2019 International Conference on Robotics and Automation (ICRA)
Palais des congres de Montreal, Montreal, Canada, May 20-24, 2019

Dense Surface Reconstruction from Monocular Vision and LiDAR

Zimo Li, Prakruti C. Gogia, and Michael Kaess

Abstract— **In this work, we develop a new surface reconstruction pipeline that combines monocular camera images and LiDAR measurements from a moving sensor rig to reconstruct dense 3D mesh models of indoor scenes. For surface reconstruction, the 3D LiDAR and camera are widely deployed for gathering geometric information from environments. Current state-of-the-art multi-view stereo or LiDAR-only reconstruction methods cannot reconstruct indoor environments accurately due to shortcomings of each sensor type. In our approach, LiDAR measurements are integrated into a multi-view stereo pipeline for point cloud densification and tetrahedralization. In addition to that, a graph cut algorithm is utilized to generate a watertight surface mesh. Because our proposed method leverages the complementary nature of these two sensors, the accuracy and completeness of the output model are improved. The experimental results on real world data show that our method significantly outperforms both the state-of-the-art camera-only and LiDAR-only reconstruction methods in accuracy and completeness.**

Fig. 1. Textured mesh without shading from our reconstruction pipeline that exploits the complementary properties of cameras and LiDARs. Our method preserves fine shapes while reconstructing textureless surfaces.

I. Introduction

Dense 3D reconstruction has gained popularity in recent years because of its growing applications, such as inspection [20], cultural heritage preservation [6] and urban reconstruction [14]. In computer vision, multi-view stereo (MVS) methods employ only cameras to accomplish dense reconstructions. Moving cameras enable precise reconstruction and texture mapping of object surfaces. However, the performance of MVS highly depends on the lighting condition and the richness of textures. Even for scenes with appropriate lighting and rich textures, MVS may still fail in areas with similar camera viewing angles due to insufficient baseline. On the contrary, 3D LiDARs that are widely used in robotics for 3D perception, provide geometric information independent of visual features or textures. However, it is difficult for LiDAR-only methods to reconstruct compact objects accurately in indoor scenes since LiDAR measurements are sparse compared with pixel measurements from cameras. Besides, the noise of LiDAR is relatively large for close-up objects. Therefore, the complementary nature of the two sensors enables us to obtain precise geometric information for both small objects and low-texture structures in indoor environments.

In the last decade, the combination of camera and LiDAR has been widely utilized in the field of autonomous robotics [22]. Previous research on reconstruction that employs the combination of the two sensors typically adopts measurements from only LiDAR in the final geometric surface reconstruction stage, while images are only used to colorize or texturize the surfaces. Different from previous methods, our method utilizes measurements from both sensors to generate 3D reconstructions in the form of a surface mesh.

The pipeline of our system consists of two main stages as shown in Fig. 2. In the first stage, we use LiDAR measurements as priors to improve point cloud densification from images. In the second stage, for areas where the depth cannot be estimated from images, we add LiDAR measurements to the visual point cloud and extract a surface mesh from the point cloud containing measurements from both sensors. Our proposed method leverages integral advantages of 3D LiDAR and camera, hence improves the reconstruction results over state-of-the-art MVS pipelines. Our main contributions are:

- A new pipeline combining LiDAR and camera in a MVS reconstruction framework to create a dense point cloud of indoor environments;
- An extension of the Delaunay tetrahedra and graph optimization framework described in [9] to include both LiDAR and camera measurements in surface mesh extraction;
- Evaluation against state-of-the-art MVS pipelines (PMVS2 [5], OpenMVS [2]) and LiDAR-only reconstruction on real world data.

II. Related Work

Over the past few decades various algorithms that combine LiDAR and camera data for the purpose of 3D mapping or reconstruction have been developed [21][14]. These works

This work was supported by Autel Robotics under award number A020215. We would like to thank Cong Li, Weizhao Shao, and Srinivasan Vijayarangan for data collection.

The authors are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA. {zimol, kaess}@cmu.edu, pgogia@alumni.cmu.edu

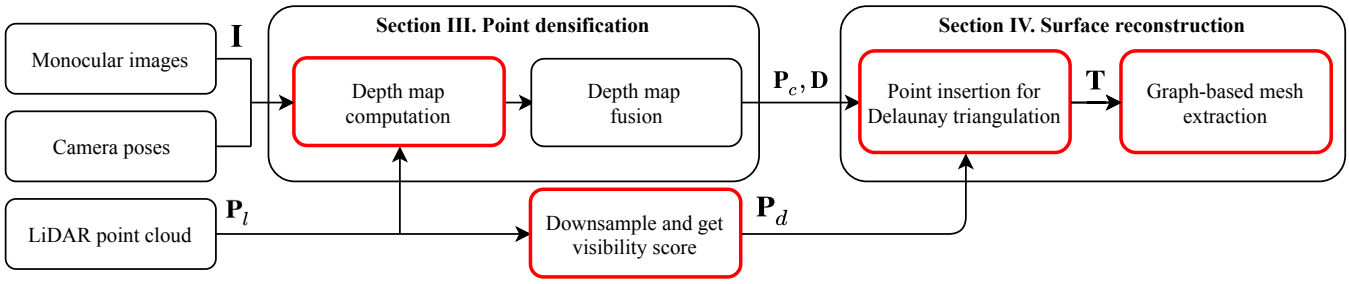


Fig. 2. Diagram of our reconstruction pipeline. Boxes marked in *red* are the parts that we modify based on MVS pipeline. Notations are detailed in following sections.

can be divided into two categories. The first category creates the surface mesh or other representations using only the LiDAR points and uses camera data for texture mapping. The second category uses the LiDAR points as priors for multi-view stereo to generate a dense point cloud from images.

In the first category methods, their reconstruction is mostly represented as a registered LiDAR point cloud. Zhang and Singh [24] generate a map in the form of registered LiDAR point clouds. This system applies camera measurements to help with the state estimation, but the final surface model is made from only LiDAR measurements. With a 2D range sensor and a camera, Martin et al. [13] present an algorithm that extracts a surface model from rangefinder measurements and maps image textures to that model. Similarly, [19] and [16] also utilize texture mapping in their algorithms. While they exploit the advantages of LiDAR in planar and distant scenes, the geometric information in images is not combined with range data in the reconstruction or mapping stage.

For methods in the second category, [12] and [15] both utilize 3D LiDARs and cameras to estimate dense depth maps. Although these methods adopt probabilistic methods to generate accurate visual point clouds with LiDAR priors, LiDAR points are not directly integrated into the reconstruction stage. These methods also only apply to single view depth estimation, while we are interested in larger-scale reconstruction from a sequence.

For vision-only reconstruction, a number of MVS algorithms have been developed in recent years [17]. Furukawa and Ponce [5] developed a patch-based MVS pipeline (PMVS) to reconstruct compact objects. Since PMVS depends on finding pixel-level correspondences across images, low texture environments result in low completeness maps. In [23], Vu et al. proposed a dense scene reconstruction pipeline which can generate a surface mesh even under uncontrolled imaging conditions. With global visibility taken into account, Vu's pipeline improves the accuracy of the surface mesh. However, its performance still relies on a number of features to extract a dense point cloud for generating a precise mesh.

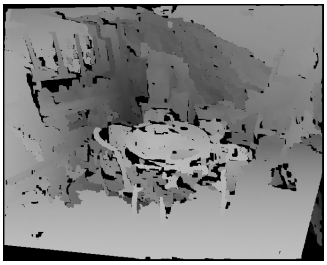
In this paper, we use OpenMVS [2] as our baseline method, which is implemented based on Vu's method. We assume the followings are provided: known camera and LiDAR poses, sparse 3D feature points from SfM pipeline, and the known calibration between the LiDAR and camera for registering images and LiDAR scans into a common coordinate frame.

III. LiDAR-Improved Point Cloud Densification

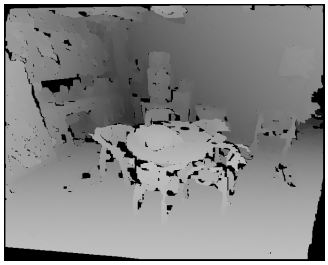
There are three stages in point cloud densification: depth map initialization, refinement, and fusion. We incorporate LiDAR measurements in the first stage to initialize the depth map. We denote the registered LiDAR point cloud from multiple scans using corresponding poses as $\boldsymbol{P}_l$. Input image frames are represented by the set $\boldsymbol{I} = \{I_0, ..., I_{n-1}\}$, and corresponding depth maps are denoted by the set $\boldsymbol{D} = \{D_0, ..., D_{n-1}\}$. Comparing to the method in [3], which randomly initializes the depth for each pixel but can not always converge to the correct depth, our method uses LiDAR measurements as prior to improve the initialization. We initialize the depth map D_i, by projecting points in $\boldsymbol{P}_l$ back to I_i's image frame. There are several cases when we initialize the depth for one pixel:

- *Several projected LiDAR measurements available*: only the closest measurement to the camera center is used for initialization, which accounts for occluding surfaces.
- *Sparse feature points available*: we take the depth information from the sparse feature point for initialization, even when projected LiDAR measurements exist for the pixel. As mentioned above, the sparse feature point is from structure from motion, which employs robust multi-view geometry methods to calculate feature point positions, therefore it is more accurate than LiDAR measurements.
- *Neither camera points nor projected LiDAR measurements available*: We use the initialized pixels which are outputs from previous two cases as vertices to form a 2D triangulation inside the image plane. For each triangle facet in the triangulation, the depth of uninitialized pixels inside it is set to the distance from camera center to the facet.

After the initialization, we find matching patches and perform spatial propagation for refinement as detailed in [3]. Finally, we project all depth maps in $\boldsymbol{D}$ to 3D space and use the fusion method of [18] to reject inconsistent depths. Fig. 3 shows that OpenMVS [2] fails to estimate depth in textureless areas, but incorporating LiDAR measurements improves the depth map estimation. Eventually, we generate a dense point cloud $\boldsymbol{P}_c$ from images in $\boldsymbol{I}$.



(a) Depth map from OpenMVS [2]



(b) Depth map from our method

(c) Corresponding original image

Fig. 3. Comparison of depth maps from OpenMVS and our pipeline. (a) Textureless areas, such as the wall and chairs, result in poor depth estimation when using image-based MVS. (b) Fusing LiDAR measurements with the MVS depth map significantly improves depth estimation in the low texture regions.

IV. Surface Reconstruction from Fused Measurements

We modify the OpenMVS pipeline to use LiDAR measurement during surface reconstruction. After generating the point cloud $\boldsymbol{P}_c$ from the previous step, we combine it with a subset of LiDAR points downsampled from $\boldsymbol{P}_l$ into one point cloud $\boldsymbol{P}_{all}$. Then, we use 3D Delaunay tetrahedralization on $\boldsymbol{P}_{all}$ and fit the resulting tetrahedra into a graph for *s-t* cut algorithm to label each tetradedron as either inside or outside of the surface. Finally we generate a watertight surface mesh.

A. Point insertion

We insert all points in $\boldsymbol{P}_c$ into the tetrahedralization based on the method in [9] since they are accurate from the point cloud densification, but not insert all points from $\boldsymbol{P}_l$. The first reason is that it is expensive for graph-cut algorithm to run through a large number of nodes. Secondly, LiDAR points usually have larger noise[1] than triangulated camera points for short distance measurements, which make the reconstructed surface bumpy. Therefore, we downsample $\boldsymbol{P}_l$ to $\boldsymbol{P}_d$ by clustering points in $\boldsymbol{P}_l$ within a given radius r to a single mean point. After downsampling, for each point p in $\boldsymbol{P}_d$, we project it back to every camera frame I_i. If p is inside I_i's camera view, we record I_i in a set $\boldsymbol{I}_p$, of which the corresponding set of depth maps is called $\boldsymbol{D}_p$. If the depth of projected p is not calculated in any depth map in $\boldsymbol{D}_p$, p is inserted into the tetrahedralization. As a result, inserted LiDAR measurements do not pollute the camera measurements in the same areas.

[1]Up to $\pm$3cm for Velodyne VLP-16 according to its datasheet.

B. Graph-based extraction of surface mesh

For 3D mesh generation, previous methods [10][9][7] use a graph-cut algorithm [4] to extract a surface. In our proposed method, we fit a set of tetrahedra $\mathbf{T}$ in the *s-t* cut framework similar to [23]. We build a directed graph $G = (V, E)$ and apply the *s-t* cut to G. In graph G, each node V denotes a tetrahedron and each edge E between adjacent nodes V represents the facet shared by two adjacent tetrahedra. We add the source s and sink t nodes connecting to each node in V, which denote the interior and exterior of the surface respectively [9]. In G, we assign weights for each node and edge based on the energy function in Eq. 1. In the last step, the labeling process is accomplished by solving the minimum *s-t* cut on G.

$$E(\mathbf{T}) = E_{visibility} + \lambda_{quality} E_{quality} + \lambda_{lidar} E_{lidar} \quad (1)$$

Terms $E_{visibility} + \lambda_{quality} E_{quality}$ were first derived in [10] for range data. Since two sensors are incorporated in this framework, we introduce a new energy term E_{lidar}, which accounts for the different noise model of the camera and LiDAR to smooth the bumpy surfaces from noisy LiDAR measurements. Since we maintain the original formulation of the quality term from [10] in our pipeline, Section IV-C and IV-D provide details about our formulation of the visibility and energy terms. After calculating the energy of the whole graph based on Eq. 1, we apply the minimum *s-t* cut algorithm to determine the binary label of each tetrahedron.

C. Visibility information

Since we have camera and LiDAR measurements, the visibility term can be divided into two parts accordingly. For the camera visibility term, [7] has derived a weighting scheme for tetrahedra $\mathbf{T}$ to be consistent with the visibility of the camera by penalizing visibility conflicts. Specifically, α_{vis} is introduced as the unit confidence value for each ray from the center of camera to a visual point, which is proportional to the number of camera views seeing the point. For LiDAR points, α_{vis} is calculated in a different method since they are sparsely distributed in different locations across different scans. Due to the downsampling of the LiDAR point cloud during point insertion, we set α_{vis} for each point in $\boldsymbol{P}_d$ proportional to the number of points in r, which is calculated during downsampling for visibility consistency. This removes redundant points while keeping the additional visibility support that these points offer.

In three cases, the edge weight is incremented by α_{vis} for measurements and its corresponding sensor's center:

- For the ray from a sensor to a point inside its view intersecting the facet f_i shared by tetrahedra n_1 and n_2, the corresponding edge weight in the graph is incremented.
- For the cell directly behind the line of sight which is the line segment connecting the sensor and the point, its edge connected to the source node is incremented.
- For the cell containing the sensor, its edge connected to the target is incremented.

Algorithm 1 E_{lidar} calculation

```
for node i in G do
  for facet f of node i do
    if vertex of f from both LiDAR and visual points then
      weight(f) += γ
    else
      weight(f) += β
    end if
  end for
end for
```

D. LiDAR smoothing term

As mentioned in Section IV-A, LiDAR points have relatively larger noise than that of camera points. However, the visibility and quality terms cannot account for the different noise levels of LiDAR and camera measurements, since these two terms do not identify the source of vertices in tetrahedra. For example, Fig. 4 shows the scenario where both LiDAR and camera points exist for a surface. With original visibility and quality terms, the extracted mesh surface is bumpy and mostly determined by LiDAR points lying out of the surface because of their larger noise. Therefore, we add the LiDAR smoothing term to improve the quality of the surface where LiDAR and camera measurements overlap. We alternate the weights of tetrahetra which contain both LiDAR and camera points as vertices, hence these tetrahedra are more likely to be labeled as out of the surface by the graph-cut algorithm. As Fig. 4 shows, in zoomed view, only the facet in red of the tetrahedron is marked as inside. For the set of tetrahedra, the extracted surface is mostly determined by camera points closer to the real surface.

We examine the three vertices of each tetrahedron's facet to decide the term E_{lidar}. When three vertices are composed by LiDAR points or camera points only, we add a large constant value β to the edge in G that corresponds to the facet. The large value represents a large penalty to cut the edge off. On the other hand, when both LiDAR points and camera points are included in three vertices, we give this edge a small weight γ, which indicates higher probability

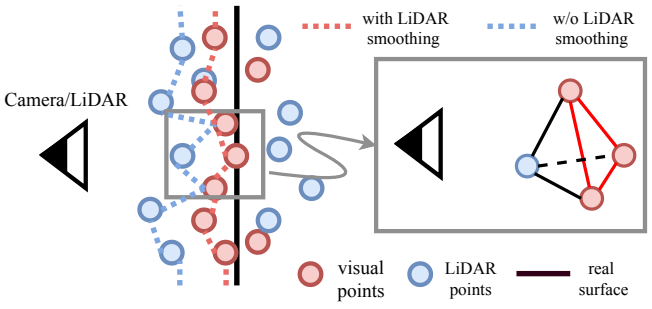
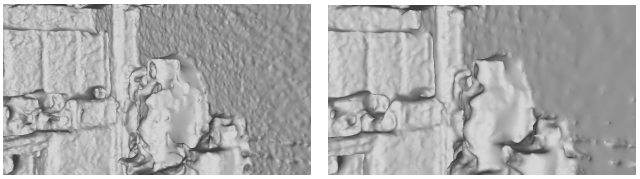
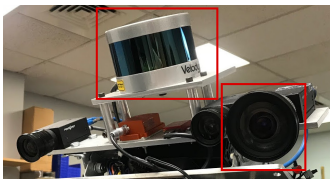


Fig. 4. Different LiDAR and visual points noise level around the ground truth surface and a zoomed view of a tetrahedron around the surface. In zoomed view, based on the term E_{lidar}, the facet with red edges is most likely to be labeled as inside since it has larger weight than the other three facets containing LiDAR points.

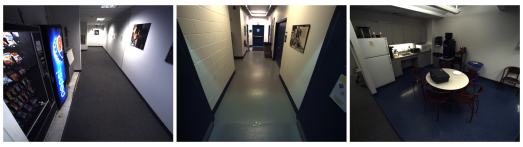


(a) Mesh result without E_{lidar} (b) Mesh result with E_{lidar}

Fig. 5. Comparison of reconstructed mesh with/without the LiDAR smoothing term (E_{lidar}). The smoothing term reduces noise from LiDAR measurements while preserving fine structures recovered from vision.



(a) The sensors within the red boxes are used to collect datasets



(b) Pictures of data collection scenes

Fig. 6. Our data collection device and experiment scenes. In (b), from left to right, data are collected in lift lobby, hallway, and kitchen.

to cut the edge off in *s-t* cut algorithm. (See tetrahedron in Fig. 4 and Algorithm 1). In this way, the labeling process is much more robust to noise around surfaces from LiDAR measurements as can be seen in Fig. 5.

In terms of the selection of γ and β in Algorithm 1, we need to consider the density of LiDAR points and camera points. In this paper, we choose $\beta = \frac{\alpha_{vis}}{2}$ and $\gamma = 1$, in which α_{vis} refers to the visibility value of LiDAR points and we choose $\alpha_{vis} = 32$ based on [10]. When β value increases, small structures or objects are wiped out in the model. It is important to keep the visibility term dominant in the energy function for correct geometry in the final model, and select a moderate α_{vis} and β.

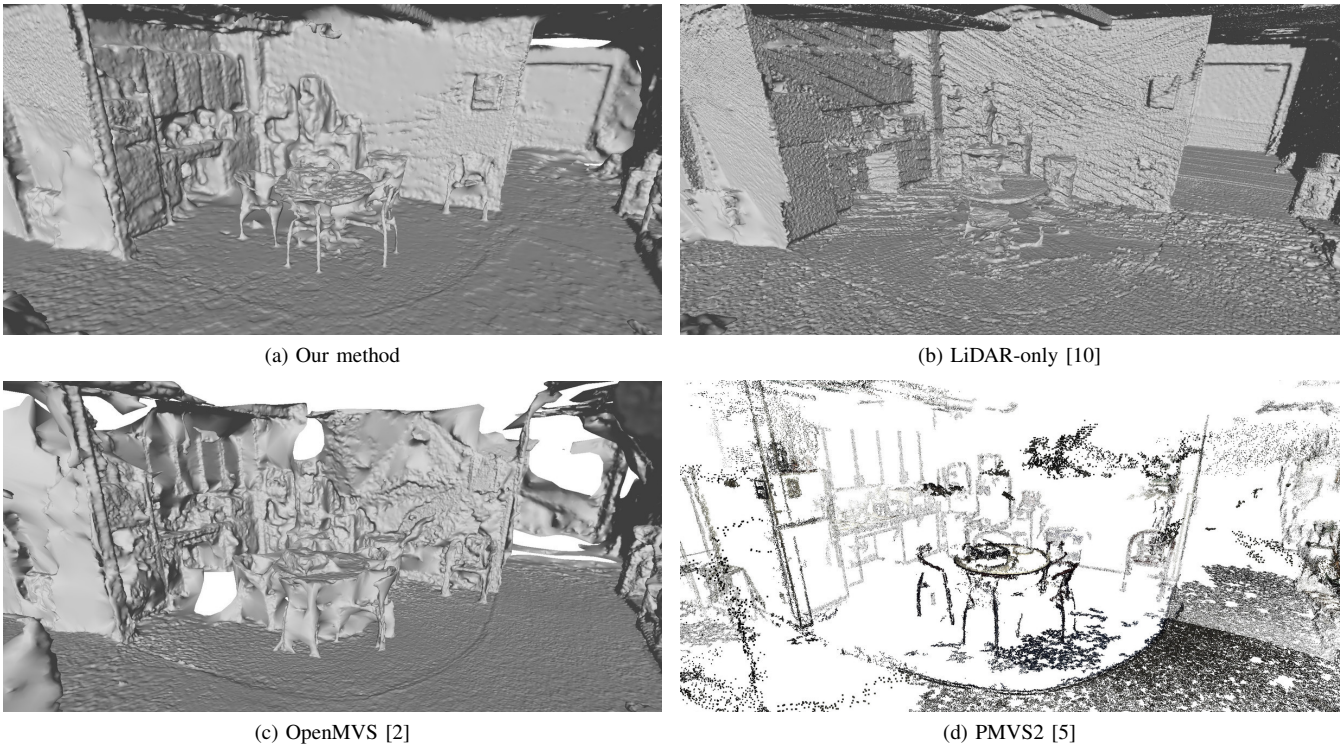
V. Experiments and Results

A. Implementation

We implement our method in C++ based on the open source library OpenMVS [2], by integrating depth map initialization, LiDAR measurements processing and the new formulation of the energy function into the pipeline. We use CGAL[1] to manipulate tetrahedra and Delaunay triangulation. All experiments are run on a Ubuntu desktop with Intel i7-7700 @3.60GHz CPU and 32GB RAM.

B. Experimental settings

Our method requires high-resolution images and LiDAR scans to reconstruct indoor scenes, but currently no public benchmark datasets containing such data exist. Therefore, we collect our own datasets using a custom-built sensor rig (Fig. 6a) with a Velodyne VLP-16 LiDAR and a FLIR



(a) Our method (b) LiDAR-only [10]

(c) OpenMVS [2] (d) PMVS2 [5]

Fig. 7. Shaded meshes of the kitchen dataset. By combining LiDAR and vision, our method preserves fine shapes while reconstructing textureless surfaces.

Grasshopper3 camera. We use a survey LiDAR scanner FARO Focus 3D to collect the ground truth model. We compare our method with state-of-the-art algorithms PMVS2 [5], OpenMVS [2], and a LiDAR-only method [10] on three indoor datasets.

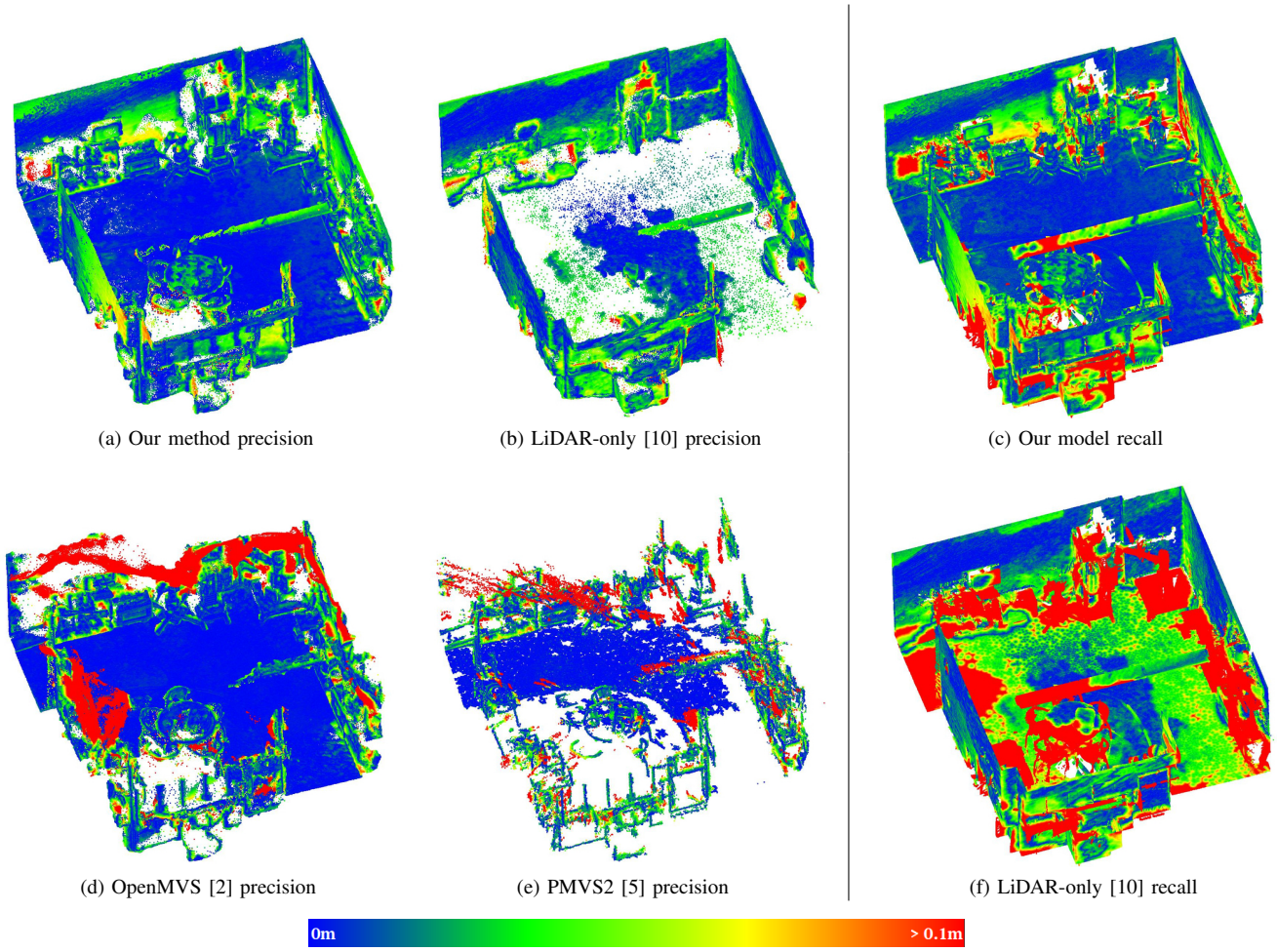
C. Evaluation

A qualitative comparison of reconstructed models is presented in Fig. 7. The mesh model from the LiDAR point cloud can accurately depict the structure of the room, but it has two shortcomings. First, it cannot keep thin structures or small objects in the final result because of its sampling sparsity. Second, with more LiDAR points added for reconstruction, more noise and artifacts from the sensor are introduced to the mesh model. This may be seen in Fig. 7b, where the floor and walls are bumpy and display repeated stripe-shaped artifacts that correspond to the laser scan lines. For OpenMVS (Fig. 7c), the geometry of the scene is wrong starting from the point densification step (see Fig. 3). The PMVS2 result (Fig. 7d) is relatively accurate in terms of patch positions, but only edges and corners are reconstructed in textured areas. The surface mesh from our method (Fig. 7a) can preserve the details of thin structures and small objects, and recover the textureless surfaces more accurately and smoothly.

Quantitatively, we evaluate our pipeline against other methods using metrics presented in [8]. The ground truth is provided in point cloud format, but our resulting model is a surface mesh. Hence, for comparison, we extract the vertices of tetrahedron facets which are labeled as inside the surface in the mesh as a point cloud. Since the camera and LiDAR sensors have different coverage of the scene, we manually bound our resulting point clouds to areas that are viewed by both sensors. After aligning the reconstructed model with the ground truth, we compare them according to the metrics in [8]. In [8], precision $P(d)$, recall $R(d)$, and *F-score* are defined for measuring the accuracy and completeness in the unified metric. Here d is a threshold of distance. $P(d)$ is the percentage of points in the reconstructed model of which the distance to their closest point in the ground truth model is smaller than d. $R(d)$ is calculated the other way around, by computing a similar percentage score from the ground truth model to the reconstructed model. Recall $R(d)$ indicates the percentage of the ground truth model that is captured by the reconstructed model. *F-score* is a summary measure, which is the harmonic mean of precision and recall given threshold d. In our evaluation, we set d to $0.05m$.

$$\textit{F-score} = \frac{2P(d)R(d)}{P(d) + R(d)} \tag{2}$$

Precision and recall of reconstructed models are visualized as false-color map in Fig. 8. We compare across three datasets, collected in lift lobby, hallway, and kitchen, respectively. The kitchen dataset is considered as the most difficult one since it contains many small objects, thin structures and occlusions. Lift lobby is the least challenging one because most surfaces are flat walls, the floor, or the ceiling. In the experiments with other pipelines, the parameters are set to default values as provided in open source code or the corresponding literature.



(a) Our method precision (b) LiDAR-only [10] precision (c) Our model recall

(d) OpenMVS [2] precision (e) PMVS2 [5] precision (f) LiDAR-only [10] recall

Fig. 8. (a)(b)(d)(e) False-color map for precision which evaluates reconstructed models against ground truth model are shown to the left. (c)(f) False-color map for our model and LiDAR recall. In (f), almost all compact objects in red are missing. The color representation of the error is shown in the bottom.

As Table I shows, the *F-score* of our method is better than the state-of-the-art pipelines on all three datasets. In terms of precision and recall, LiDAR-only and OpenMVS methods both do well in precision, but their coverages are mostly on large structures and textured objects respectively. From LiDAR recall error in Fig. 8f, we can see that the chairs around the table are missing in the LiDAR-only result, while the walls in OpenMVS [2] and PMVS2 results (Fig. 8d, 8e) are in poor reconstruction. So their recall percentage is relatively low in the indoor scenarios. Because of integrating both LiDAR and camera visibility information, our pipeline can reconstruct the scene with both clustered objects and textureless structures accurately.

TABLE I

PRECISION/RECALL/F-SCORE FOR DIFFERENT PIPELINES. BEST RESULT SHOWN IN **BOLD**.

Method	*Lift Lobby*	*Kitchen*	*Hallway*
Ours	**96.6**/**88.8**/**92.5**	**91.9**/**82.3**/**86.9**	93.1/**75.2**/**83.2**
OpenMVS	88.6/28.4/43.0	90.0/64.0/74.8	**93.7**/16.3/27.8
PMVS2 [5]	86.8/26.1/40.2	87.8/44.1/58.7	34.4/10.9/16.5
LiDAR [10]	95.7/85.1/90.1	86.2/69.1/76.7	91.6/66.3/77.0

VI. DISCUSSION AND CONCLUSION

In this paper, we present a novel LiDAR-integrated MVS dense reconstruction pipeline for indoor environments, which contains textureless areas and various compact objects. Integrating LiDAR measurements as priors during point densification enhances the accuracy of visual point clouds. By extending the graph-cut framework to accommodate both LiDAR and camera measurements, we fuse measurements from both sensors into a surface mesh. We demonstrate the advantages of our pipeline in indoor scene reconstruction, specifically the improvement of the reconstruction accuracy of smaller objects as well as textureless areas.

Limitations of our current system include the assumption of known camera and LiDAR poses. Errors in these pose estimates will impact the accuracy of our reconstruction.

In the future, our pipeline can be improved by incorporating more geometric features from both sensors such as lines, which is an active research topic in dense reconstruction [11]. In terms of mesh quality, a mesh refinement step can be added into the pipeline inspired by [23], which can refine the mesh vertices by minimizing the reprojection error from vertices to image pairs.

REFERENCES

[1] "CGAL: Computational geometry algorithms library." [Online]. Available: http://www.cgal.org

[2] "OpenMVS." [Online]. Available: https://github.com/cdcseacave/openMVS

[3] M. Bleyer, C. Rhemann, and C. Rother, "PatchMatch stereo – stereo matching with slanted support windows," in *British Machine Vision Conf. (BMVC)*, Jan. 2011.

[4] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

[5] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010.

[6] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *Intl. Conf. on Computer Vision (ICCV)*, Oct. 2007, pp. 1–8.

[7] M. Jancosek and T. Pajdla, "Multi-view reconstruction preserving weakly-supported surfaces," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, Jun. 2011, pp. 3121–3128.

[8] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[9] P. Labatut, J. Pons, and R. Keriven, "Efficient multi-view reconstruction of large-scale scenes using interest points, Delaunay Triangulation and Graph Cuts," in *Intl. Conf. on Computer Vision (ICCV)*, Oct. 2007, pp. 1–8.

[10] P. Labatut, J.-P. Pons, and R. Keriven, "Robust and efficient surface reconstruction from range data," *Computer Graphics Forum*, vol. 28, no. 8, pp. 2275–2290, 2009. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8659.2009.01530.x

[11] S. Li, Y. Yao, T. Fang, and L. Quan, "Reconstructing thin structures of manifold surfaces by integrating spatial curves," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2887–2896.

[12] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3D LIDAR and dense stereo," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 2181–2188.

[13] C. Martin and S. Thrun, "Real-time acquisition of compact volumetric 3D maps with mobile robots," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2002, pp. 311–316.

[14] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. Gool, and W. Purgathofer, "A survey of urban reconstruction," *Comput. Graph. Forum*, vol. 32, no. 6, pp. 146–177, Sep. 2013. [Online]. Available: http://dx.doi.org/10.1111/cgf.12077

[15] C. Premebida, L. Garrote, A. Asvadi, A. P. Ribeiro, and U. Nunes, "High-resolution LIDAR-based depth mapping using bilateral filter," in *Intl. Conf. on Intelligent Transportation Systems (ITSC)*, Nov. 2016, pp. 2469–2474.

[16] T. Schenk, "Fusion of LIDAR data and aerial imagery for a more complete surface description," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 34 (Part 3A)*, 2002.

[17] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1, Jun. 2006, pp. 519–528.

[18] S. Shen, "Accurate multiple view 3D reconstruction using patch-based stereo for large-scale scenes," *IEEE Trans. on Image Processing*, vol. 22, no. 5, pp. 1901–1914, May 2013.

[19] I. Stamos and P. K. Allen, "Integration of range and image sensing for photo-realistic 3D modeling," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Apr. 2000, pp. 1435–1440.

[20] L. Teixeira and M. Chli, "Real-time local 3D reconstruction for aerial inspection using superpixel expansion," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2017, pp. 4560–4567.

[21] S. Thrun, "Robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millennium*, G. Lakemeyer and B. Nebel, Eds. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=779343.779345

[22] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge," *J. of Field Robotics*, vol. 25, no. 8, pp. 425–466, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1002/rob.v25:8

[23] H. Vu, P. Labatut, J. Pons, and R. Keriven, "High accuracy and visibility-consistent dense multiview stereo," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 34, no. 5, pp. 889–901, May 2012.

[24] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: low-drift, robust, and fast," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2015, pp. 2174–2181.