Joint Surface Reconstruction from Monocular Vision and LiDAR

Zimo Li

CMU-RI-TR-19-57

August 2019

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Michael Kaess (Chair)
Simon Lucey
Kumar Shaurya Shankar

Submitted in partial fulfillment of the requirements for the degree of Master of Science.

In memory of my grandfather.

Abstract

In recent years, dense reconstruction gains popularity because of its broad applications in inspection, mapping and planning. Cameras or LiDARs are generally deployed for 3D dense reconstruction. However, current reconstruction pipelines based on cameras or LiDARs have significant limitations in achieving an accurate and complete scene reconstruction in certain environments due to the properties of LiDARs and cameras.

In this thesis, we propose 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 different scenes accurately, especially of scenes that are challenging for visual-only or LiDAR-only reconstruction. In particular, we exploit the advantages of the multi-view stereo algorithm in the reconstruction and integrate with the LiDAR measurements to further improve the robustness and accuracy. Current approaches employing cameras and LiDARs mainly focus on texture mapping with the color information from the camera images or improving camera depth estimation with the LiDAR. However, such methods only exploit the geometric information from single sensor measurements instead of fusing the geometric information from both sensors. In contrast, the proposed pipeline uses a two-stage approach to fuse the structural measurements from LiDAR with the camera images to generate a surface mesh. In the first stage, LiDAR measurements are integrated into a multi-view stereo pipeline to help with the visual point cloud densification. After combining the dense visual point cloud with LiDAR point cloud, a graph-cut algorithm is applied to extract a watertight surface mesh. To validate the proposed pipeline, we collect data from different kinds of scenes and compare results from our method with state-of-the-art reconstruction methods. The experimental results show that our method outperforms both the camera-only and LiDAR-only reconstruction pipelines in terms of accuracy and completeness.

Acknowledgments

I would like to thank my advisor Prof. Michael Kaess for his patience, suggestion and inspiration on my research. I also would like to thank my committee members Prof. Simon Lucey and Kumar Shaurya Shankar for their advice on my thesis. I am grateful for everyone in Robot Perception Lab (RPL), for all the discussion and motivation. And many thanks to my labmates in 2204 for happiness even in stressful time.

I would like to thank the Near Earth Autonomy for making the outdoor dataset available to us. And thanks to Cong Li, Weizhao Shao, and Srinivasan Vijayarangan for the indoor data collection.

I am deeply thankful to my family and Yueying Wang for their love and continuous encouragement throughout the two years. Finally, thanks to all my friends in Carnegie Mellon for helping me out in the two years, especially Chao, Chengqian, Yi, Jerry, Bobby, Bowen, Ziyi, and Jiayuan.

Contents

Chapter 1

Introduction

1.1 Motivation

In recent years, high-fidelity 3D reconstruction has gained popularity due to its growing applications. Accurate reconstruction of various environments such as outdoor or indoor scenes helps with different tasks. For example, Teixeira et al. [1] demonstrate building inspection using cameras on a drone; Goesele et al. [2] show reconstruction for cultural heritage preservation.

In computer vision, much work has been done on 3D dense reconstruction using cameras. Images provide plentiful geometric and color information, and several dense reconstruction pipelines are well developed, such as COLMAP [3], and OpenMVS [4]. But the performance of camera-based reconstruction algorithms highly depends on the lighting condition and the richness of textures. Therefore, these algorithms have limited performance in indoor environments, due to the presence of low-texture areas such as walls.

In the robotics community, 3D LiDARs are widely used for 3D perception and mapping. Comparing with cameras, LiDAR provides geometric measurements robustly, independent of the presence of visual features and variant lighting conditions. However, LiDAR measurements are much sparser than dense pixel measurements from the camera and it is not feasible to find direct correspondences between consecutive LiDAR scans. Other than that, the noise of LiDAR is relatively large for compact objects in a close range, which limits the reconstruction performance of LiDAR in indoor environments. As both camera and LiDAR have bottlenecks in different environments due to their sensor modality, fusing two sensor measurements together enables us to exploit the complementary nature of LiDAR and camera to robustly obtain 3D reconstructions in various environments.

The last decade has seen an increasing interest in combining LiDAR and camera together in the field of robotics. In the domain of reconstruction, the research that employs a combination of the two sensors typically derives the geometry from LiDAR only, while images are utilized for texture mapping. However, these methods are not fully leveraging the geometric information from both LiDAR and camera, since only color or texture information is extracted from images. The different sensor modalities make it difficult to fuse both kinds of measurements in a unified way. Therefore, applying the combination of LiDAR and camera in reconstruction remains challenging. In this thesis, we review previous LiDAR and camera dense reconstruction methods, and detail our proposed method to tackle the challenges in the joint reconstruction. We then evaluate the results from our work, and conclude with a discussion of future work.

1.2 Scope and Approach

In this thesis, we develop an offline dense surface reconstruction pipeline that combines geometric information from LiDAR and camera measurements to improve the robustness and accuracy of the reconstructed model. While the developed method is specifically designed for indoor environments, it also shows improvements of accuracy in complicated outdoor scenes. Generally it performs robustly in both indoor and outdoor environments.

For the dense reconstruction using a camera, multi-view stereo (MVS) methods have matured through the years. State-of-the-art dense reconstruction pipelines, such as OpenMVS [4], support reconstruction from images to dense mesh models with high accuracy. On the reconstruction benchmark Tank and Temple [5], OpenMVS is the top ranking open-source software. It is used in this thesis as the baseline method to provide the dense reconstruction from vision, and as a method for comparison in the experiments. Regarding LiDAR, although the state estimation is not our focus in this thesis, obtaining accurate poses is essential for 3D reconstruction. Therefore, we adopt LOAM [6] proposed by Zhang et al. to estimate poses from the LiDAR measurements. Besides poses of the sensors, extrinsic calibration is necessary for registration of measurements across sensors. We use line and plane constrained camera-LiDAR calibration based on [7]. The calibration result and analysis against other calibration algorithms are evaluated in the appendix.

Once we have the poses of LiDAR and camera in a sequence and the transformation between two sensors, we can consider how to fuse their geometric information together to recover a 3D model of the environment. We take the approach of using MVS as the baseline and adding LiDAR measurements to improve the dense reconstruction steps, since images have more dense information than LiDAR scans. To be specific, our pipeline can be divided into three steps: preprocessing, point densification and surface reconstruction. In the preprocessing step, since a single LiDAR scan is too sparse, we merge multiple LiDAR scans into a single point cloud using their poses to generate a prior map of the reconstructed environment. From this step on, we start to fuse LiDAR and camera measurements together. Since LiDAR measurements are different from camera images, which have feature correspondences across the sequence, it is difficult to match them together based on the features in the environment. Moreover, the different noise models between LiDAR measurements and camera measurements also lead to the problem. Therefore, we first merge LiDAR and visual point clouds together, then do a Delaunay tetrahedralization on the point cloud, and eventually use a graph-cut algorithm to extract the surface mesh from the point cloud. In the graph-cut algorithm, we formulate the term of the energy function to consider the difference between LiDAR points and camera points.

1.3 Contribution and Organization

In this thesis, we present an approach that combines LiDAR scans and camera images to achieve 3D surface reconstruction with improved accuracy and completeness. First, we address the challenges in the dense scene reconstruction using LiDAR or camera. We then provide our reconstruction pipeline that fuses geometric information from LiDAR and camera measurements to recover dense surfaces. The main contributions of this thesis are as following:

- The state-of-the-art vision-based reconstruction pipeline struggles on non-texture areas while LiDAR-based mapping lacks for detail and color information in the reconstructed model. We propose a novel approach to combine them together and exploit their complementary nature to improve the reconstruction result in both indoor and outdoor environments comparing with vision-only or LiDAR-only methods.
- To combine the LiDAR and camera information together, the difference of LiDAR and camera noise model makes it difficult to integrate two measurements for reconstruction probabilistically. We formulate the problem by putting their measurements into a unified Delaunay tetrahedra and using a graph-cut algorithm to solve based on the sensor property.
- To evaluate the performance of the proposed reconstruction pipeline, we collect real-world data to experiment with our pipeline and compare with the results from current state-of-the-art methods. The results overall show improvements in both indoor and outdoor datasets. We provide detailed analysis and visualization to illustrate the advantages and shortcomings of the proposed pipeline.

The thesis is organized as follow: Chapter 2 discusses background and related work. Chapter 3 provides the preliminaries for the graph-cut algorithm and the connection between graph-cut and surface reconstruction. Chapter 4 introduces the proposed surface reconstruction pipeline using LiDAR and camera, including detailed derivation and evaluation. Eventually, in Chapter 5, we summarize our contributions and discuss future work. In the Appendix 5.2, we introduce the calibration algorithm for LiDAR-camera calibration and the comparison between two different calibration algorithms.

Chapter 2

Background and Related Work

In this chapter, we classify the related work based on the sensor type which they use in their methods: LiDAR mapping, camera-based reconstruction and methods involving both sensors. Over past decades, an extensive amount of literature have been produced in the reconstruction or mapping fields with focus on different environments and different tasks. We will summarize their advantages and shortcomings in order to highlight the contribution of this thesis.

2.1 LiDAR Mapping

For LiDAR-only methods, 3D reconstruction is usually formulated as a SLAM (Simultaneous Localization And Mapping) problem, which is constructing a map while localizing the sensor itself on the map. Since LiDAR scans consist of sparse points in the space, the representation of the LiDAR map is limited. One straightforward representation is a registered point cloud. An example is shown in Fig. 2.1. Zhang and Singh [6] propose to use geometric features such as edges and planes in consecutive LiDAR scans to estimate the LiDAR odometry, and register LiDAR points using the estimated odometry. This idea is generally utilized or extended in [8, 9, 10]. In these systems, the map is represented by a sparse point cloud. The other representation adopted in LiDAR mapping is the surfel-based map, which is often used in the RGB-D camera mapping literature. In [11] and [12], LiDAR measurements are represented as a surfel map which realizes probabilistic fusion. Using surfels enables the data association between closest LiDAR points in consecutive LiDAR scans to adjust the surfel position and normal, although there is no explicit point-to-point correspondences between individual LiDAR scans. The surfel map is a more suitable representation than a point cloud since the radius of a surfel represents the uncertainty and it supports the dense visualization of surfaces. However the evaluations of these works focus more on the SLAM metrics, which are on the error of the trajectory, instead of the accuracy of the geometry reconstruction. Besides, there is research that use voxel grid to represent the LiDAR map, such as [13] and [14], but these works mostly focus on segmentation or planning.

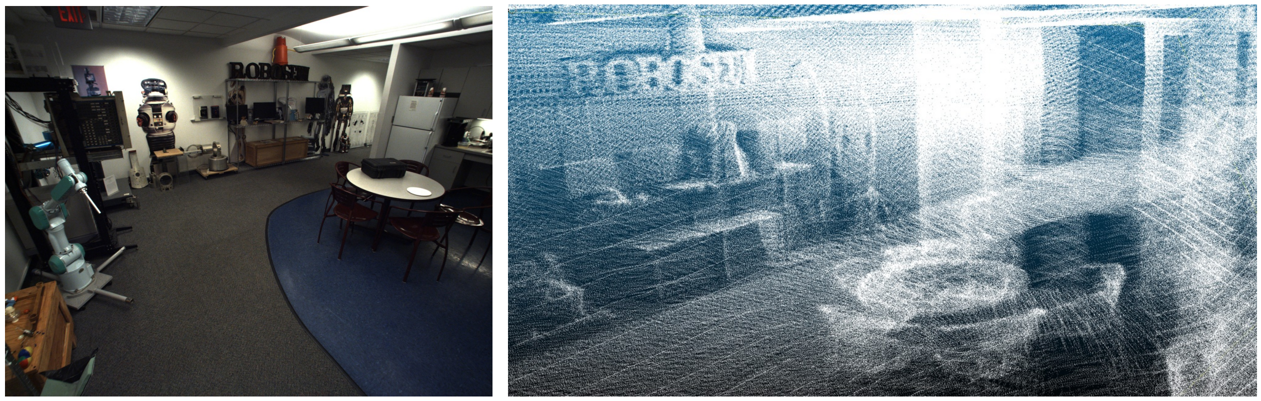

(a) Original image (b) LiDAR map

Figure 2.1: An example of a registered LiDAR point cloud in a kitchen. It shows the artifacts of LiDAR scan lines and the sparsity of the LiDAR map. The poses used for LiDAR map are from LOAM [6].

2.2 Vision-based Reconstruction

For vision-only reconstruction, a number of multi-view stereo (MVS) algorithms have been developed in recent years. In this section, we focus on the approaches which are designed or applicable for scene reconstruction. To distinguish from object-focused reconstruction, which is usually under full visibility without any occlusion, scene reconstruction has the following properties: the reconstruction target may be within a cluttering environment and the view points may be limited [15].

The authors of [16] propose the MVS reconstruction pipeline based on depth map estimation and merging. Although the method is applied to object reconstruction, current state-of-the-art pipelines, such as [3, 4], adopt its depth map based method but with more robust system design and implementation to adapt to images taken from different scenes. Furukawa and Ponce [15] develope a patch-based MVS pipeline (PMVS) to reconstruct compact objects. Instead of estimating the depth maps directly, PMVS estimates the 3D positions of patches from the images. Since PMVS depends on finding pixel level correspondences across images to define patches, low texture environments result in low completeness maps. As shown in Fig. 2.2, wall areas, carpets and some other low texture areas are not reconstructed well. In [17], 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 and a mesh deformation step, Vu's pipeline improves the accuracy of the surface mesh and achieves impressive results in the outdoor scenes. However, its performance still relies on a number of features to extract a dense point cloud for generating a precise mesh. Therefore the indoor dataset is still challenging fot their pipeline. The authors of [18] introduce the latest MVS benchmark ETH3D which includes high resolution images input for both indoor and outdoor scenes with the ground truth 3D models. They evaluate several state-of-the-art scene reconstruction pipelines on their new dataset including [3, 15] and the results show that although the accuracy is high in the reconstructed area, all existing methods

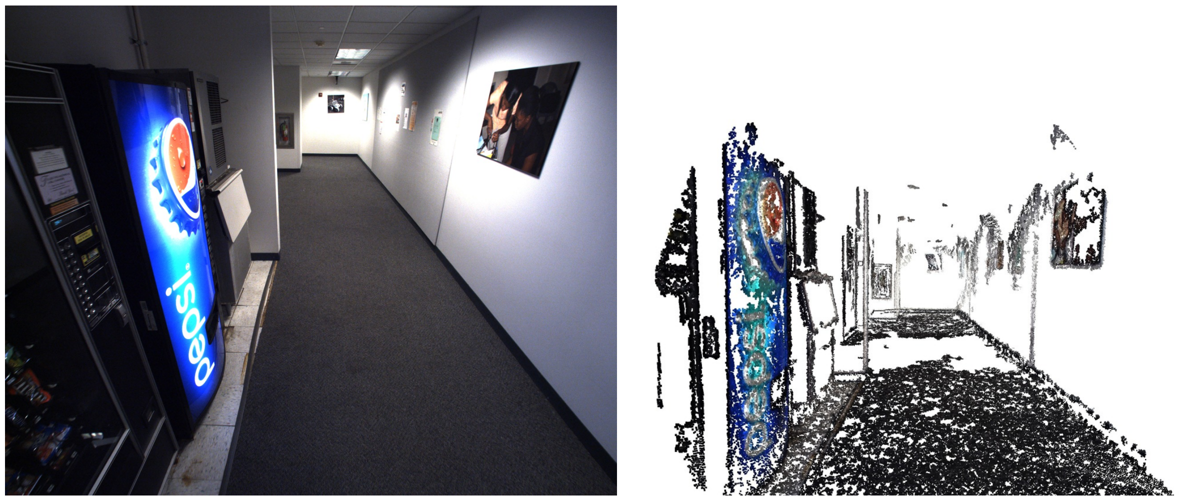

(a) Original image (b) Result from PMVS [15]

Figure 2.2: The result of running PMVS [15] in a hallway. All the non-texture area is not recovered in the model.

struggle to have a high completeness towards low texture areas.

Besides the geometric based methods, many learning based MVS algorithms have been proposed recently including [19**?**]. Although these methods achieve impressive results on public benchmarks, their generalization to the unknown scenes without training on specific datasets is not clear. Therefore in this thesis we will not compare against these kinds of methods.

2.3 Visual-LiDAR Mapping

Research with both LiDAR and camera gains popularity in recent years in the robotics community. In terms of the reconstruction, [20] and [21] map the image textures to the LiDAR registered point cloud. 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. [22] and [23] utilize 3D LiDARs and cameras to estimate dense depth maps. Although these works adopt probabilistic methods to generate accurate visual point clouds from a single image and LiDAR scan pair, LiDAR points are not directly integrated into the reconstruction stage. These methods achieve single view depth estimation, while we are interested in larger-scale reconstruction from a sequence of images and LiDAR scans.

2.4 Summary

To summarize, LiDAR-only methods have impressive localization accuracy and map consistency, but due to the sparse representation and the nature of sensor, it lacks for details in terms of compact objects. For MVS algorithms, some of the-state-of-the-art pipelines achieve impressive accuracy on different benchmarks. However, almost all of them rely on the visual features in the bundle adjustment or point cloud densification step. In low-texture or even non-texture scenes such as many indoor walls, the MVS algorithm shows low completeness or even wrong geometry. Little research has been done in dense reconstruction using both vision and LiDAR, and most research addresses texture mapping, which is similar to the LiDAR-only method but with the color texture from the images. Therefore, to improve the robustness and exploit the advantages of the two sensors, we propose to combine the LiDAR and MVS pipeline together in the thesis.

Chapter 3

Preliminaries

3.1 Surface Reconstruction using Graph-cut Algorithm

3.1.1 Graph-cut Algorithm

The graph-cut algorithm is from graph theory. In graph theory, a cut is to divide the vertices of a graph into two disjoint subsets [24]. In the flow network (example in Fig. 3.1), we have two special nodes, source (*s*) and sink (*t*), connecting to other nodes in the graph as Fig. 3.2 shows. A *s*-*t* cut is a cut that partitions the graph in order to place *s* and *t* in different subsets. And the *s*-*t* cut only cuts the edges directing from the source side to the sink side. A minimum *s*-*t* cut is a *s*-*t* cut that minimizes the total weights of edges on the cut.

Finding the minimum cut in a flow network has been proven to be equivalent to finding the maximum flow in the same network in [25]. Therefore the algorithms for finding the maximum flow are generally applied in the minimum cut problem. Many famous algorithms are proposed to solve the maximum flow problem, such as the Ford-Fulkerson algorithm [25]. In [26], Boykov et al. propose their own algorithm and compare it with other methods. Based on [26], the incremental breadth-first search [27] is introduced to improve the running time. In our system, the graph-cut problem is solved by [27]. After [27], Goldberg et al. propose a better method [28], which interested readers can refer to.

3.1.2 Surface Reconstruction Formulation

The surface reconstruction problem is defined as follows: given an unstructured point cloud and camera visibility information, extract the surface from the point cloud. Following [29, 30], we adopt a similar Delaunay method formulation, which represents the surface using Delaunay triangles in the space. The advantage of this method is that the resulting model is watertight and non-intersecting inside the surfaces. The Delaunay tetrahedralization is used on the input point

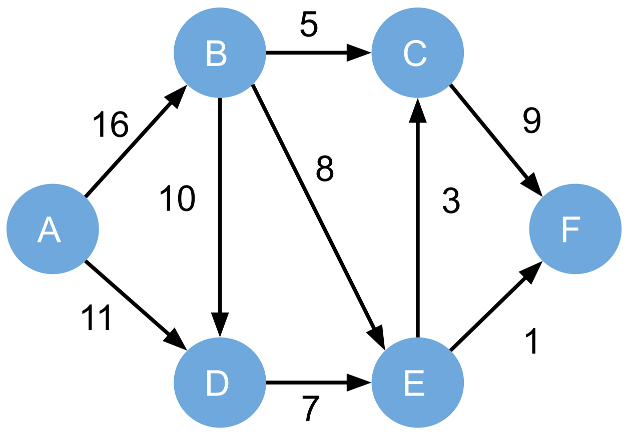
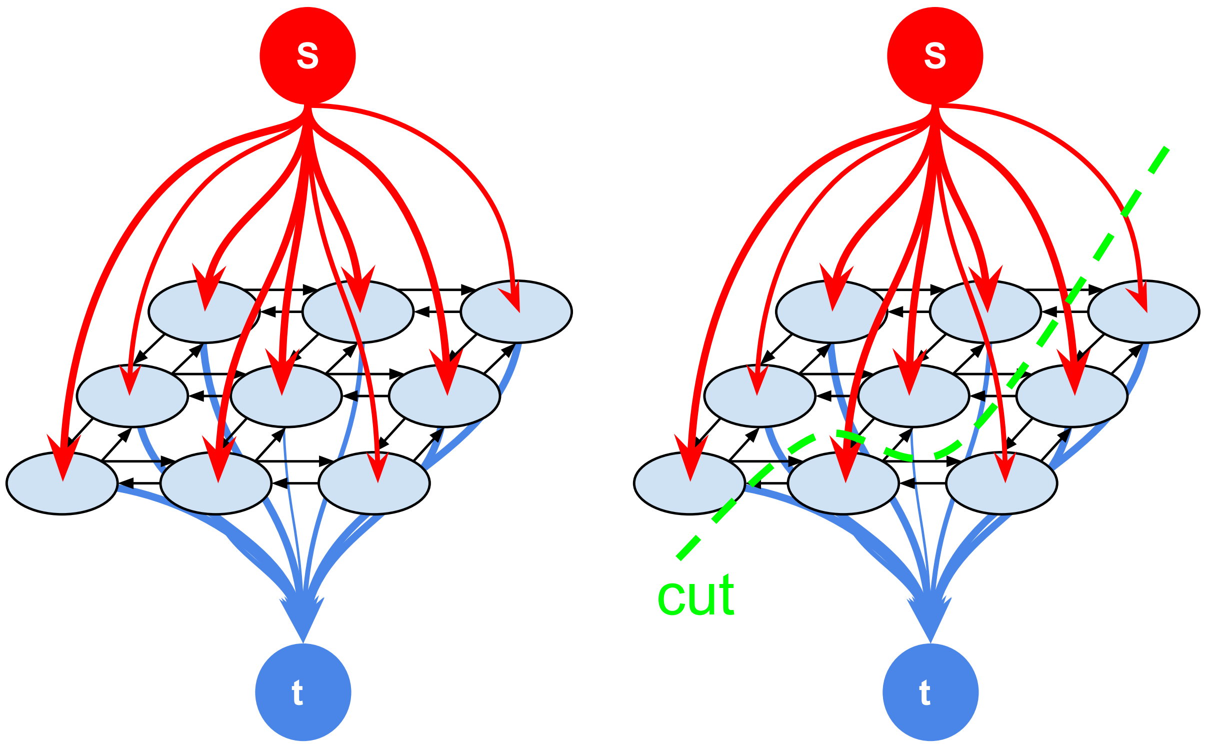


Figure 3.1: An example of a flow network



(a) An example graph

(b) A cut on the example graph

Figure 3.2: An example graph and cut in the computer vision formulation. All red, blue and black edges are edges in the graph. Red and blue represent the edge to the source or the sink. The thickness of the edge represents the cap.

cloud to divide the space into a set of Delaunay tetrahedra. Next, the surface reconstruction problem is converted to a Delaunay tetrahedra labeling problem. Each tetrahedron is labeled as inside or outside of the surface. After the label assignment, the triangular facets shared by a inside tetrahedron and an outside tetrahedron are written into the visualized mesh.

The core of the surface reconstruction is the formulation of the Delaunay tetrahedra labeling problem. We follow [29] to define an energy function on the surface and formulate the problem as an energy minimization problem. [31] first introduces the application of the minimum graph-cut algorithm in solving the energy minimization. Interested readers can refer to [31], which clearly demonstrates the connection between the graph-cut algorithm and the energy minimization problem on early vision problems. Therefore, the graph-cut algorithm here is applied to minimize the energy function across the graph. To represent the set of tetrahedra using a graph, each tetrahedron is a node and shared facets between neighboring tetrahedra are the edges. The weight of an edge, known as the cap of the flow, is assigned based on the energy function. Minimizing the energy function is equivalent to finding the minimum-cut in the graph, therefore the weighting scheme controls how the surface will be extracted. In this thesis, we adopt similar idea as in [29, 30, 32] to modify the weighting scheme for cutting the graph. The graph-cut algorithm in the surface reconstruction finds the global optimum for labelling the tetrahedra based on the energy function. But the tradeoff is the longer running time when more points are added in the space. The detail of weighting will be introduced in Section 4.

Chapter 4

Joint Surface Reconstruction from Monocular Vision and LiDAR

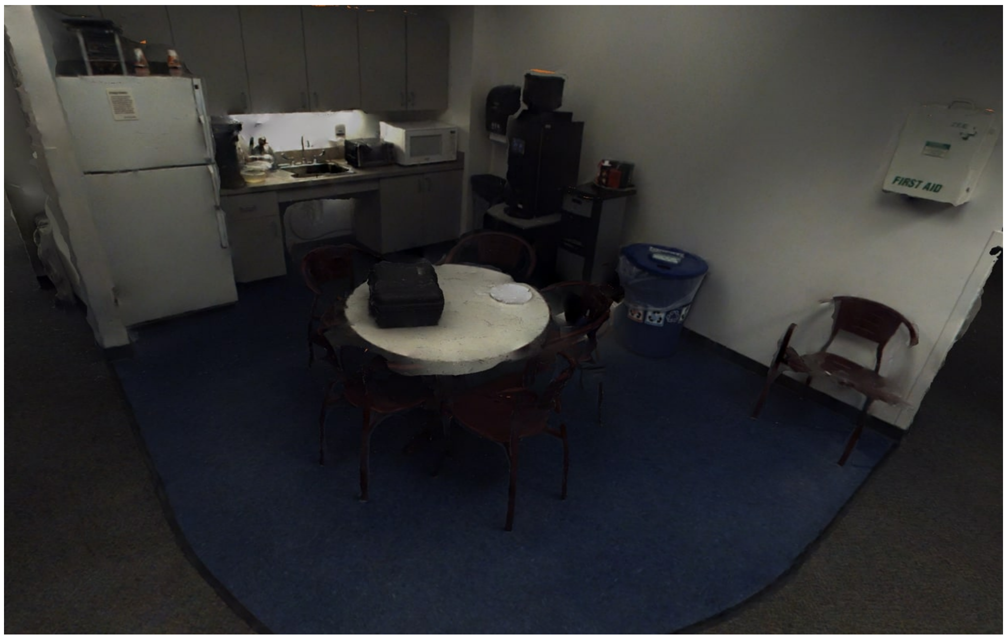


Figure 4.1: Textured mesh without shading from our reconstruction pipeline that exploits the complementary properties of cameras and LiDARs. Our method preserves fine shapes while reconstructing textureless surfaces.

In Section 2, we review the different frameworks on 3D reconstruction using LiDAR and camera, and analyze the advantages and drawbacks of these methods. In this chapter, we detail our proposed dense surface reconstruction pipeline, including the two main stages: the generation of the dense point cloud, and the mesh reconstruction using minimum *s*-*t* cut.

In the dense point cloud generation step, the depth-map-based point densification method is applied due to the following reasons. First of all, in an offline system, calculating depth map from many

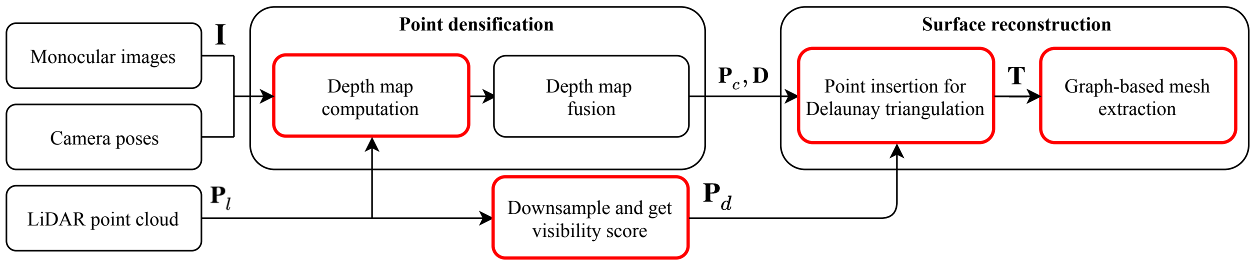


Figure 4.2: Diagram of our reconstruction pipeline. Boxes marked in red are the parts that we modify based on MVS pipeline. Notations are detailed in following sections.

different views improves the robustness and accuracy of the depth estimation. Secondly, a depth map approximately exploits all the pixel information provided by an image and its overlapping views. Lastly, considering the function of LiDAR measurements in the densification step, it is straightforward to use LiDAR points in the depth map computation. Next, we extend the baseline graph-cut algorithm introduced in Section 3.1.2 by formulating a new energy term accounting for the different noise models of the LiDAR and camera.

The pipeline is initially tested in indoor environments, and it shows robustness and accuracy outdoors as well. We distinguish the indoor and outdoor environments because they are mostly different in data collection, scene scale, and impacts on LiDAR and camera measurements. For camera, the lighting condition in the indoor scene has the property of Lambertian reflectance, but it varies outdoors. Regarding LiDAR, the indoor scene limits its range measurements (generally to less than 10m), while a longer range is normally better for LiDAR measurements. Besides, the distinction also depends on the complexity of different scenes, e.g. cluttered or open space. Details will be provided in the evaluation in terms of different datasets to test on.

4.1 LiDAR Scans and High-resolution Images Preprocessing

The goal of the preprocessing allows us to have registered LiDAR and camera poses under the same coordinate. In our pipeline, we use a LiDAR pose estimation pipeline [6] to get poses for LiDAR. Since the calibration between camera and LiDAR has been obtained from Appendix A, it is used to get the camera poses from the LiDAR poses. Therefore, we have the scale information from LiDAR odometry and set the camera poses as the prior before passing them into the bundle adjustment pipeline. We use OpenMVS [4] as our baseline method since it has implemented many methods for bundle adjustment and dense reconstruction in the consideration of robustness and accuracy, such as [32, 33, 34]. After running the full bundle adjustment, we obtain the adjusted camera poses. We use the calibration again to get the adjusted LiDAR poses from camera poses, and merge the LiDAR scans based on the adjusted poses. From the preprocessing, we produce the merged LiDAR point cloud, aligned poses of the two sensors and a sparse visual point cloud from bundle adjustment.

4.2 LiDAR-improved Point Cloud Densification

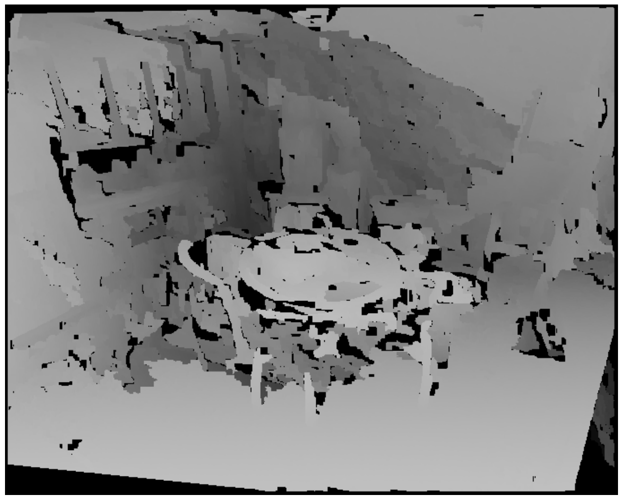
There are two stages in point cloud densification: depth map estimation and fusion. In the baseline, the depth map estimation is developed based on [34]. In [34], starting with random initialization of each pixel with a depth value, spatial and temporal propagation gradually refine the estimation, until the final convergence of the depth map. For multi-view cases, since a set of sparse 3D feature points are extracted in the bundle adjustment stage, it is utilized to initialize the depth for specific pixels. By performing 2.5D Delaunay triangulation inside the image plane, each pixel can be initialized by the distance from the camera center to the triangular facet that it lies on. The baseline method works well with the feature-rich areas because there are enough features to ensure a reasonably good initial estimation. However, for frames or environments which do not have enough visual features, such as wall areas, the depth estimation does not perform well as Fig. 4.3a shows. Therefore, we incorporate LiDAR measurements in the first stage to initialize the depth map estimation. 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 [34], 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:

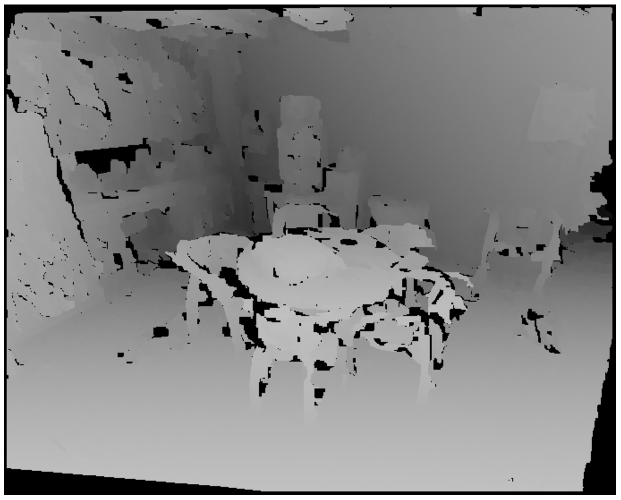
- *Several projected LiDAR measurements available*: only the closest measurement to the camera center is used for initialization, which accounts for occluding surfaces.

- *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 3D 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 [34]. Finally, we project all depth maps in $\boldsymbol{D}$ to 3D space and use the fusion method of [33] to reject inconsistent depths. Fig. 4.3a shows that OpenMVS [4] 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 [4]


(b) Depth map from our method

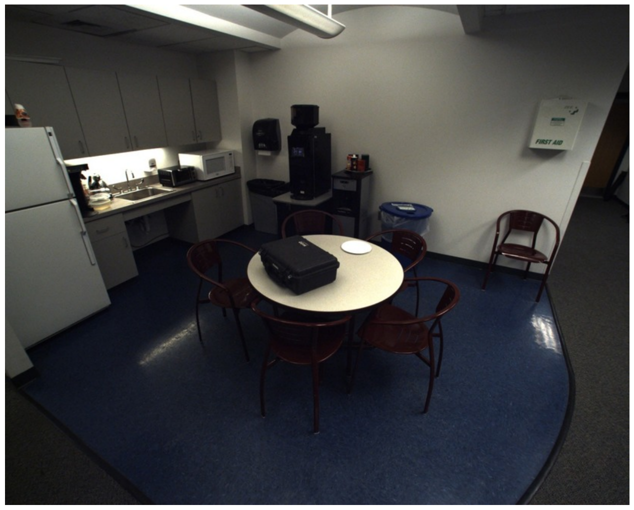

(c) Corresponding original image

Figure 4.3: 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.

4.3 Surface Reconstruction from Fused Measurements

Besides fusing LiDAR geometric information with visual measurements, LiDAR measurements help with the surface reconstruction as well. Although the LiDAR points are already utilized in the densification, their visibility information is not exploited. Therefore, we combine the LiDAR and camera measurements together and use their visibility information jointly to extract the surface. To be specific, after generating the point cloud $\boldsymbol{P}_c$ from the last section, we combine it with LiDAR points $\boldsymbol{P}_l$ into one point cloud $\boldsymbol{P}_{all}$. Then, we perform Delaunay tetrahedralization on $\boldsymbol{P}_{all}$ and convert the resulting tetrahedra to 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.

4.3.1 Point Insertion

Generally all points in $\boldsymbol{P}_c$ and $\boldsymbol{P}_l$ can be inserted in the tetrahedralization process. However, since the huge number of points result in a long running time, efficiency needs to be taken into account. The other drawback with a large number of points, especially LiDAR points, is that LiDAR points usually have larger noise[1] than triangulated camera points for short distance measurements. In that case, the LiDAR points make the reconstructed surface bumpy. Therefore, we design an optional downsample scheme to decrease the number of points inserted while maintaining the visibility information. In the insertion stage, 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 valid 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.

4.3.2 Graph-based Extraction of Surface Mesh

For 3D mesh generation, we use the graph-cut algorithm introduced in Section 3 to extract the mesh. In our proposed method, we fit a set of tetrahedra $\mathbf{T}$ in the *s-t* cut framework similar to [17]. We propose to add a term in the energy function to take account of the discrepancy in two sensor modalities.

$$E(\mathbf{T}) = E_{visibility} + \lambda_{quality} E_{quality} + \lambda_{lidar} E_{lidar} \tag{4.1}$$

Terms $E_{visibility} + \lambda_{quality} E_{quality}$ were first derived in [29] 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 [29] in our pipeline, Sections 4.3.3 and 4.3.4 provide details about our formulation of the visibility and energy terms. After calculating the energy of the whole graph based on Equation 4.1, we apply the minimum *s-t* cut algorithm to determine the binary label of each tetrahedron.

4.3.3 Visibility Information

Since we have camera and LiDAR measurements, the visibility term can be divided into two parts accordingly. For the camera visibility term, [32] has derived a weighting scheme for tetrahedra

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

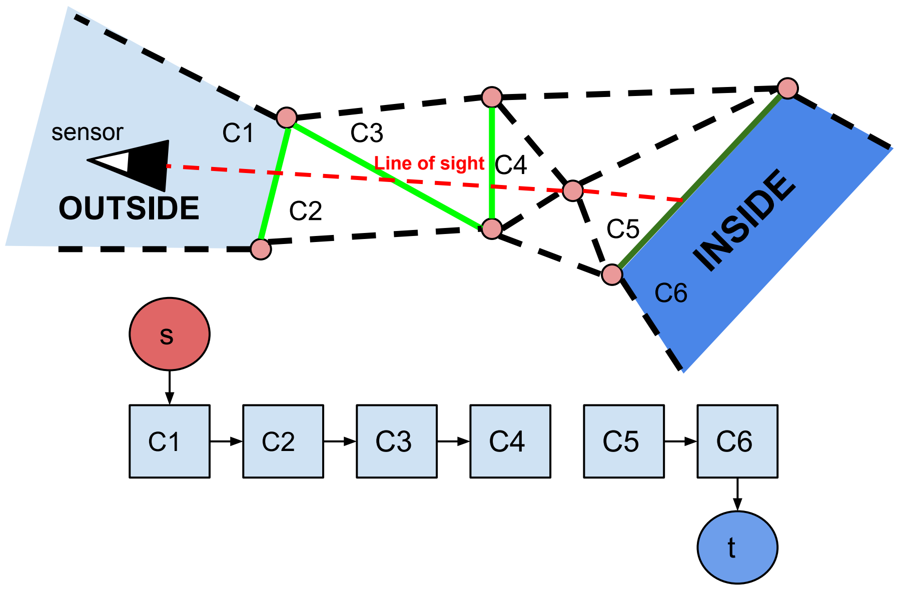


Figure 4.4: The example of how the visibility is assigned in the graph. The red dots in the top part is the points in the space. The lines connecting the red dots are facets of the tetrahedra, which is corresponding to the edges in the bottom part. The light green edges are in front of the sampled points and the dark green edge is behind the line of sight. The bottom part is the graph representation of the tetrehedra in the top. The cell containing the sensor is connected to source and the cell behind the the cell of the sample point is connected to the sink.

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. If the downsampling of the LiDAR point cloud during point insertion is performed, 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. If without downsampling, α_{vis} is set to a constant value for LiDAR points.

In three cases, the edge weight is incremented for measurements and its corresponding sensor's center in the graph of tetrahedron:

- 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 by α_{vis} multiplying with the inverse of distance from camera center to the intersection.
- 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 by α_{vis}.
- For the cell containing the sensor, its edge connected to the source is set to a large value.

The corresponding relation between the tetrahedra and the graph is shown in Fig. 4.4. In the space the tetrahedra has four facets, so the Fig. 4.4 is the simplified example.

4.3.4 LiDAR Smoothing Term

As mentioned in Section 4.3.1, LiDAR points have relatively larger noise than 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 a single tetrahedron. For example, Fig. 4.5 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 the edges connecting with the tetrahedron which contains 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. 4.5 shows, in the 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.

Algorithm 1 E_{lidar} calculation

```
1: for node i in G do
2:     for facet f of node i do
3:         if vertex of f from both LiDAR and visual points then
4:             $weight(\mathbf{f}) += \gamma$
5:         else
6:             $weight(\mathbf{f}) += \beta$
7:         end if
8:     end for
9: end for
```

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 to cut the edge off in *s-t* cut algorithm. (See tetrahedron in Fig. 4.5 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. 4.6.

In terms of the selection of γ and β in Algorithm 1, we need to consider the density of point cloud and the visibility information in the scene. The parameter selection essentially affects the final visualization of the mesh. Generally, 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

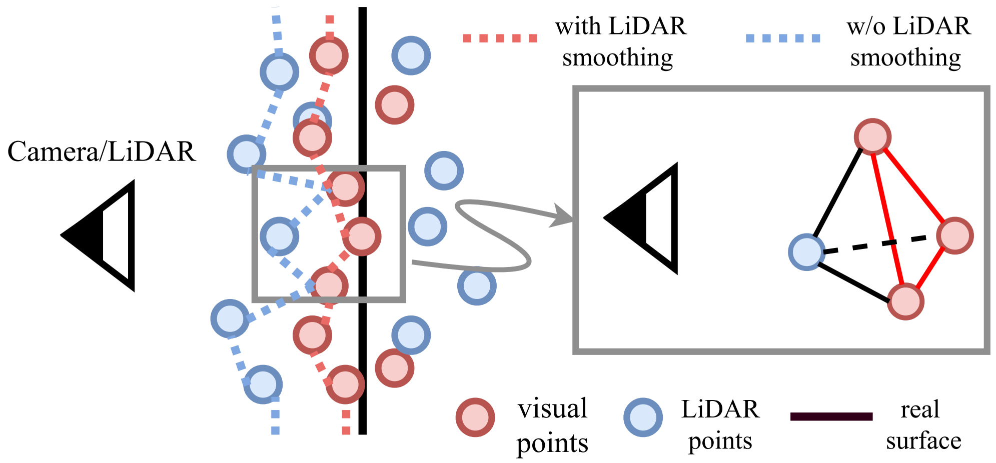
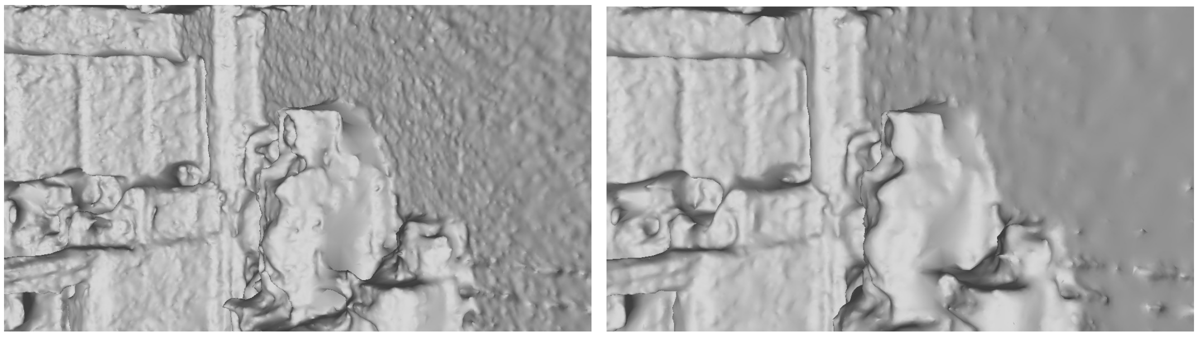


Figure 4.5: Different LiDAR and visual points noise level around the ground truth surface and a zoomed view of a tetrahedron around the surface. In zoomed view, based on the term E_{lidar}, the facet with red edges is most likely to be labeled as inside since it has larger weight than the other three facets containing LiDAR points.



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

Figure 4.6: 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.

correct geometry in the final model, and select a moderate α_{vis} and β. In the Section 4.4.5, we will discuss about how the weighting method works and show the comparisons between different weighting schemes.

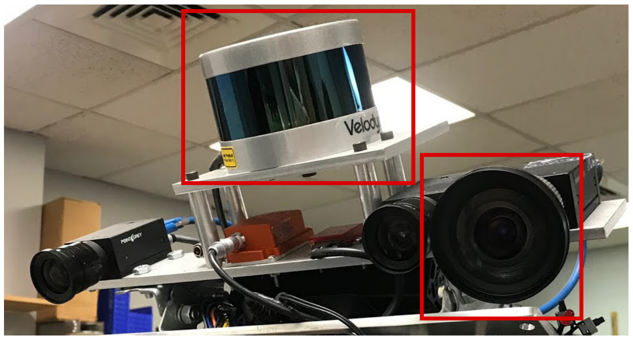
4.4 Experiments and Results

4.4.1 Implementation

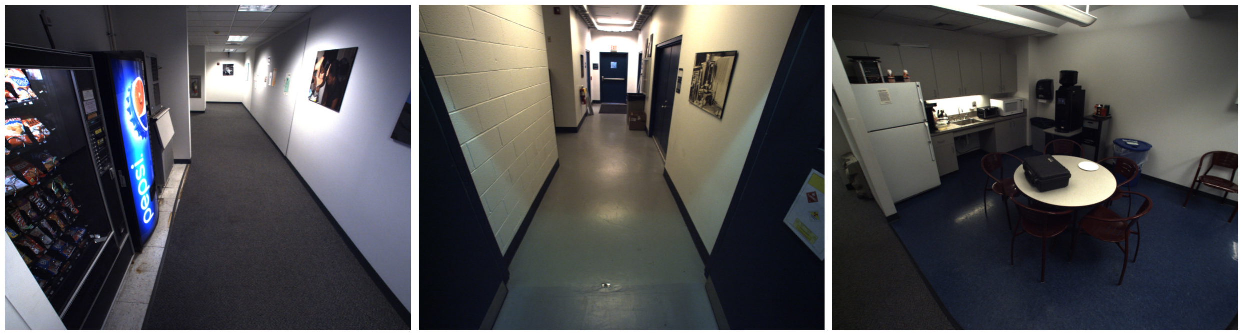
We implement our method in C++ based on the open source library OpenMVS [4], by integrating depth map initialization, LiDAR measurements processing and the new formulation of the energy function into the pipeline. We use CGAL [35] to manipulate tetrahedra and do Delaunay triangulation. All experiments are run on a Ubuntu desktop with Intel i7-7700 @3.60GHz CPU and 32GB RAM.

4.4.2 Experimental Settings

We first test the performance of our pipeline in the indoor scene. 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. 4.7a) with a Velodyne VLP-16 LiDAR and a FLIR Grasshopper3 camera. They are time synchronized. 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 [15], OpenMVS [4], and a LiDAR-only method [29] on three indoor datasets. We also extend the experiment to the outdoor environments using a large-scale outdoor dataset.



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



(b) Pictures of data collection scenes

Figure 4.7: Our data collection device and experiment scenes. In (b), from left to right, data are collected in lift lobby, hallway, and kitchen.

4.4.3 Indoor Evaluation

A qualitative comparison of reconstructed models is presented in Fig. 4.7. 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 is seen in Fig. 4.8b, where the floor and walls are bumpy and display repeated stripe-shaped artifacts that correspond to the laser scan lines. For OpenMVS (Fig. 4.7c), the geometry of the scene is wrong starting from the point densification step (see Fig. 4.3). The PMVS2 result (Fig. 4.7d) 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. 4.8a) 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 [5]. 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 shared by tetrahedra labeled as inside and outside of the surface 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 [5]. In [5], 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 $0.05m$.

$$\textit{F-score} = \frac{2P(d)R(d)}{P(d) + R(d)} \tag{4.2}$$

Precision and recall of reconstructed models are visualized as the false-color map in Fig. 4.8. 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.

As Table 4.1 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. 4.9b, we can see that the chairs around the table are missing in the LiDAR-only result, while the walls in OpenMVS [4] and PMVS2 results (Fig. 4.8c, 4.8d) are in

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 [4]	88.6/28.4/43.0	90.0/64.0/74.8	**93.7**/16.3/27.8
PMVS2 [15]	86.8/26.1/40.2	87.8/44.1/58.7	34.4/10.9/16.5
LiDAR [29]	95.7/85.1/90.1	86.2/69.1/76.7	91.6/66.3/77.0

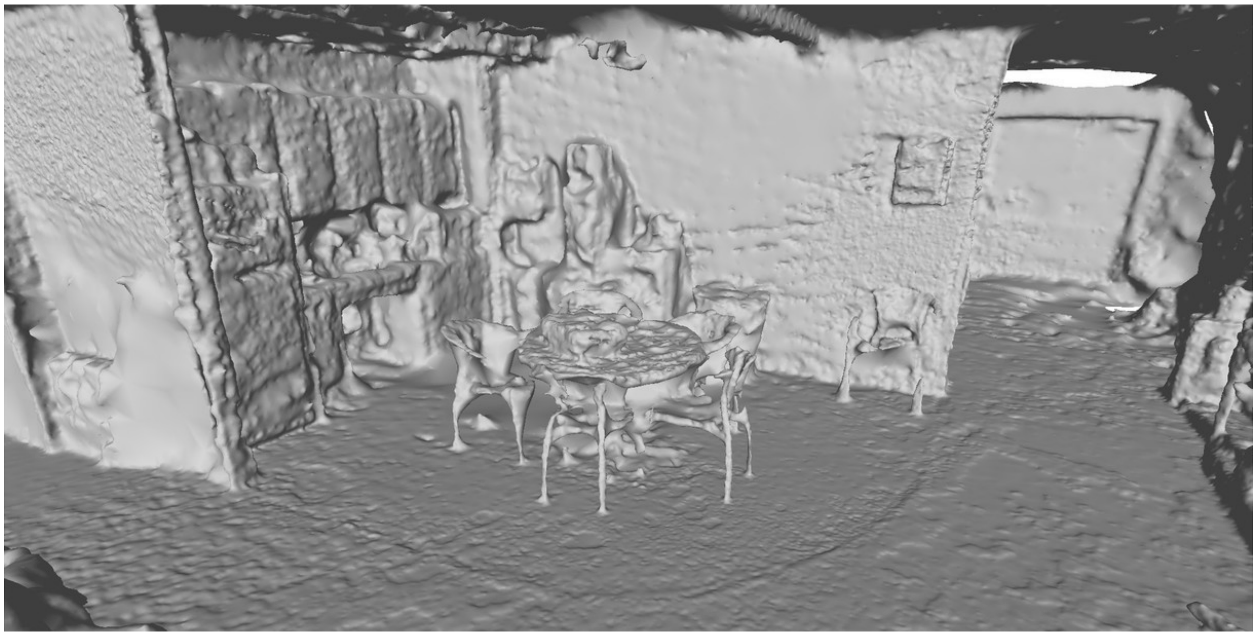
Table 4.1: Precision/Recall/F-score for different pipelines. Best result shown in **bold**.

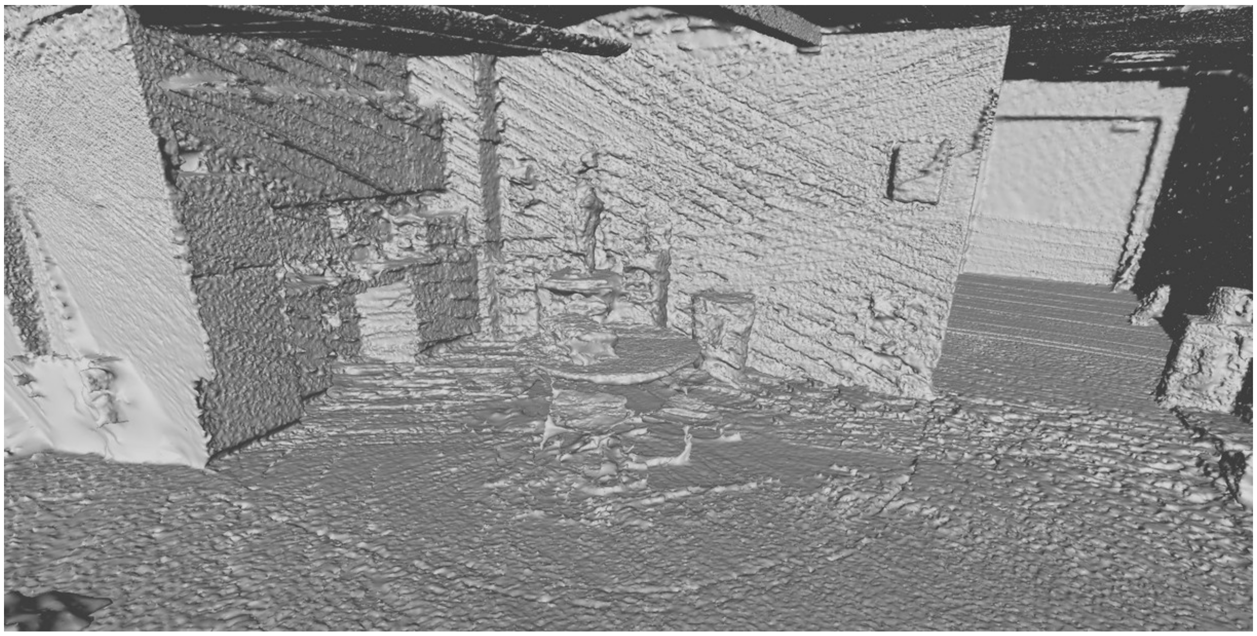
poor reconstruction. So their recall percentage is relatively low in the indoor scenarios. Because of integrating both LiDAR and camera visibility information, our pipeline can reconstruct the scene with both clustered objects and textureless structures accurately.

Besides the visualization and the quantitative evaluations, we dive deeper into the statistics of the result from different method on the kitchen dataset. As Fig. 4.10 shows, the yellow line in the graph is the average error of the result from different algorithms. We see that vision-based methods have more accurate measurements than our method and the LiDAR-only method. Especially for OpenMVS [4], its robust design rejects many outliers in the pipeline, which results in the high portion of points with error smaller than 2 cm. But the incompleteness in the model is not shown by the figure. For LiDAR measurements, the average error is 2 cm, and the distribution of the error is broadly spread since the error value of points with peak occurrence only occupies 3 percent of all points. Therefore, although the completeness is largely improved from vision-based methods, the accuracy is polluted by the LiDAR noisy measurements in some parts of the scene. The statistics results do not evalute against all points in the mesh since we reject the errors greater than 10cm. For example, in the kitchen dataset, the estimated depth around the walls are filtered out due to large error.

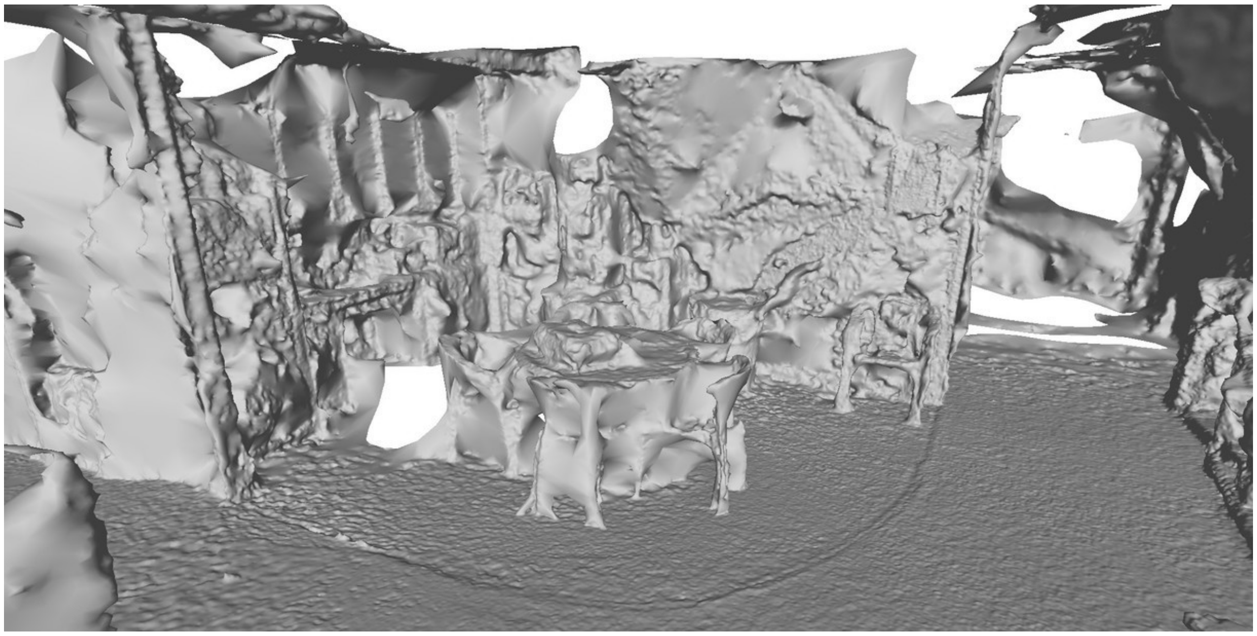
4.4.4 Outdoor Evaluation

Besides the indoor experiments, we extend our evaluation to an outdoor dataset as well. An outdoor dataset was collected by Near Earth Autonomy in a debris with the sensors on a drone. The images from the dataset are shown in Fig. 4.11. In this scene, extensive visual features are available for the camera images, but there are also many occlusions in the scene due to the image viewing angle. Different from previous experiment setting, a concatenated LiDAR point cloud is given beforehand, so we do not need to preprocess the LiDAR raw data. On the outdoor dataset, the LiDAR-improved point densification part does not show a boost in the point cloud density and accuracy comparing with the baseline MVS method due to the richness of the features. But since the data is collected by a drone, the views are mostly from top, and the occlusion is an inevitable problem without the LiDAR prior map. As Fig. 4.12 shows, many parts are occluded, and that will result in problems when extracting the surface mesh. Since no ground truth model is available for the outdoor dataset, we only compare the result from our pipeline to the camera-only and LiDAR-only method by the appearance.


(a) Our method


(b) LiDAR-only [29]

The mesh results are shown by Fig. 4.12. For the mesh from MVS, due to occlusions, there are many inconsistencies in the result. The unobserved surfaces are connected together such as the two-story platform and the side of the building. The mesh from LiDAR has better coverage of the scene because of the LiDAR's wide field-of-view. However, since the mesh extraction steps need line of sight to cut the space, the sparsity of the LiDAR prior map introduces the problem of weakly observed surface. Fig. 4.13 demonstrates the problematic surface. We visualize the


(c) OpenMVS [4]

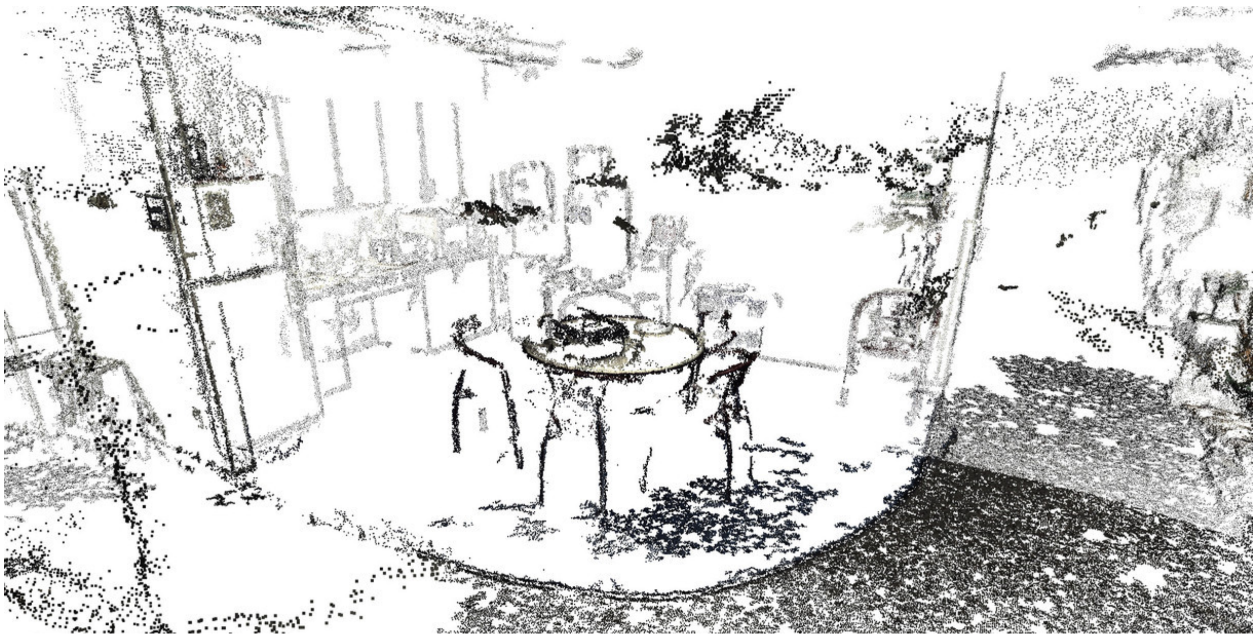
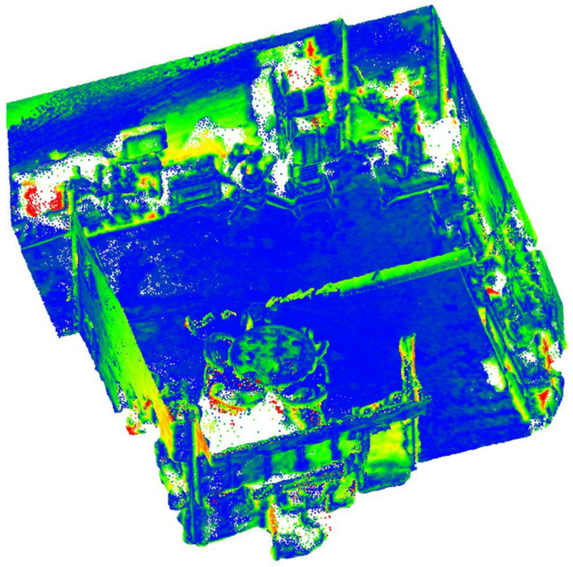
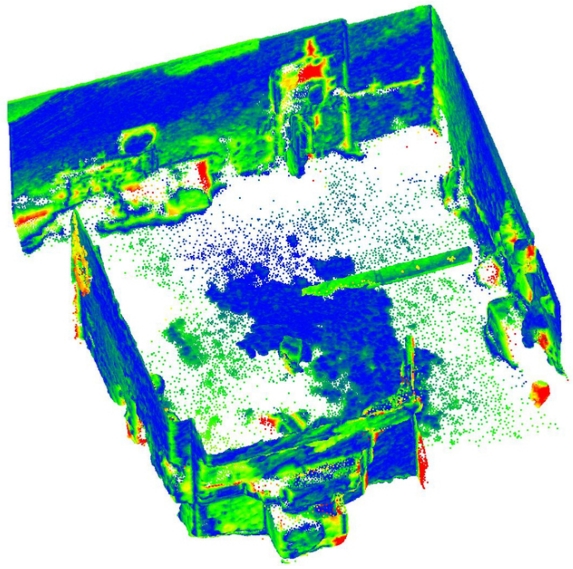
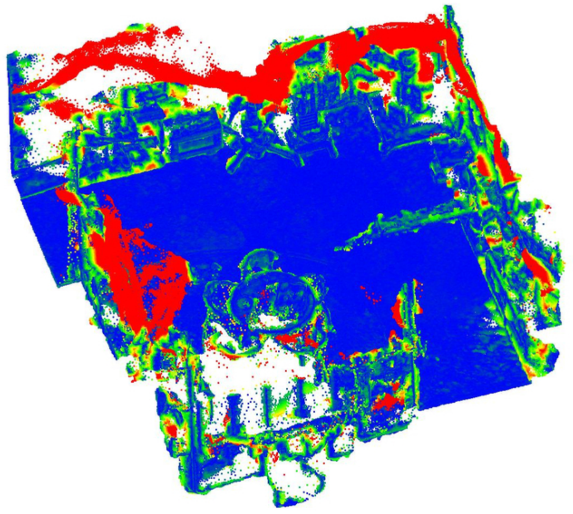

(d) PMVS2 [15]

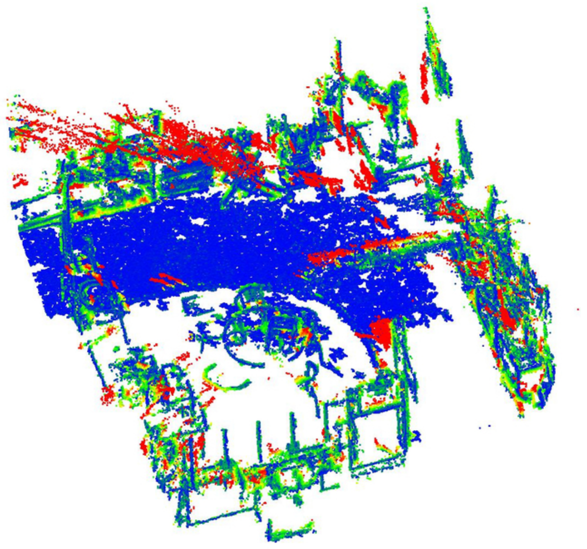
Figure 4.7: Shaded meshes of the kitchen dataset. By combining LiDAR and vision, our method preserves fine shapes while reconstructing textureless surfaces.

density of the points on different surfaces. As shown in the figure, all the points inside occluded areas have fewer than 8 neighboring points in a circle with 10cm radius, while other areas have higher density. This will introduce weakly supported surfaces in the mesh extraction stage since


(a) Our method precision


(b) LiDAR-only [29] precision


(c) OpenMVS [4] precision


(d) PMVS2 [15] precision

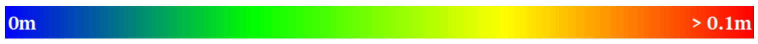


Figure 4.8: False-color map for precision which evaluates reconstructed models against ground truth model are shown to the left.The color representation of the error is shown in the bottom.

intuitively the sparse visibility for the occluded points cannot make the cut happen in the graph. Therefore, there are some missing parts in the LiDAR mesh, such as the occluded middle floor of the platform. Compared with the LiDAR-only result, our algorithm achieves a better reconstruction because we integrate the LiDAR and camera visibility information together, which is really helpful

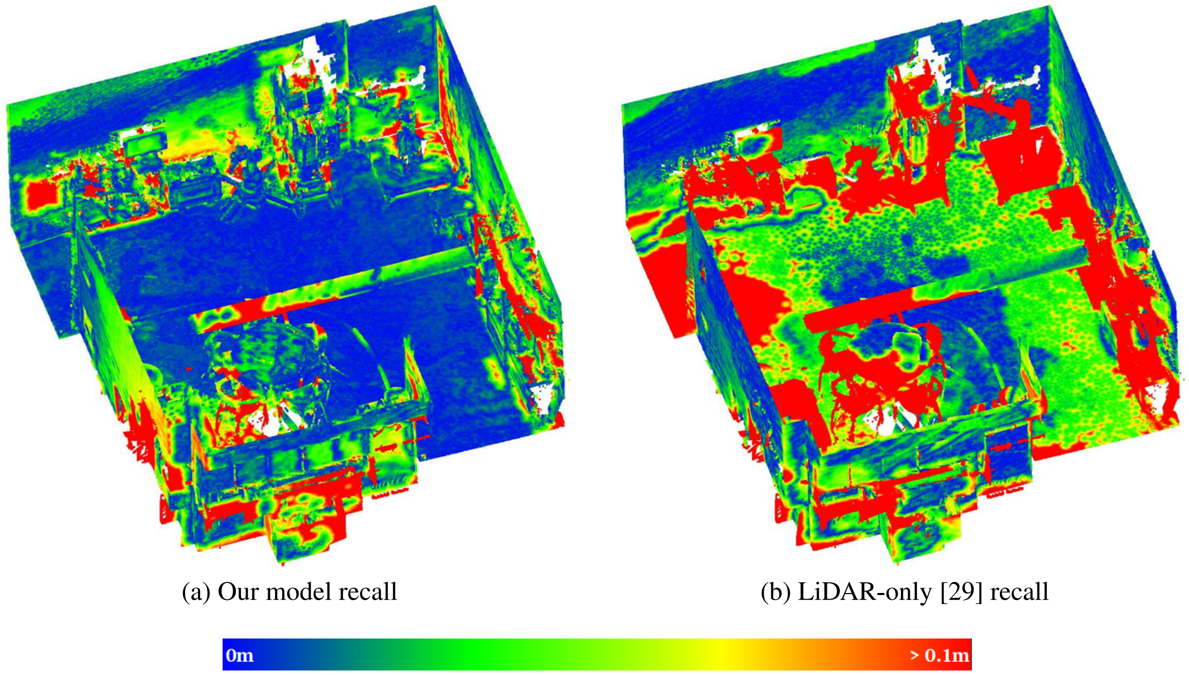


Figure 4.9: False-color map for our model and LiDAR recall. In LiDAR recall, almost all compact objects in red are missing. The color representation of the error is shown in the bottom.

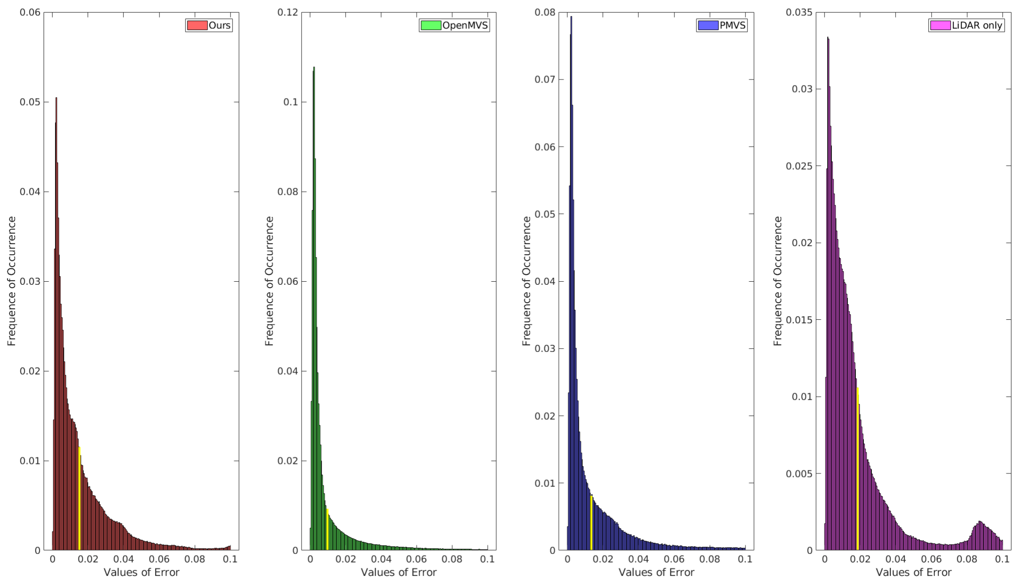


Figure 4.10: The distribution of the errors in different methods. In the figure, the yellow vertical line in the four individual subplots is the mean value of the error in their respective result.

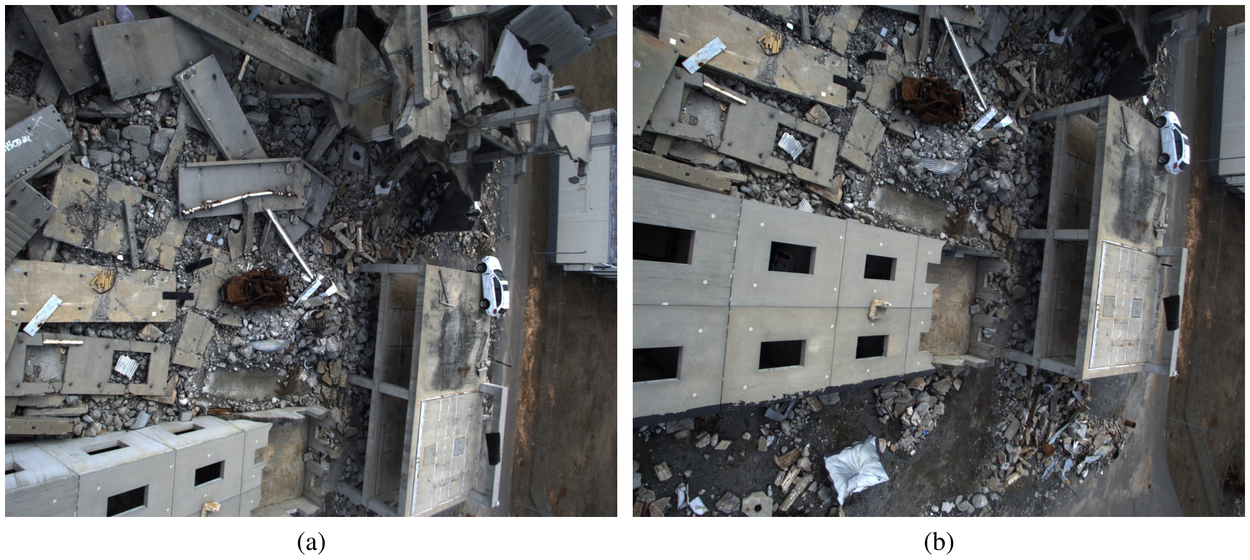

(a) (b)

Figure 4.11: The example images from NEA dataset

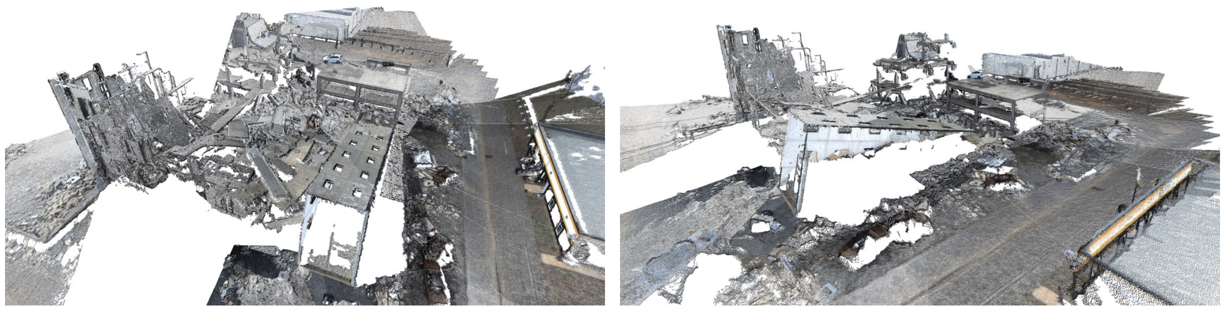


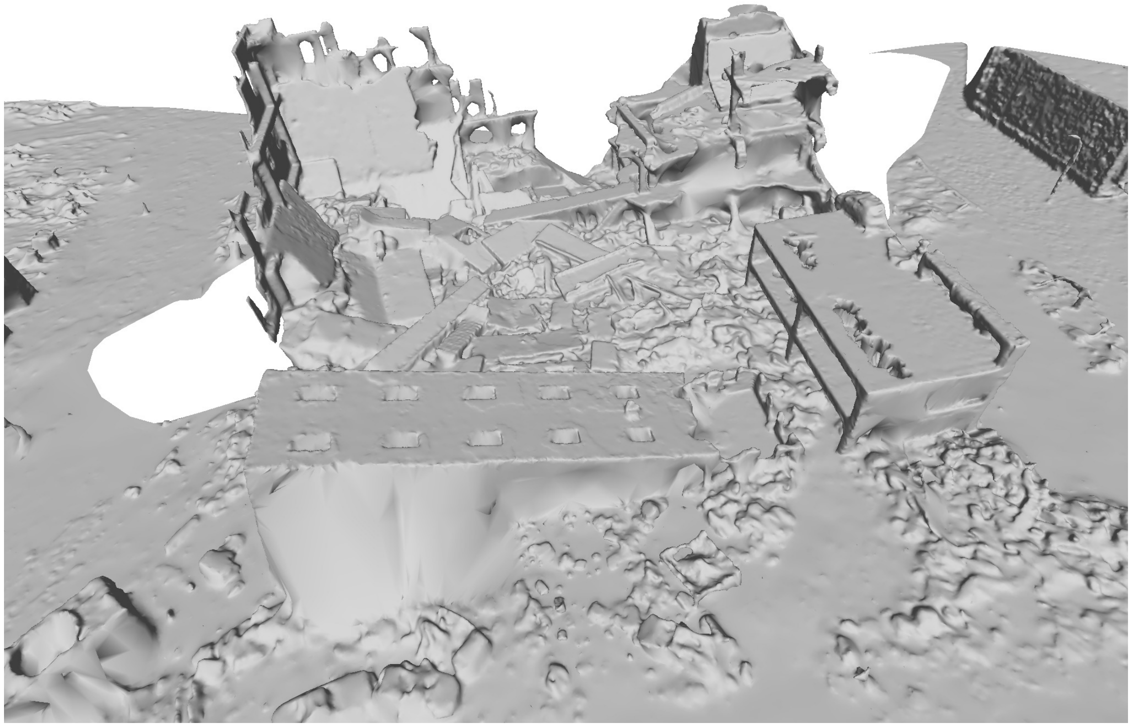
Figure 4.12: The dense point cloud from the pipeline. As shown in (b), the platform and the side of the building only has the top layer, and all other parts are occluded.

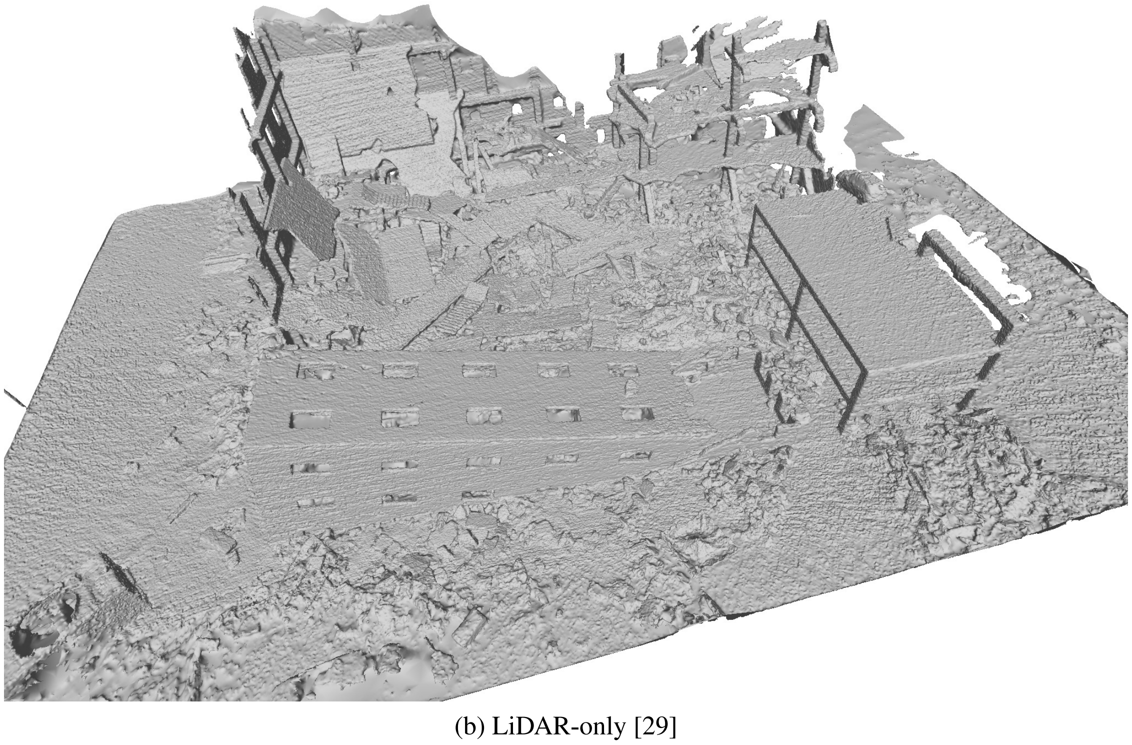
in a scene with occlusions. As shown in the comparison in Fig, 4.14 the platform from our pipeline is reconstructed well. We texture map the mesh from the proposed pipeline and the result is shown in Fig. 4.15. Some areas are not textured due to occlusion and limited field-of-view of the images.

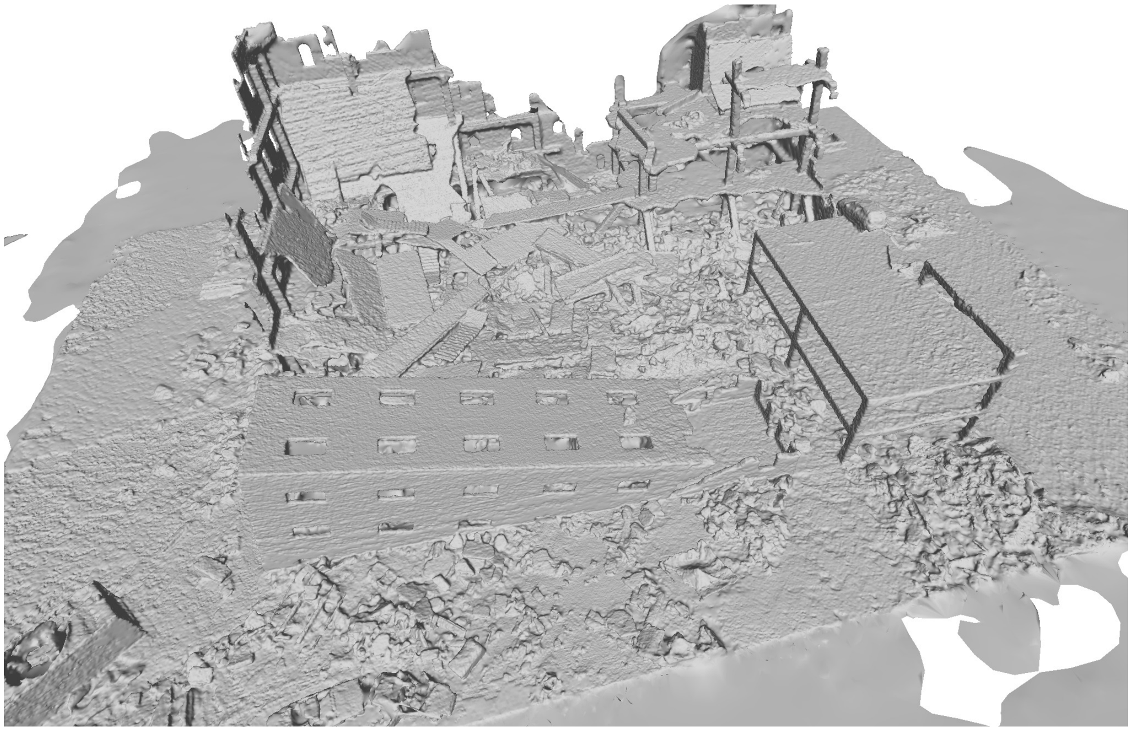
4.4.5 Discussion

Weight Selection

Although [29, 30, 32] all employ the graph-cut method in their pipelines, they are not explicitly reasoning about the selection of the parameters, such as α_{vis}. In this thesis, since we have different


(a) OpenMVS [4]


(b) LiDAR-only [29]



(c) Our results

Figure 4.12: Shaded meshes of the kitchen dataset. By combining LiDAR and vision, our method preserves fine shapes while reconstructing textureless surfaces.

α_{vis} setting for LiDAR and camera measurements, we investigate the selection of the parameters in the proposed pipeline. For reconstructions containing only single type of measurements, the weight selection does not change much of the cut position of the graph for visual point cloud since it usually has less noise compared with LiDAR points and most of the edge caps are proportional to the parameter α_{vis}. However, the mesh from LiDAR have different appearances with different weight selection due to its large measurement noise. Small experiments are conducted based on a part of the outdoor dataset. We qualitatively evaluate how the weighting affects the mesh results. As Fig. 4.16 shows, when α_{vis} of LiDAR points increases, more LiDAR points are kept in the final mesh but more noisy. Although there is not a quantitative guide on how to select α_{vis}, one important observation is that once the weight is beyond a specific value, the appearance of the mesh does not change much because almost all points are already vertices in the mesh.

After adding the LiDAR points to the visual point cloud, as Section 4.3.4 mentions, two parameters γ and β are added for the LiDAR smoothing term. In the experiments, we set γ to 1 and change the value of β to observe the change of appearance in the mesh result. Moreover, the experiments shown here are not for the best reconstructed result but for demonstrating the function of the weighting of the smoothing term. In the Fig. 4.17, we show the impact of a large LiDAR smoothing term weight in the mesh result. In the red bounding box, artifacts of LiDAR scan lines can be seen. With a larger weight, the artifacts areas are smoother in the Fig. 4.17b. But as shown, some other

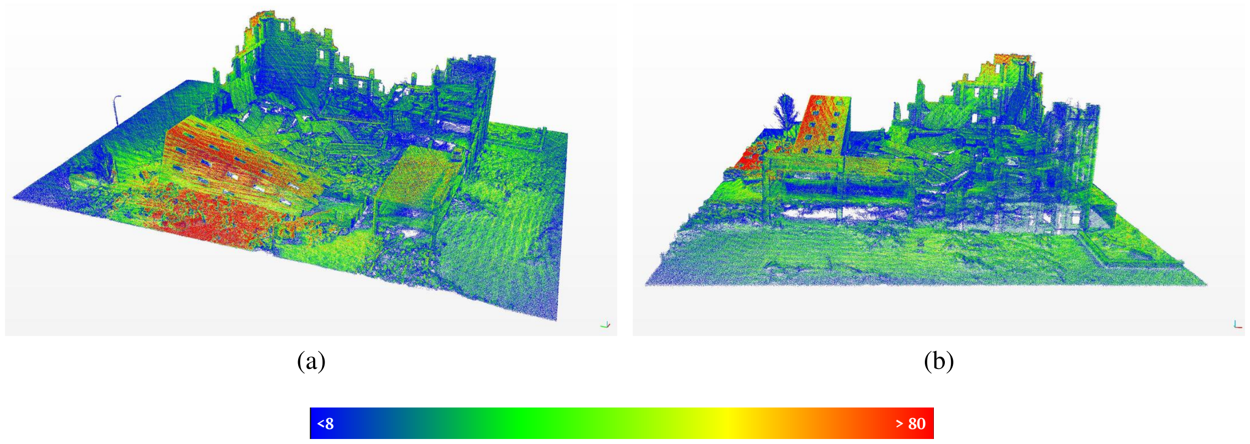


Figure 4.13: The visualization of the LiDAR prior map density. We use the number of neighbors to colorize the map. As the color bar shows in the bottom, blue represents points with 8 or fewer neighbor points in 10cm radius and red represents points with more than 80 neighbors in the radius of 10cm.

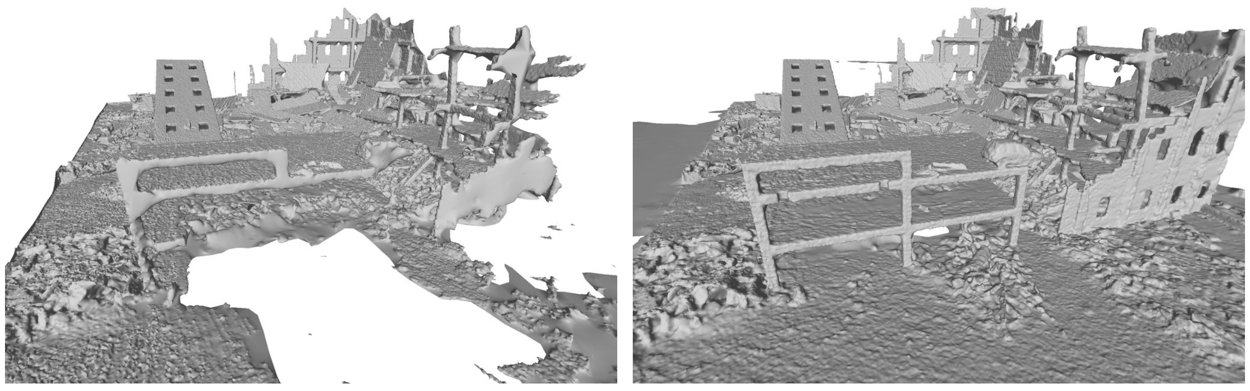
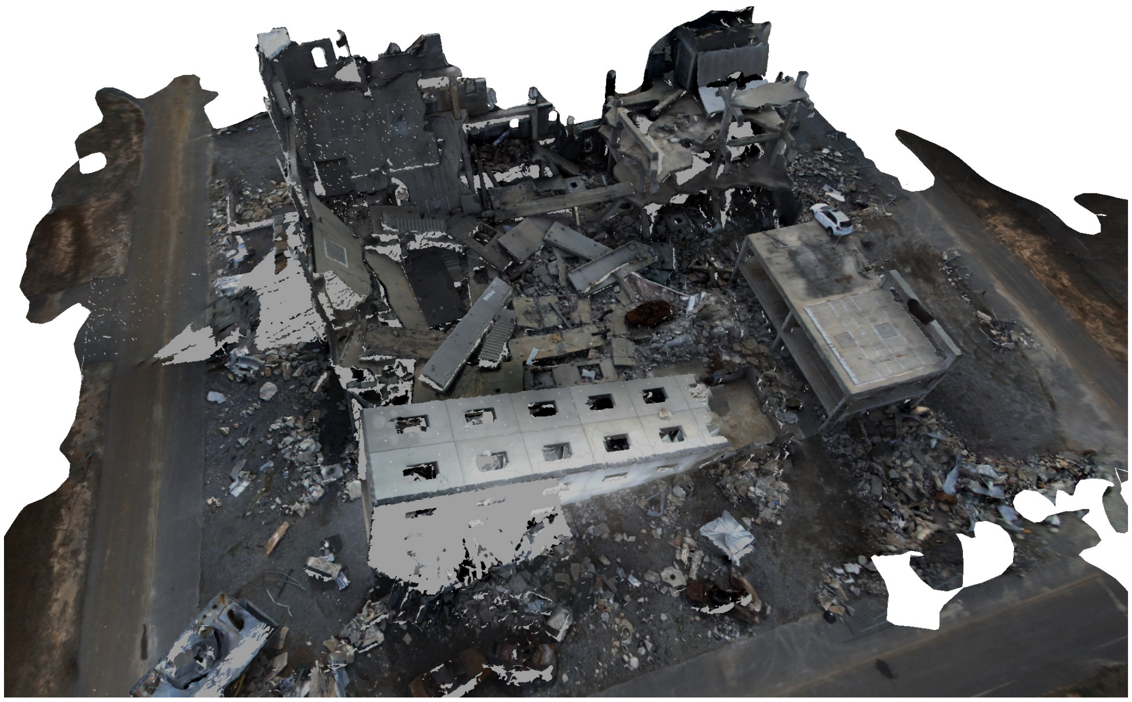


Figure 4.14: The comparisons between the LiDAR-only result and our result. The platform and the walls are recovered well by our method.


(a)

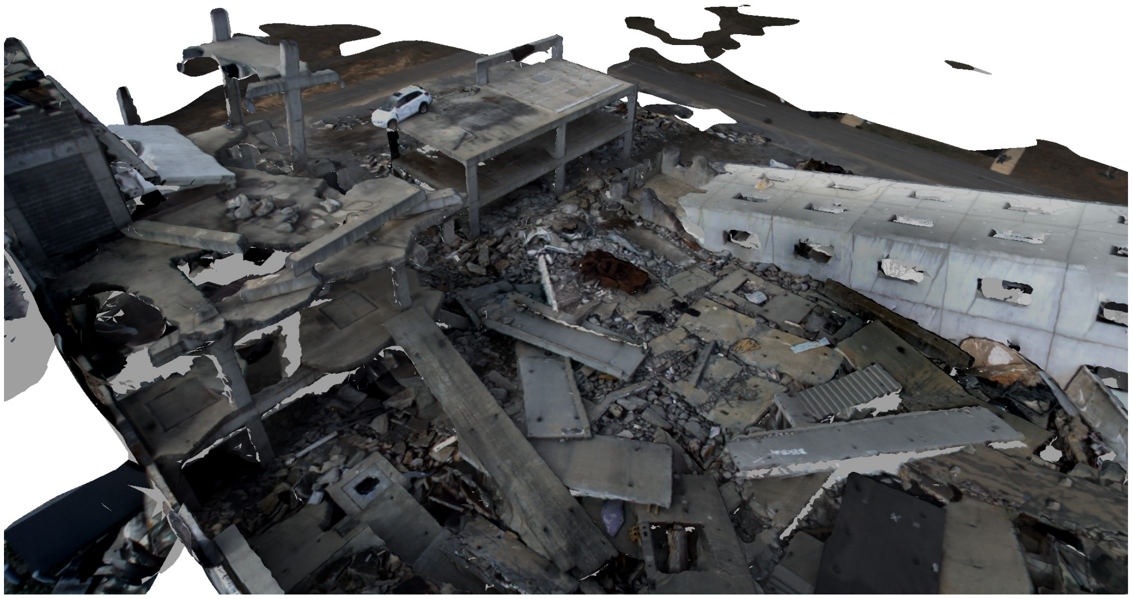

(b)

Figure 4.15: The textured mesh result from our method under no shading option. The gray areas in the figure do not have textures due to the limited field-of-view of the camera.

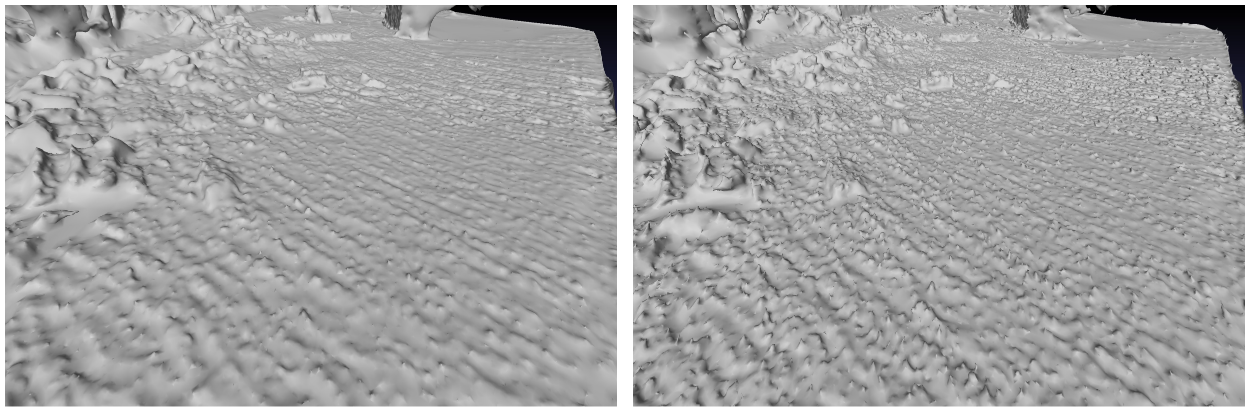

(a) Small weight (b) Large weight

Figure 4.16: The mesh appearance with different weighting in the graph-cut algorithm.

parts such as the roof are discarded since the smoothing term is too large.

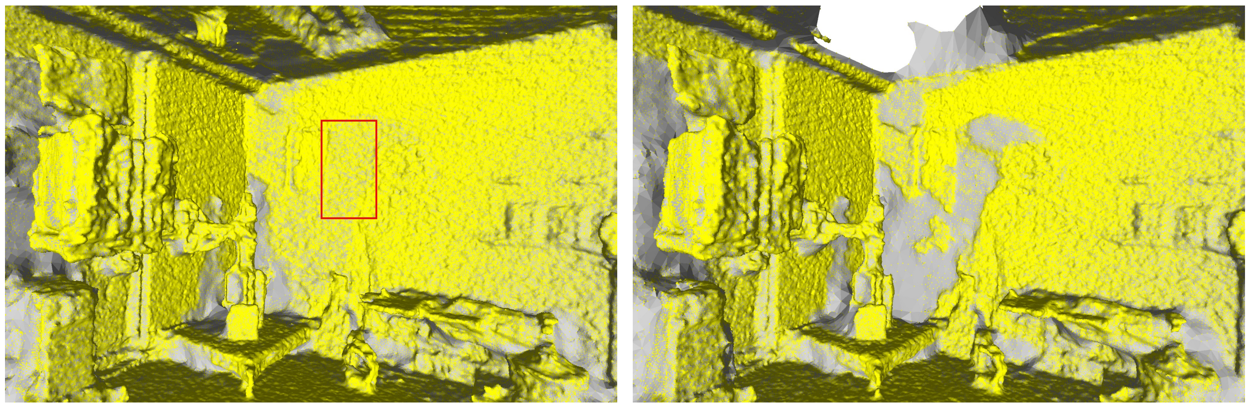

(a) Small smoothing weight (b) Large smoothing weight

Figure 4.17: The figure shows how the smoothing weights affect the mesh result. The yellow points in the figure are vertices of the mesh. In the left figure, the artifacts of LiDAR scan lines are visualized in the red bounding box. After increasing the weight the LiDAR point are discarded in the same area.

Bottlenecks

The long running time is the bottleneck of the proposed pipeline. The graph-cut algorithm in the mesh extraction step has the complexity $O(mn^2)$. Here m is the number of edges and n is the number of the nodes. Therefore, the running time grows quadratically with the increasing number of points in the scene. One solution is to use fewer images to generate the visual dense point cloud and downsample the LiDAR point cloud to have fewer points. The other solution is to look for a local solution instead of the global graph-cut algorithm.

Chapter 5

Conclusions

5.1 Contributions

This thesis presents the following contributions towards joint surface reconstruction combining the LiDAR and the camera.

- It presents a novel algorithm to reconstruct the scene by fusing the geometric information from LiDAR and camera measurements together. The LiDAR points are integrated into the multi-view stereo pipeline in a two-stage manner.
- It shows that LiDAR measurements are helpful for MVS pipeline to get the true scale and achieve the accurate dense depth estimation even in the low-texture area.
- We utilize the graph-cut algorithm after combining the LiDAR and camera measurements in the Delaunay tetrahedralization, which leverages the visibility from both sensors to extract the surface mesh. Besides, our formulation considers the different properties between LiDAR and camera, and the result shows the improvement from our formulation.
- It shows the robustness and accuracy in both of the indoor and outdoor scenes with the proposed pipeline, especially for scenes which are challenging for vision-only or LiDAR-only methods.

5.2 Discussions and Future Work

In the mesh reconstruction stage, we apply the graph-cut algorithm to extract the surface mesh. However, one main drawback of the graph-cut algorithm is that the running time grows quickly with the increasing number of scene points. It is the main bottleneck for the scalability of the reconstruction. Under our experimental setting, when the number of points excess 50 million points, the mesh reconstruction step takes half day to process. Therefore, one potential direction is to make the pipeline scalable. Instead of solving the graph at once, dividing the graph into several parts could be one solution, which will be more tractable when the number of vertices grows.

Another potential direction is to further process the extracted mesh from the graph-cut algorithm. As mentioned in [17], the mesh can be refined using photometric information from the images and it performs well on thin structures. As we showed in the previous sections, these structures are generally difficult to recover in the graph-cut algorithm. The mesh refinement step will be necessary when there are more thin or complicated structures in the environments. Besides the mesh refinement, adding other high-level geometric features into the graph-cut framework can potentially improve the result. As the method stated in [36], it shows impressive results on keeping the edge features in the mesh. Similar algorithms may benefit our pipeline, since the sparsity of LiDAR always results in broken thin structures in the mesh result.

Finally, besides the geometric information, semantics information from scene understanding is a higher level feature to be integrated into the pipeline. If the pipeline can extract the semantics information of the scene, it is more straightforward to select better measurements from LiDAR or camera based on different objects to reconstruct. For instance, visual points can be utilized for compact objects, while LiDAR points may be adopted for structural objects such as walls.

Bibliography

[1] 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. 1

[2] 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. 1

[3] J. L. Schönberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 6

[4] “OpenMVS.” [Online]. Available: https://github.com/cdcseacave/openMVS 1, 2, 6, 14, 15, 16, 20, 21, 22, 23, 25, 26, 29

[5] 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. 2, 22

[6] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping in real-time,” in *Robotics: Science and Systems (RSS)*, 2014. 2, 5, 6, 14

[7] L. Zhou, Z. Li, and M. Kaess, “Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences,” in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018. 2, 41, 42, 43, 47

[8] H. Ye, Y. Chen, and M. Liu, “Tightly coupled 3D Lidar inertial odometry and mapping,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2019. 5

[9] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized Lidar odometry and mapping on variable terrain,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct 2018, pp. 4758–4765. 5

[10] C. L. Gentil, T. Vidal-Calleja, and S. Huang, “IN2LAAMA: Inertial Lidar localisation autocalibration and mapping,” *arXiv preprint arXiv:1905.09517*, 2019. 5

[11] J. Behley and C. Stachniss, “Efficient surfel-based slam using 3D laser range data in urban environments,” in *Robotics: Science and Systems (RSS)*, 2018. 5

[12] C. Park, P. Moghadam, S. Kim, A. Elfes, C. Fookes, and S. Sridharan, “Elastic LiDAR Fusion: Dense map-centric continuous-time slam,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, May 2018, pp. 1206–1213. 5

[13] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, “SegMap: 3D segment mapping using data-driven descriptors,” in *Robotics: Science and Systems (RSS)*, 2018. 5

[14] F. Gao, W. Wu, W. Gao, and S. Shen, “Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments,” *J. of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21842 5

[15] 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, 7, 21, 23, 25, 26

[16] M. Goesele, B. Curless, and S. M. Seitz, “Multi-view stereo revisited,” in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, Jun. 2006, pp. 2402–2409. 6

[17] 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. 6, 17, 36

[18] T. Schöps, J. L. Schönberger, S. Galliani, T. Sattler, K. Schindler, M. Pollefeys, and A. Geiger, “A multi-view stereo benchmark with high-resolution images and multi-camera videos,” in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2538–2547. 6

[19] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan, “MVSNet: Depth inference for unstructured multi-view stereo,” *Eur. Conf. on Computer Vision (ECCV)*, 2018. 7

[20] 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. 7

[21] 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. 7

[22] 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. 7

[23] 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. 7

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009. 9

[25] L. Ford and D. Fulkerson, “Maximal flow through a network,” *Canadian journal of Mathematics*, vol. 8, no. 3, pp. 399–404, 1956. 9

[26] 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. 9

[27] A. V. Goldberg, S. Hed, H. Kaplan, R. E. Tarjan, and R. F. Werneck, "Maximum flows by incremental breadth-first search," in *Algorithms – ESA 2011*, C. Demetrescu and M. M. Halldórsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 457–468. 9

[28] A. V. Goldberg, S. Hed, H. Kaplan, P. Kohli, R. E. Tarjan, and R. F. Werneck, "Faster and more dynamic maximum flow by incremental breadth-first search," in *Algorithms - ESA 2015*, N. Bansal and I. Finocchi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 619–630. 9

[29] 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 9, 11, 17, 21, 23, 24, 26, 27, 28, 29

[30] 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. 9, 11, 28

[31] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov 2001. 11

[32] 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. 11, 14, 17, 28

[33] 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. 14, 15

[34] M. Bleyer, C. Rhemann, and C. Rother, "PatchMatch Stereo – stereo matching with slanted support windows," in *British Machine Vision Conf. (BMVC)*, Jan. 2011. 14, 15

[35] "CGAL: Computational Geometry Algorithms Library." [Online]. Available: http://www.cgal.org 21

[36] 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. 36

[37] L. Zhou and Z. Deng, "Extrinsic calibration of a camera and a Lidar based on decoupling the rotation from the translation," in *2012 IEEE Intelligent Vehicles Symposium*, June 2012, pp. 642–648. 41

[38] F. Vasconcelos, J. P. Barreto, and U. Nunes, "A minimal solution for the extrinsic calibration of a camera and a laser-rangefinder," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2097–2107, Nov 2012. 41

[39] R. Unnikrishnan and M. Hebert, "Fast extrinsic calibration of a laser rangefinder to a camera," 2005. 41, 44, 47

[40] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Extrinsic calibration of a 3D laser scanner and an omnidirectional camera," *IFAC Proceedings Volumes*, vol. 43, no. 16, pp.

336–341, 2010. 41

[41] F. M. Mirzaei, D. G. Kottas, and S. I. Roumeliotis, “3D LIDAR–camera intrinsic and extrinsic calibration: Identifiability and analytical least-squares-based initialization,” *The International Journal of Robotics Research*, vol. 31, no. 4, pp. 452–467, 2012. 41

[42] R. G. von Gioi, J. Jakubowicz, J. M. Morel, and G. Randall, “LSD: A fast line segment detector with a false detection control,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 32, no. 4, pp. 722–732, April 2010. 42

[43] W. Dong and V. Isler, “A novel method for the extrinsic calibration of a 2-D laser-rangefinder and a camera,” in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5104–5109. 45

Appendix A

Extrinsics Calibration of LiDAR and Camera

To fuse LiDAR and camera measurements together, an accurate calibration between the two sensors is necessary. The extrinsic parameters contain the rotation and translation between the two sensors. To calibrate the extrinsics parameters, correspondences are needed between LiDAR measurements and camera images. To easily obtain correspondences, checkerboards are widely used in the literature [37, 38, 39, 40, 41]. We compare the results from two different methods which use different correspondences to constrain the calibration. The baseline method [39] solves the transformation by finding the plane correspondences from LiDAR scans to the images, which requires at least 3 poses. In our recent improved method [7], besides the plane constraints, line correspondences are incorporated to decrease the number of needed poses for calibration to 1. In this chapter we detail the two schemes and show comparisons through experiments.

A.1 Calibration with Plane Correspondences

For calibration, we have the actual size of the checkerboard, several LiDAR scans and their corresponding images with the same time stamp. The plane correspondence refers to the checkerboard in the calibration, and the plane is represented by a normal vector **n** and a distance d. For LiDAR scans, it is straightforward to extract the plane parameters since LiDAR provides 3D measurements directly. For camera images, since we have the intrinsics parameters and the size of the box in the checkerboard, the rotation and translation are known from the board to the camera center, and they can be converted to the plane parameters. Rotation $\mathbf{R}_L^C$ and translation $\mathbf{t}_L^C$ represent the calibration. Super script L and C represent the LiDAR frame and the camera frame respectively. And the subscript i is the pose index of the checkerboard. m is the number of LiDAR points on the i-th scan from index 1 to N_i. Therefore, we have the constraint equations:

$$\mathbf{R}_L^C \mathbf{n}_i^L = \mathbf{n}_i^C \tag{A.1}$$

$$\mathbf{n}_i^C \cdot \left(\mathbf{R}_L^C \mathbf{P}_{im}^L + \mathbf{t}\right) + d_i^C = 0 \tag{A.2}$$

By solving above equations, we obtain the initial estimation of $\mathbf{R}_L^C$ and $\mathbf{t}_L^C$, then we jointly optimize them by minimizing the cost function:

$$\left(\hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right) = \underset{\mathbf{R}_L^C, \mathbf{t}_L^C}{\arg\min} \sum_{i=1}^{N} \sum_{m=1}^{N_i} \left\| \mathbf{n}_i^C \cdot \left(\mathbf{R}_L^C \mathbf{P}_{im}^L + \mathbf{t}\right) + d_i^C \right\|^2 \tag{A.3}$$

We perform Levenberg-Marquardt method to solve the nonlinear optimization and obtain the final calibration result.

A.2 Calibration with Plane and Line Correspondences

Besides the plane of checkerboard, four edges of the checkerboard are also proposed to be adopted as the correspondences [7]. Extracting lines is more tricky than calculating the plane parameters. In the LiDAR scans, after we extract all the points on the plane, we find the end points of LiDAR scan lines (16 lines or fewer here because Velodyne VLP-16 is used in our setting) then fit them into 4 boundary lines. For images, LSD (Line Segment Detection) algorithm [42] is performed to detect a set of line segments in the image, and we select the 4 boundaries based on their position to the checker board. In terms of notations, besides the defined ones in Section A.1, notations relevant to lines are stated as following. Lines L_{ij} $(j = 1, 2, 3, 4)$ is the 4 boundaries of the i-th pose, represented as $[\mathbf{d}_{ij}; \mathbf{p}_{ij}]$. **d** is the direction of the boundary and **p** is a point on the same boundary. For LiDAR boundaries, since there are few points on each boundary,we represent the points on L_{ij}^L as $\mathbf{Q}_{ijk}^L (k = 1, ..., K_{ij})$, in which K_{ij} means the total number of points on one boundary. Therefore, besides the plane constraints, we have the equations for boundaries as well.

$$\mathbf{R}_L^C \mathbf{d}_{ij}^L = \mathbf{d}_{ij}^C \tag{A.4}$$

$$\left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \left(\mathbf{R}_L^C \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^C + \mathbf{t}_L^C\right) = \mathbf{0}_{3\times 1} \tag{A.5}$$

The Equation A.5 indicates that the LiDAR points on the boundary should have distance 0 to the corresponding boundary in the image after transformation. The $\left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right)$ part is the projection to the direction of the boundary in the image. To solve the equations together with the plane constraints, the initial estimation of $\mathbf{R}_L^C$ is obtained by stacking the Equation A.1 and A.4 together and solving the linear system. And estimated $\mathbf{t}_L^C$ can be obtained using following equations from Equation A.2 and A.5.

$$\mathbf{n}_i^C \cdot \mathbf{t}_L^C = -\mathbf{n}_i^C \cdot \mathbf{R}_L^C \bar{\mathbf{P}}_i^L - d_i^C \tag{A.6}$$

$$\left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \mathbf{t}_L^C = -\left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \left(\mathbf{R}_L^C \bar{\mathbf{Q}}_{ij}^L - \mathbf{p}_{ij}^C\right) \tag{A.7}$$

In above equations, $\bar{\mathbf{P}}_i^L$ and $\bar{\mathbf{Q}}_{ij}^L$ are the mean point of the all points on the plane and on the boundary respectively. Having obtained the initial estimation of the transformation, we further optimize them using LM method with the following cost function.

$$\begin{aligned}\left(\hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right) = \underset{\mathbf{R}_L^C, \mathbf{t}_L^C}{\arg\min} \sum_{i=1}^{N} \frac{1}{N_i} \sum_{m=1}^{N_i} \left\| \mathbf{n}_i^C \cdot \left(\mathbf{R}_L^C \mathbf{P}_{im}^L + \mathbf{t}\right) + d_i^C \right\|^2 + \\ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \left\| \left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \left(\mathbf{R}_L^C \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^C + \mathbf{t}_L^C\right) \right\|^2 \end{aligned} \tag{A.8}$$

In [7], the scale factor is also introduced in consideration of the measuring error in the checkerboard size. In that case, we need to add the scale factor into Equation A.7 and A.6, and the nonlinear optimization as well. Therefore, we rewrite the equations as following:

$$\mathbf{n}_i^C \cdot \mathbf{t}_L^C = -\mathbf{n}_i^C \cdot \mathbf{R}_L^C \bar{\mathbf{P}}_i^L s - d_i^C \tag{A.9}$$

$$\left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \mathbf{t}_L^C = -\left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \left(\mathbf{R}_L^C \bar{\mathbf{Q}}_{ij}^L s - \mathbf{p}_{ij}^C\right) \tag{A.10}$$

$$\begin{aligned}\left(s, \hat{\mathbf{R}}_L^C, \hat{\mathbf{t}}_L^C\right) = \underset{\mathbf{R}_L^C, \mathbf{t}_L^C}{\arg\min} \sum_{i=1}^{N} \frac{1}{N_i} \sum_{m=1}^{N_i} \left\| \mathbf{n}_i^C \cdot \left(s\mathbf{R}_L^C \mathbf{P}_{im}^L + \mathbf{t}\right) + d_i^C \right\|^2 + \\ \sum_{i=1}^{N} \sum_{j=1}^{4} \frac{1}{K_{ij}} \sum_{k=1}^{K_{ij}} \left\| \left(\mathbf{I} - \mathbf{d}_{ij}^C \left(\mathbf{d}_{ij}^C\right)^T\right) \left(s\mathbf{R}_L^C \mathbf{Q}_{ijk}^L - \mathbf{p}_{ij}^C + \mathbf{t}_L^C\right) \right\|^2 \end{aligned} \tag{A.11}$$

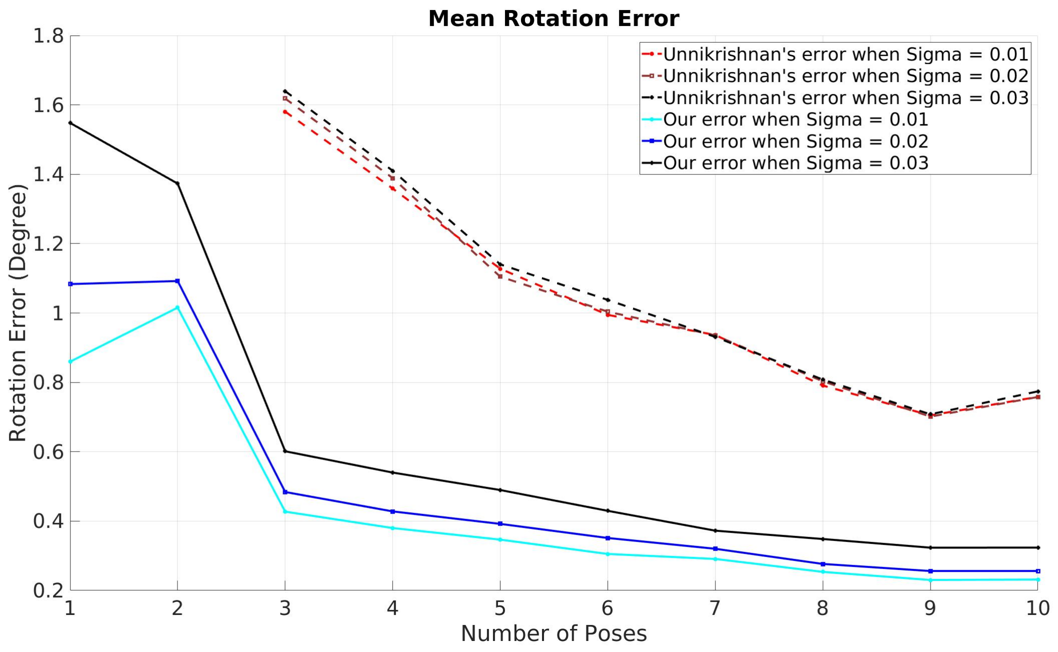
A.3 Experiments and Results

A.3.1 Synthetic Results

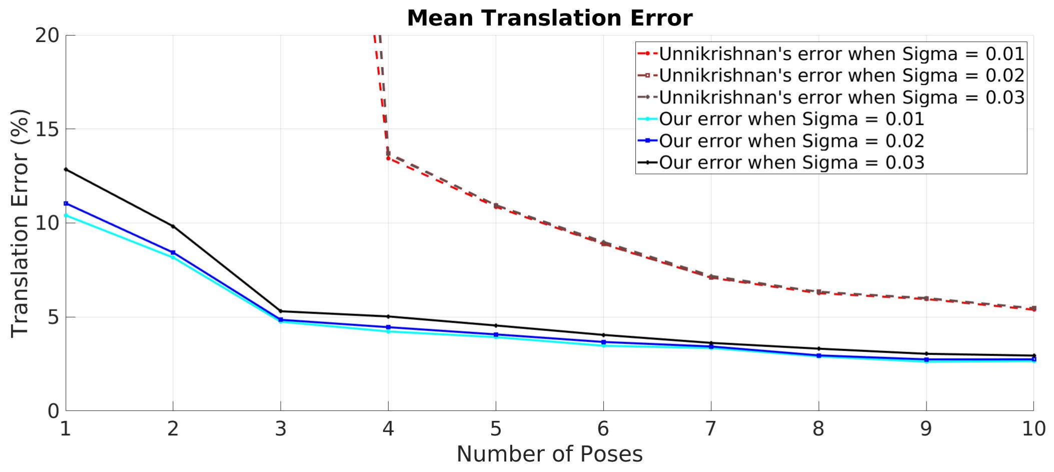
In the simulation, we randomly generate the configuration of LiDAR, camera and checkerboard poses. Specifically, the roll, pitch and yaw angle of the camera are within $\pm 45°$, and the translation elements are uniformly distributed in $\pm 0.3m$ w.r.t the LiDAR. For the checkerboard frame, its x and y components of the translation are within $\pm 0.5m$, its z component is within $[1.5m, 2.5m]$, and its orientation is within $\pm 45°$ relative to the camera. The checkerboard pose is generated in the view of the camera.

We add zero mean Gaussian noise to LiDAR and camera measurements to test the performance of different algorithms under various noise levels. The standard deviation of the LiDAR noise is set to $1cm$, $2cm$ and $3cm$, while the standard deviation of the image noise is fixed at 1 pixel. The number of checkerboard poses N is within $[1, 10]$. We run each algorithm 200 times for each N.

Fig. A.1 gives the results. It is clear that our algorithm has smaller rotation and translation errors under different noise levels compared to [39]. Additionally, our rotation and translation error are around 1.5° and 12% when only one pose is used and the standard deviation of the LiDAR noise is $3cm$. It verifies that our algorithm is able to provide accurate result using a single snapshot under large noise.



(a) Rotation result



(b) Translation result

Figure A.1: The result of calibration from the synthetic dataset

A.3.2 Real-world Results

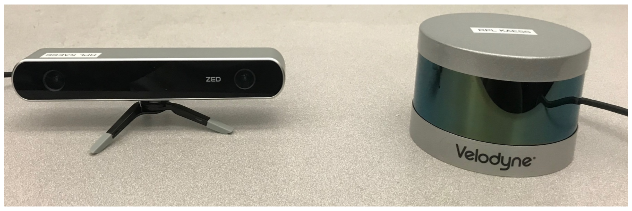


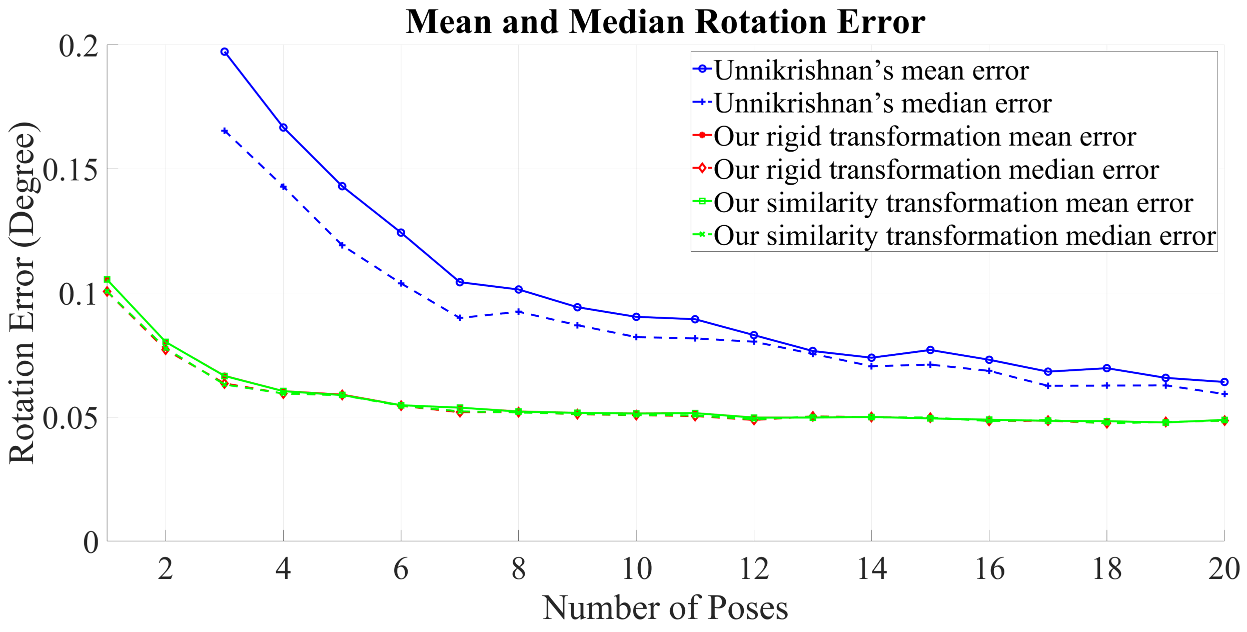
Figure A.2: Velodyne VLP-16 LiDAR and ZED stereo camera used in our experiments.

We use a Velodyne VLP-16 LiDAR and a ZED stereo camera to verify our algorithm, as shown in Fig. A.2. The Velodyne VLP-16 LiDAR has 16 scan lines, $\pm 3cm$ range error and 360 degree horizontal and $\pm 15^\circ$ vertical field of view. The threshold of the RANSAC algorithm is set to $3cm$ for plane fitting in the experiments. The ZED stereo camera has about 12cm baseline and 1280×720 resolution.

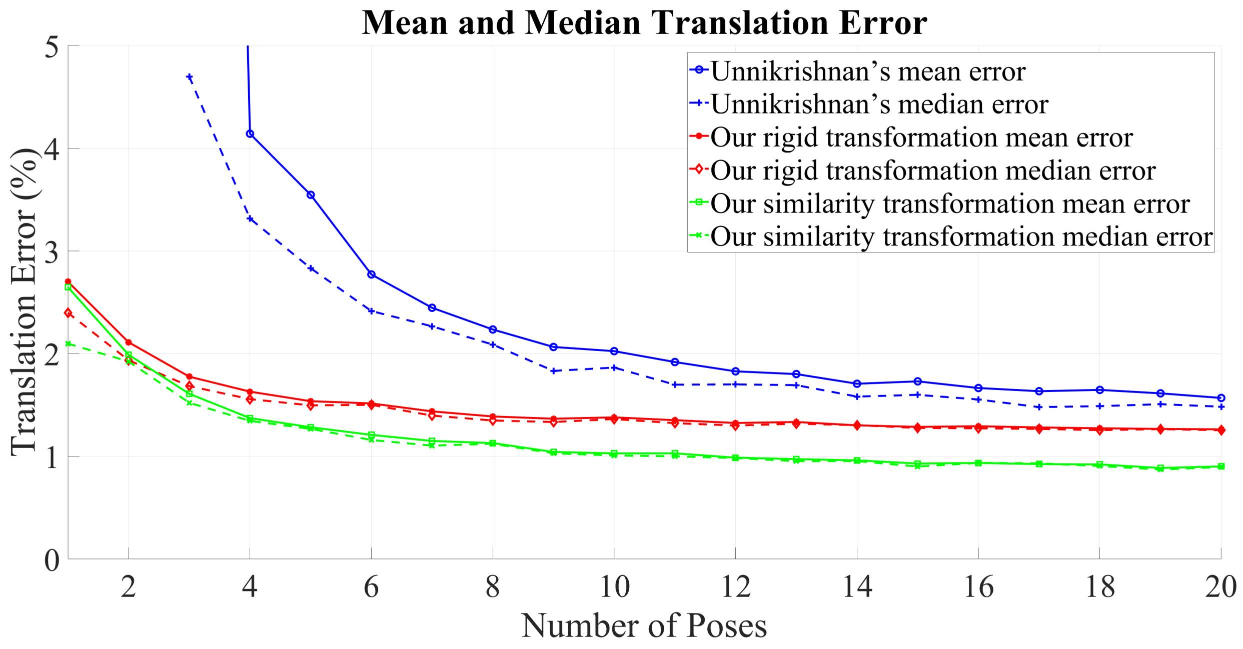
As we do not have the ground truth of the extrinsic parameters of the LiDAR and the camera, we use the extrinsic parameters of the stereo camera to evaluate the performance of the algorithm as in [43]. Specifically, we estimate the extrinsic parameters of the LiDAR and the left camera $(\hat{\mathbf{R}}_L^{C_l}, \hat{\mathbf{t}}_L^{C_l})$, and the extrinsic parameters of the LiDAR and right camera $(\hat{\mathbf{R}}_L^{C_r}, \hat{\mathbf{t}}_L^{C_r})$, respectively. Then we compute the relative pose $(\hat{\mathbf{R}}_s, \hat{\mathbf{t}}_s)$ between the left and right camera from $(\hat{\mathbf{R}}_L^{C_l}, \hat{\mathbf{t}}_L^{C_l})$ and $(\hat{\mathbf{R}}_L^{C_r}, \hat{\mathbf{t}}_L^{C_r})$. $(\hat{\mathbf{R}}_s, \hat{\mathbf{t}}_s)$ is compared with the stereo extrinsic parameters $(\mathbf{R}_s, \mathbf{t}_s)$ calculated by the MATLAB tool box. We also calculate the similarity transformation, and compute the error as the rigid transformation.

We collected 32 LiDAR and image pairs for the experiment. N LiDAR and image pairs are randomly chosen from them. N is within $[1, 25]$ for our algorithm and $[3, 25]$ for Unnikrishnan's algorithm. For each $N \in [2, 25]$, we run the experiment 200 times. For $N = 1$, we estimate the extrinsic parameters for all the 32 poses of the checkerboard. Fig. A.3 gives the result. Our algorithm yields more accurate results. The result of our algorithm from one pose is comparable to the result of Unnikrishnan's algorithm when 6 poses are used. Using similarity transformation, we get very similar rotation error, but smaller translation error than using rigid transformation. This is because there exists inevitable measurement error for the checkerboard size. These measurement errors mainly affect the estimation of the translation rather than the rotation, since the rotation matrix can be decoupled from the scale factor. Unnikrishnan's algorithm has large translation estimation error when $N = 3$. This is because there exist some planes with similar orientations. These planes are degenerate for Unnikrishnan's algorithm, but they are valid for our algorithm. The 3D line constraints significantly increase the diversity of the measurements. Therefore, our algorithm is more robust to the configuration of the poses and can get better result with fewer number of poses.

For the calibration of the system applied in the thesis, the calibration result is shown in Fig. A.4.

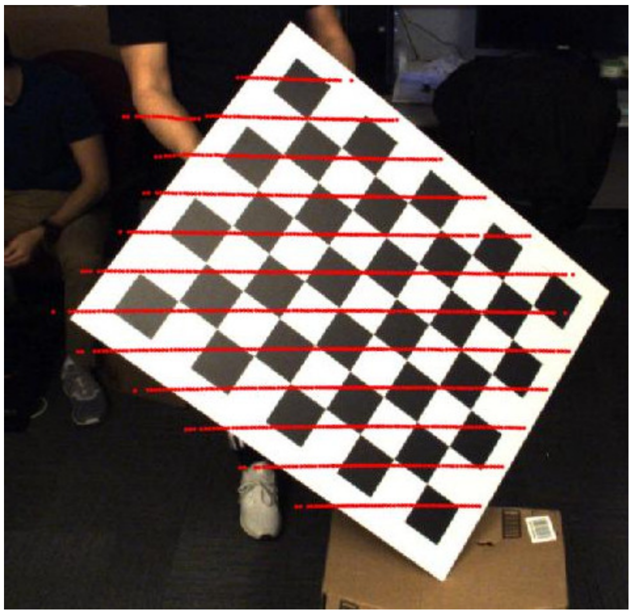


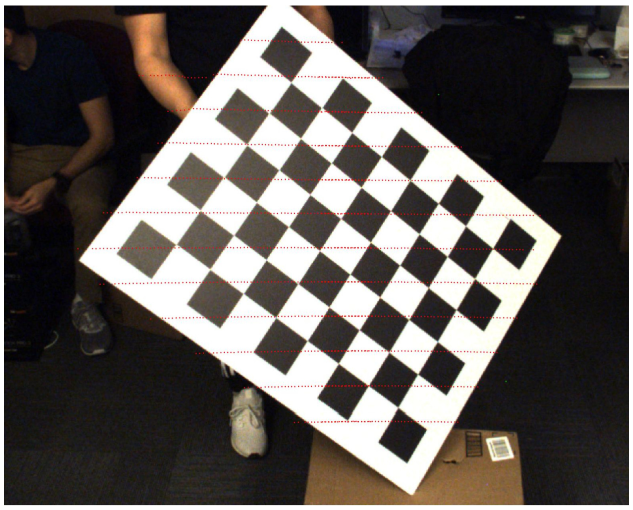
(a) Rotation result



(b) Translation result

Figure A.3: The result of calibration from the real-world dataset.


(a) Result from [39]


(b) Result from [7]

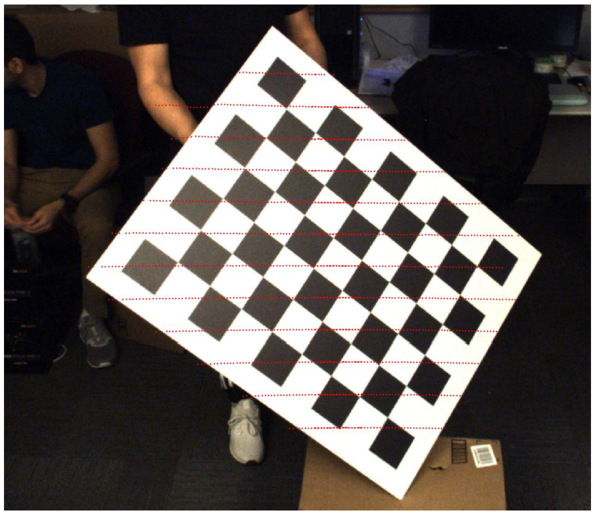

(c) Result from [7] with similarity transformation

Figure A.4: The visualization of the calibration methods on the device used for reconstruction.