d corresponding to index u as the input of the cubic spline interpolation promising the first and second derivative of the curve and get a denser data frame. As we can see in Fig. 4, the raw object points are very sparse in vertical direction. By applying depth interpolation, we can acquire denser point cloud about the object.

After interpolation, we compute the 3D coordinates for every interpolated point, resulting in a denser version of data frame. These denser point clouds bridge the gap between the depth measurements from LiDAR scan and depth camera, paving the way to utilize existing techniques for depth fusion.

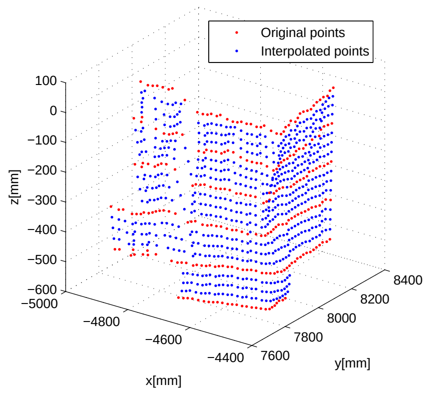


Fig. 4. Depth interpolation for the object surface.

Depth rendering Given a sequence of interpolated 3D points for the dynamic objects, we have to convert them to depth maps before surface reconstruction. In this work, we propose to generate depth maps by placing a virtual camera pointing at every object. The coordinate system of the virtual camera is shown in Fig. 5 (a). Note that we only have to rotate the virtual camera around the y-axis to make z-axis point at the object because the LiDAR is usually placed on a horizontal plane and thus its x-axis is horizontal. The rotating angle is determined by exploiting the LiDAR coordinate system. For example, the Velodyne Laser scanner ihas 64 lines in the vertical direction and 4, 000 points for every scanning line. It can be computed that the angle interval between neighboring scanning points (i.e. the horizontal precision) is $\theta_{int} = 360°/4000 = 0.09°$. Therefore, the rotating angle can be computed by $\theta = 0.09\bar{u}$, where $\bar{u}$ is the median index of the LiDAR measurements on the object.

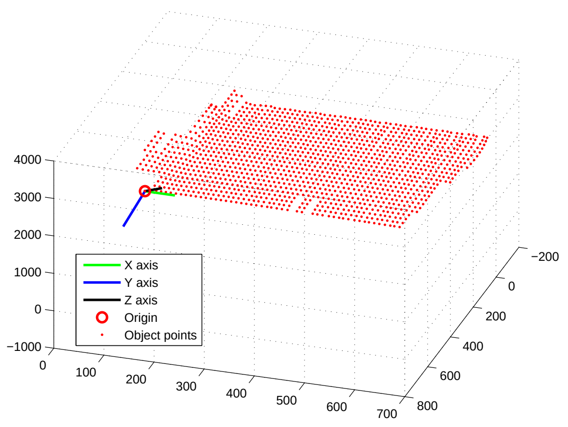
With a virtual camera pointing at the object, we can generate a depth map by projecting the 3D points to 2D image plane according to the perspective imaging model. There are several considerations for choosing the focal length f_x and f_y and the resolution of the depth map $H \times W$:

1. The focal lengths f_x and f_y can be neither too small or too large because the generated depth maps will be either too dense or too sparse;
2. The width W and the height H of the depth map should be large enough so that the object can be completely shown;
3. The resolution $H \times W$ cannot be too large due to the limitations of computation and storage resources.

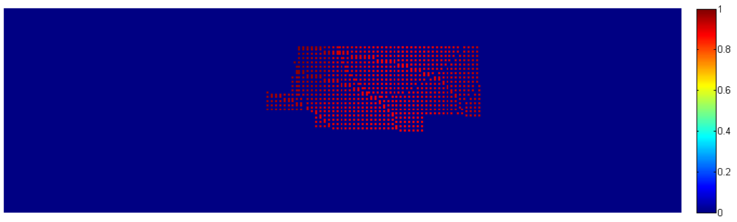
Let θ_l denote the vertical precision of LiDAR measurement. We propose to set f_x, f_y, W, and H with respect to the constraints described by the following approximations:

$$\frac{W}{H} \approx \frac{f_x}{f_y} \approx \frac{\theta_{int} * k}{\theta_l} \tag{2}$$

where k is an adaptive coefficient which is in inverse proportional to the LiDAR-object distance while in proportional to the object size. Fig. 5 (b) shows an example of the



(a) The coordinate system of the virtual camera.



(b) The depth map rendered by the virtual camera.

Fig. 5. An example of generating depth maps from 3D points of the object.

depth map rendered from a 16-line LiDAR input. In this example, we choose $f_x = 500$, $f_y = 100$, $W = 396$, $H = 96$ for the object distance of 4.0 m, horizontal precision of $0.2°$, and vertical precision of $2°$. It should be noted that for W and H that can be exactly divided by 32, one can get a higher computational efficiency in GPU implementation.

Depth fusion Given the rendered depth maps for every dynamic object, we propose to fuse them in the form of TSDF as described in KinectFusion [9]. Since the depth maps from LiDAR are different from that from Kinect, adaptions for the fusion process are necessary. Key issues include camera intrinsic parameters, volume size, and voxel size. Similar to depth rendering, TSDF fusion also requires a virtual camera. To maintain the rendering consistency, in depth fusion, we use a virtual camera with intrinsic parameters identical to that of the camera for depth rendering. Different from KinectFusion that uses a fixed volume with pre-defined voxel size, in this work, we use adaptive volume size and voxel size for each object. That is, the volume should contain the whole object, whose size is determined by the minimum and maximum coordinates of the object point cloud. More importantly, the voxel size is set according to the precision along each axis direction. For example, the voxel length along z-direction should be smaller than that along the y-direction because LiDAR has a low precision along the y-axis. Considering the data density is dependent on the distance, we suggest that the voxel size should increase as the distance between object and LiDAR increases. After determining these key parameters, we enable surface reconstruction with given LiDAR data by the TSDF module used in KinectFusion.

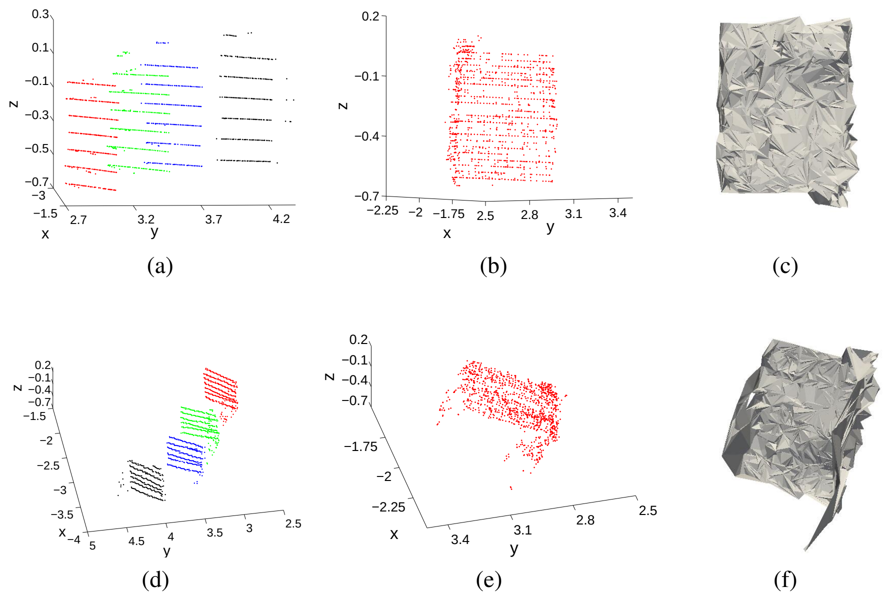


Fig. 6. An example of depth fusion for a dynamic object of box. The point cloud is obtained by Velodyne HDL-16E, which only has 16 scanning lines along the vertical direction. (a) shows 4 frames of the 10 frames of extracted LiDAR data. (b) is the fusion result of 10 frames. (c) is the mesh constructed by triangulating the point clouds extracted from the TSDF. (d), (e), and (f) are showing the same objects of (a), (b), and (c) from another view, respectively.

Taking the LiDAR data from Velodyne HDL-16E as an example, we perform the fusion process for an object, i.e. a box. The extracted data and fusion results are shown in Fig. 6, which demonstrate that the object surface is successfully reconstructed. Note that the triangulation algorithm [7] adopted for converting TSDF to mesh is also used in KinectFusion.

3 Experiments

In this section, the proposed method is evaluated from the aspects of odometry accuracy, reconstruction quality, and computational efficiency.

3.1 The odometry accuracy and computational efficiency

The odometry accuracy is evaluatied by conducting quantitative experiments on the KITTI odometry datasets. First, we conduct experiments using the adopted LiDAR odometry on two sequences, Seq. #3 and #7. In Seq. #3, the LiDAR is moved from an urban environment to a highway. In Seq. #7, the LiDAR keeps moving on a common street. Related results are demonstrated in Fig. 7, in which (a) and (b) are for Seq. #3 and (c) and (d) are for Seq. #7, respectively. It is shown that the adopted LiDAR odometry achieves nice accuracy on these two data sets. The relatively big error occurs in the estimation of translation vector for Seq. #3 is caused by the scene change from urban environment to a highway. In open environments such highway, the LiDAR measurements will concentrate on far distances and thus ICP based odometry will be inaccurate.

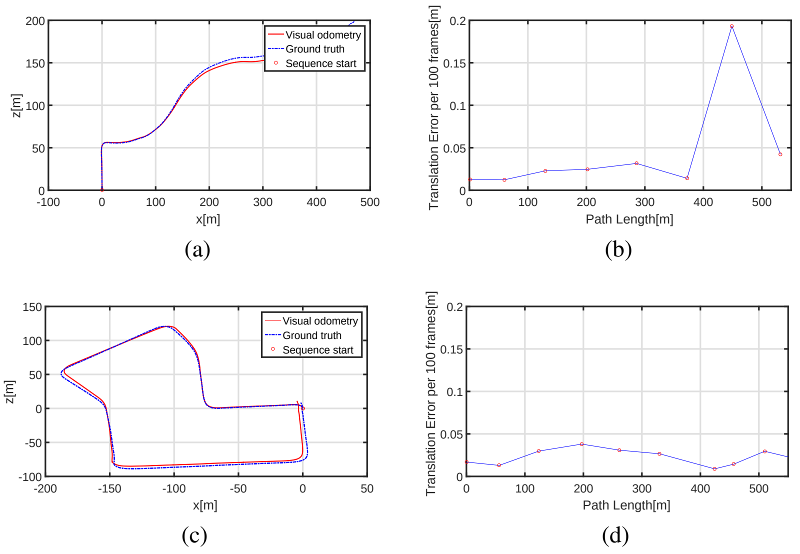


Fig. 7. Odometry results of the adopted LiDAR odometry on two data sets of the KITTI odometry data. (a) Comparison of odometry trajectories on Seq. #3; (b) The translation errors on Seq. #3; (c) Comparison of odometry trajectories on Seq. #7; (d) The translation errors on Seq. #7.

For further evaluations, we compare the adopted LiDAR odometry with two state-of-the-art methods are compared, generalized iterative closest point (GICP) [12] and collar line segments (CLS) [13]. Note that the CLS method can be improved by processing multiple scans. This improved version, named CLS-M, is also included in our evaluations. Table. 1 illustrate the overall odometry results and computational efficiencies of the compared methods on 7 data sets, where bold fonts indicate the state-of-the-art results. It is shown that the proposed method can achieve comparable odometry accuracy while keeping the lowest computational complexity. Note the failure cases, i.e., Seq. #1 and Seq. #2, of the adopted LiDAR odometry are not reported because these cases only include open environments. Considering other data sets are obtained

Data set	Frame number	GICP [12]	CLS [13]	CLS-M [13]	Proposed
Seq. #0	4540	**0.0315**	0.0622	0.0529	0.0931
Seq. #3	800	**0.0218**	0.0275	0.2390	0.0442
Seq. #4	270	0.0497	0.0316	0.0394	**0.2320**
Seq. #5	2760	**0.0228**	0.0726	0.0413	0.0884
Seq. #6	1100	0.0362	**0.0327**	0.0383	0.0384
Seq. #7	1100	0.0132	0.0222	**0.0117**	0.0208
Seq. #8	4070	0.0626	0.1001	0.0643	**0.0451**
Avg. time/frame	/	25.68 s	2.36 s	28.56 s	**1.58 s**

Table 1. Quantitative evaluation of the odometry accuracy and computational efficiency.

from common urban environments, the reported results are believed to be sufficiently representative.

3.2 Computational efficiency

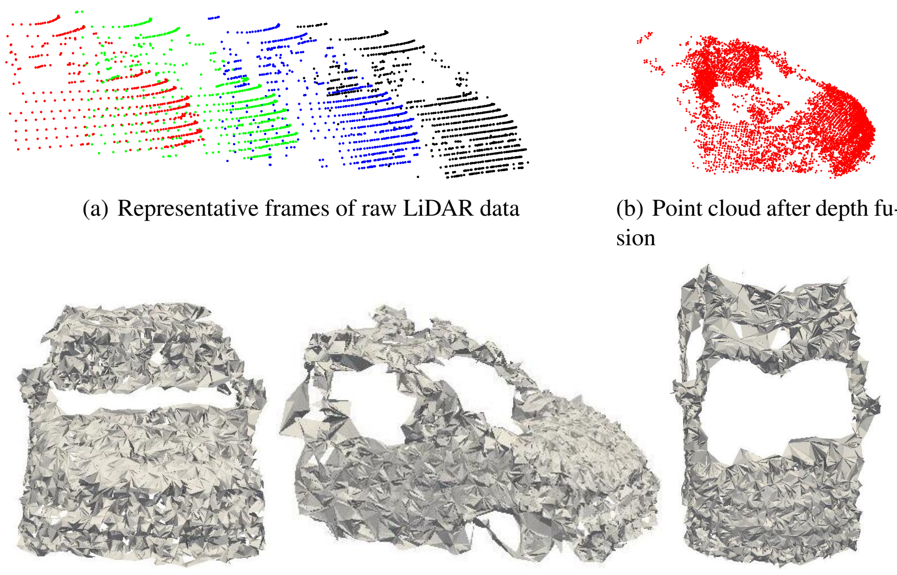
Besides the LiDAR odometry, our solution also employ several lightweight algorithms. The employed algorithms are implemented in C/C++ without any code optimization. By analyzing the execution time of these algorithms on a common PC with a 3.6 GHz CPU using a small subset of the KITTI dataset, we obtain the following efficiency results.

1. The spatial clustering consumes 42 ms on average;
2. The temporal association consumes 0.35 ms on average;
3. The partial GPU implementation of depth fusion consumes 131 ms for fusing every depth map with around 7×10^3 non-zero pixels.

Note that all these employed algorithms are only applied on the short sequence of dynamic objects. They can be done in a separated thread and thus do not affect the real-time performance of the LiDAR odometry.

3.3 Surface reconstruction for objects

To verify the effectiveness of the proposed depth fusion, we take as an example a dynamic object extracted by the employed algorithms from Seq.#3. As shown in Fig. 8, the extracted object is a car at the distances ranging from 15 m to 18 m. According to the principles described in Eq. (2), we use a voxel size of $0.035 \times 0.07 \times 0.035$ m^3. The depth maps are rendered with a virtual camera whose focal lengths are $fx = 500$ mm and $fy = 180$ mm and a depth resolution of 352×96. Note that both 352 and 96 can be exactly divided by 32. The visual comparison between Fig. 8 (a) and (b) indicates that, with proper parameter setup, the adopted TSDF depth fusion successfully merge the depth maps constructed from the sparse LiDAR data. The point cloud extracted from the TSDF representation is denser compared to the raw LiDAR input. Although the reconstructed surfaces are with artifacts, they still illustrate the major shape of the whole object. It is believed that when the object is closer or there are more LiDAR measurements, the reconstruction quality can be further improved.



(a) Representative frames of raw LiDAR data

(b) Point cloud after depth fusion

(c) Frontal view

(d) Side view

(e) Vertical view

Fig. 8. The result of surface reconstruction on a dynamic object (car) extracted from Seq. #3. (a) The 1st, 3rd, 6th, and 9th frames of LiDAR data on the object; (b) The point cloud of the object constructed by fusing 10 frames of LiDAR data via depth fusion; (c), (d), and (e) illustrate different views of the reconstructed object surface.

4 CONCLUSIONS

In this paper, we have studied the challenging problem of reconstructing dynamic objects from sparse LiDAR data and proposed a pioneer solution named LiDAR odometry oriented to fusion (LOOF). Several lightweight algorithms are employed to exploit the information about dynamic objects from classical ICP-based odometry framework. It is shown that the extracted information is sufficient to reconstruct reasonable object surfaces without adding complex components. It is believed that LOOF can achieve better reconstruction when the depth input is denser. Future works may include faster implementation of the employed algorithms and integrating LOOF and LiDAR-SLAM in a multi-thread framework.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant 61602533, NSFC-Shenzhen Robotics Projects (U1613211), The Fundamental Research Funds for the Central Universities, and Science and Technology Program of Guangzhou, China (201510010126).

References

1. Besl, P.J., Mckay, N.D.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2), 239–256 (1992)
2. Cesic, J., Markovic, I., Juric-Kavelj, S., Petrovic, I.: Detection and tracking of dynamic objects using 3d laser range sensor on a mobile platform. In: 2014 11th International Conference on Informatics in Control, Automation and Robotics (ICINCO). pp. 110–119 (2014)
3. Chen, Y., Medioni, G.: Object modelling by registration of multiple range images. Image and vision computing 10(3), 145–155 (1992)
4. Dewan, A., Caselitz, T., Tipaldi, G.D., Burgard, W.: Motion-based detection and tracking in 3d LiDAR scans. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 4508–4513. IEEE (2016)
5. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. ACM (1981)
6. Kohlbrecher, S., von Stryk, O., Meyer, J., Klingauf, U.: A flexible and scalable SLAM system with full 3D motion estimation. In: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics. pp. 155–160 (Nov 2011)
7. Marton, Z.C., Rusu, R.B., Beetz, M.: On fast surface reconstruction methods for large and noisy point clouds. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. pp. 3218–3223. IEEE (2009)
8. Moosmann, F., Stiller, C.: Joint self-localization and tracking of generic objects in 3D range data. In: 2013 IEEE International Conference on Robotics and Automation (ICRA). pp. 1146–1152. IEEE (2013)
9. Newcombe, R.A., Davison, A.J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., Fitzgibbon, A.: KinectFusion: Real-time dense surface mapping and tracking. In: Int'l Symposium Mixed Augmented Reality (ISMAR). pp. 127–136 (2011)
10. Opromolla, R., Fasano, G., Rufino, G., Grassi, M., Savvaris, A.: LIDAR-inertial integration for UAV localization and mapping in complex environments. In: 2016 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 649–656 (June 2016)
11. Pomerleau, F., Krüsi, P., Colas, F., Furgale, P., Siegwart, R.: Long-term 3d map maintenance in dynamic environments. In: 2014 IEEE International Conference on Robotics and Automation (ICRA). pp. 3712–3719. IEEE (2014)
12. Segal, A., Haehnel, D., Thrun, S.: Generalized-icp. In: Robotics: science and systems (RSS). vol. 2 (2009)
13. Velas, M., Spanel, M., Herout, A.: Collar line segments for fast odometry estimation from velodyne point clouds. In: IEEE International Conference on Robotics and Automation (ICRA). pp. 4486–4495. IEEE (2016)
14. Vu, T.D., Burlet, J., Aycard, O.: Grid-based localization and online mapping with moving objects detection and tracking: new results. In: 2008 IEEE Intelligent Vehicles Symposium. pp. 684–689 (June 2008)
15. Wang, W.T., Wu, Y.L., Tang, C.Y., Hor, M.K.: Adaptive density-based spatial clustering of applications with noise (DBSCAN) according to data. In: International Conference on Machine Learning and Cybernetics. pp. 445–451 (2015)
16. Yang, S.W., Wang, C.C.: Simultaneous egomotion estimation, segmentation, and moving object detection. Journal of Field Robotics 28(4), 565–588 (2011)